UNIVERSITÀ DEGLI STUDI DI NAPOLI "FEDERICO II"

CORSO DI DOTTORATO IN SCIENZE COMPUTAZIONALI E INFORMATICHE

XXV CICLO

A CONTENT MANAGEMENT SYSTEM FOR SPATIO-TEMPORAL DATA: TADAIMA

A Case of Study in Cultural Heritage field

TESI DI DOTTORATO DI

VINCENZA ANNA LEANO

31/03/2013

COORDINATORE DEL DOTTORATO

PROF.SA GIOCONDA MOSCARDIELLO

TUTOR ACCADEMICO

PROF. ERNESTO BURATTINI

REFERENTI SCIENTIFICI:

PROF. FRANCESCO CUTUGNO

PROF. ADRIANO PERON

A mio padre Pasquale

“e figl se vasn n’suonn…”

Abstract

This thesis focuses on spatio-temporal data modeling and visualization with an application case of study in the Cultural Heritage field.

Spatio-temporal data visualization assumes an important role presenting data to users. Offering a synchronized view on three dimensions of data (i.e. descriptive, temporal and spatial) helps users in their knowledge discovery process. In Cultural Heritage field, time assumes an important role to explore data. The same timeline could be viewed in different thematic contexts, temporal domain could be stratified and the time reference could be qualitative and imprecise. Managing this kind of features improves the ability of the users to recognize patterns in data.

This thesis presents a framework oriented to the manipulation of Spatio-Temporal data with a particular attention to the temporal specification needs of the Cultural Heritage context, producing a prototype of a Content Management System (CMS). The proposed framework exploits the RDF technology for definition and manipulation of (meta-) data and adopts the OGC standards and open source technologies (PostGIS, Geoserver and Openlayers) for encoding, representing and retrieving spatial information. The available Spatio-Temporal metaphors are parametric, so users can personalize them depending on the specific application context and needs. A real case study in the Cultural Heritage field, concerning spatio-temporal information contained in literary Latin and Greek texts referred to the geographic area of *Campi Flegrei* (Naples, Italy), describes the framework functionalities.

Contents

List of Figures

List of Tables

Introduction

Studies show that 80% of data stored in a database has a spatial reference (Franklin, 1992) and it could be reasonable to think that this proportion holds also for the temporal reference. The current state of art has always highlighted how this two features are "special" (Egenhofer, 1993), defining ad hoc representation models, exploration operators and visualization metaphors to better render them to the user.

From a theoretical point of view, a spatio-temporal data model has to be designed with the purpose of making representable all (or a lot of) the possible aspects of temporal, spatial and spatio-temporal feature, offering operators to manipulate and exploit them (Yuan, 1996). Many applications managing spatio-temporal data, based on desktop or on web architecture, have been proposed e.g. (Andrienko et al., 2007), (MacEachren et al., 2004) etc. Often these applications focus on particular aspects of spatio-temporal data, using proprietary tools for the representation or the visualization of data. The first goal of this thesis is to design a general spatio-temporal data model able to manage all the aspects of a spatio-temporal phenomenon based on existing standards and relying on a flexible architecture independent from proprietary visualization and storage tools.

The temporal domain assumes a strategically role in several fields, like for example the *Cultural Heritage* one. Experts in this field would organize the temporal domain in different ways according to the context under analysis (i.e. "*History of Naples*" or "*History of Literature*"). Each context might have different granularity levels (for example "*History of Naples*" could have levels: "*Domination*", "*Empire*", "*Battle*", "*Important Events*"). The elements of this timeline might not have a precise quantitative temporal dating, but only topological relation with other events in the same context. From the visualization point of view, mapping this complex temporal structure into common temporal metaphor like the one-dimensional time-bar representation, would cause a loss of information and could compromise the data exploration process. This work offers the possibility of modeling such a particular domain together with an adequate

visualization metaphor to explore it.

The requirement of organizing a domain in thematic context having different granularity levels applies to the spatial one too. Furthermore, the evolution of spatial domain elements over time, and the correlation of ancient places with nowadays spatial locations, assumes a strategically role in user knowledge discovering process. Users can reference an object in space in several manners, for example, they can give absolute georeferences by providing coordinate or drawing geometry on a map, or they can use a semantic link, referring to a place by its name. As for the temporal domain concerns, it can happen that users have not precise information about the spatial reference, but only incomplete information starting from a well-known place, for example *"10 km north of Naples"*, *"inside an area"*, etc. This work provides a spatial model fully based on existing encoding and operators standards able to manage those spatial domain particular needs.

Several different forms of spatio-temporal data types are available in real world and the current state of art offers many ways to classify them (e.g. see (Asproth et al., 1995), (Kisilevich, 2005), (Nadi & Mahmoud, 2003)). For each spatio-temporal dataset several visualization metaphors have been proposed (Andrienko et al., 2003) in order to improve users knowledge discovering process. Many of the currently available spatio-temporal visualization systems focus on the spatial and descriptive representation of the spatio-temporal phenomena, limiting the representation of the temporal dimension to a numeric value or to a one-dimensional time-bar (timeline). This thesis provides a web interface that allows users to explore data in their spatial, temporal and descriptive dimensions in an independent and synchronized way.

A spatio-temporal application has to handle a data model able of managing temporal, spatial and spatio-temporal domains, and has to offer adequate visualization metaphor in order to represents the managed features.

Building an application that allows users to model, store and interact with these features and their spatial, temporal and spatio-temporal dimensions implies dealing with a set of technologies like web servers, spatial databases,

geobrowsers, and temporal data structures that common users may not master.

The underlying idea proposed in this thesis consists on providing a framework that allows users to model spatial, temporal and spatio-temporal features regardless their specific application domain. Starting from the triad model (Peuquet, 1994) we will consider our data as just having a set of properties, regardless their semantic meaning.

The RDF data model proposed by W3C seems to suit perfectly this purpose. In RDF, a resource is represented by a triple <*Subject, Predicate, Object*>. Each triple represents a statement of a relationship between the things denoted by the nodes that it links (W3C, 2004). Therefore, a triple represents a link (*predicate*) between two nodes (*subject*, *object*); the set of all triples generates an *RDF Graph.*

An object/feature instance in the triad model can be seen as a set of triples where the subject is the feature itself, the predicates are the different properties (spatial, temporal and descriptive) and the objects are the corresponding property values.

Using RDF Schema concepts one can model a generic feature by defining classes and properties in a way very similar to the Object-Oriented design.

One of the more important advantages of using RDF data model is that the model and the data have the same data structure (*graph*) and one can query both of them with the same query languages (*SPARQL*), even if the specific data model is unknown. This is a very important feature in order to be independent from the specific domain application, because we can refer to a property as a general predicate, which will be instantiated and personalized by user needs.

SPARQL does not (still) support spatial and temporal operators. Some works introduce spatial and temporal query algebra, but none of them is nowadays a W3C recommendation. In addition, the proposed model does not provide support to the thematic and hierarchical stratification of the spatial and temporal domains proposed in this work.

To overcome this limitation this thesis provides a data model specification for spatial, temporal and spatio-temporal domains. The “semantic” entities are modeled using RDF/OWL structures while the specific operators are implemented

using ad hoc data structures and/or defined standards.

In order to allow users to define a generic *Feature*, an RDF(S)/OWL meta-model is designed, that allows the definition of spatial, temporal, spatio-temporal and descriptive properties.

The CMS back-end provides user-friendly methods to create features and model their properties. The creation process allows also importing standard RDF/OWL ontology and the possibility of personalizing some predefined Cultural Heritage class of feature. Users can choose between different types of views and visual metaphors for visualizing their data in the front-end.

The CMS front-end incorporates and synchronizes spatial, temporal, and descriptive views in an integrated and extensible way. It provides a new interaction metaphor that exploits the hierarchical and stratified temporal domain defined and presented in our work (Cerasuolo et al., 2012). The front-end consists in a configurable interface that allows to independently interact with the three dimensions of data (spatial, temporal and descriptive) and it offers spatiotemporal visualization with a high level of personalization. Users can choose between different types of views and visual metaphors to see, filter and compare objects into an area handled by a geobrowser, and activate or disable information layers of their interest. By interacting with a map, users can perform spatial queries to obtain information about the referencing objects and their content. Users can visualize the complex temporal structure in an intuitive way and they can perform complex temporal queries by clicking on the desired temporal element.

The CMS front-end extends our work presented in (Cutugno et al., 2012), i.e. a framework aiming at merging a spatio-temporal data model in a web-architecture which can be compliant with existing standards and independent from data storage and visualization tools. The framework defines a flexible three-tier architecture for web applications that shows low coupling among tiers and uses standard exchange data formats like WFS, KML, GML to guarantee independence from storage and visualization tools.

As an instance, we propose a case study adopting the above outlined approach aimed at promoting a rich archaeological site in the area of *Campi*

Flegrei in the neighbourhood of Naples. In particular, the CMS prototype manages at the moment about 200 literary excerpts from Greek and Latin authors annotated with their spatial (places in the *Campi Flegrei* area) and temporal references. The temporal domain is composed of about 400 temporal period-events structured in three thematic contexts ("*History of Campi Flegrei*", "*History Events*", "*Authors' Life*"). Each thematic context is arranged in 5 granularity layers ("*Epoch*", "*Ages*", "*Empire/Domination*", "*Political and Historical Events*", "*Important Events*").

Outline

The core of this thesis is divided in two parts. **PART I** introduces some relevant background concepts on standard spatial models and operators and on RDF technology, together with an overview of the most relevant related work. In particular Part I is composed by:

- **CHAPTER 1**: this chapter introduces the *Open Geospatial Consortium* (OGC) standards to *encode spatial data* (GML, KML) and the *spatial web services* that implements the standard spatial operators (WMS, WFS, WCS). It also describes the suggested *architecture for web application* and gives an overview of the most used *GIS tools*.
- **CHAPTER 2**: this chapter shows some background concepts on *Resource Description Framework* (RDF) data modelling, used for the CMS data model, and its query language *SPARQL*. An overview on *Semantic Web* and *OWL* it is also provided.
- **CHAPTER 3**: the topics of this chapter are *related work* on spatio-temporal *data modelling*, spatio-temporal *visualization* and spatial, temporal and spatio-temporal RDF/OWL *ontologies*.

Part II describes the proposed work in terms of data model, prototype and case of study:

- **CHAPTER 4**: this chapter shows the data-model the CMS relies on. In particular in this chapter we will define: the *abstract meta-model* that

allows users to describe features having spatial, temporal, spatio-temporal and descriptive properties; the *temporal domain*, that uses an original model to handle user defined contexts, granularities and qualitative temporal references; the *spatial domain*, that uses a model inspired by the OGC standards able to manage semantic, absolute and uncertain spatial references; the *integration of space and time properties* that manages the various types of spatio-temporal phenomena.

- **CHAPTER 5**: this chapter illustrates the CMS implemented prototype by showing the Back-End functions for creating a model and personalizing the visualization, and the Front-End interface that allows user to independently interact with the three dimensions of data in a synchronized way and offers a new visual metaphor to explore the hierarchical and stratified temporal domain.
- **CHAPTER 6**: this chapter describes the application of the CMS prototype to a real case of study in Cultural Heritage. It shows a deliverable of the project named TRACCIA supported by “FARO – Università degli Studi di Napoli Federico II – Polo delle Scienze Umane e Sociali”, documents a joint work with Latin philologists of the department "Filologia Classica F. Arnaldi". The aim of the project was to find, document and give public access to the literary and historical evidences of typical agricultural products in the area of *Campi Flegrei*

This thesis is closed by some *Conclusions and Future work* remarks.

Part I
Background and Related Work

A Content Management System for Spatio-Temporal Data: Tadaima

Chapter 1 OGC and Spatial Standard

The availability of tools dealing with spatial data and able to perform spatial operations is more and more increasing. With the advent of internet technologies, also GIS applications moved to this domain. When geographic data are shared between organizations dealing with different applications, there might be a heterogeneity problem if the organizations use different GIS platforms, hence, producing different digital formats of the data.

The Open Geospatial Consortium (OGC) is an international voluntary standards organization, which defines open standards for geospatial content and services and suggests best practice for Web-GIS architecture. In the rest of the chapter the encoding spatial data format, the OGC Web services, the Web GIS Architecture and some geospatial tools will be exploited.

1.1 Encoding Spatial Data

Today, the internet is the main platform for data sharing. Thus, to share and integrate geographic data in the internet environment requires a standard data format, which is interoperable, extensible and suitable for internet technology.

OGC, whose mission is to address the lack of interoperability between systems that process geo-spatial data, has developed encoding and interface standards to satisfy syntactic interoperability among geospatial web services. The main encoding standards are Geography Markup Language (GML) (Cox et al., 2002) and KML (Wilson, 2008) described in the following subsections.

1.1.1 GML

The Geographic Markup Language (GML) is an XML encoding that implements the standard ISO 19118 for the transport and storage of geographic information modeled according to the conceptual modeling framework used in the ISO 19100 series and including both the spatial and non-spatial properties of geographic features (Cox et al., 2002).

GML documents, like XML, are both human and machine-readable. Therefore,

they are easier to understand and maintain than proprietary binary formats.

GML separates content of geographic data from its presentation. GML mainly describes the structure of geographic data without regard to how the data can be presented to a human reader.

OGC initially developed GML 1.0, which was based on a combination of XML DTDs and the Resource Description Framework (RDF). GML 2.0, which replaces GML 1.0, was developed and adopted in March 2001 by OGC. It is entirely based on XML Schema. Adoption of XML Schema in GML incorporates support for type inheritance, distributed schema integration, and namespaces. GML 2.0 is based on linear geometry, it does support coordinates to be specified in three dimensions, but it does not provide direct support for three-dimensional geometric constructs. GML 3.0 has been extended to represent geo-spatial phenomena in addition to simple 2D linear features, including features with complex, non-linear, 3D geometry, features with 2D topology, features with temporal properties, dynamic features, coverage and observations. It also provides more explicit support for properties of features and other objects having complex value.

1.1.1.1 GML Feature

GML is based on the geographic model developed by the OGC, which describes the world in terms of geographic entities called features. This geographic model is based on the OGC Abstract Specification (Open Geospatial Consortium, 2003), which defines a geographic feature as “an abstraction of a real world phenomenon, it is a geographic feature if it is associated with a location relative to the Earth”.

Thus, real world phenomena are represented digitally as a set of features. The state of a feature is defined by a set of properties, where each property has a name, value and type descriptions. Geographic features are those features whose properties may be geometry-valued.

In GML a *feature* is represented as an XML element. The name of the feature element indicates the *Feature Type*. The content of a feature element

is a set of elements, which describes the feature in terms of a set of properties. Each child element of the feature element is a *property*. The name of the property element indicates the property type.

The value of a property is given in-line by the content of the property element, or by-reference as the value of a resource identified in a link carried as an XML attribute of the property element. If the in-line form is used, then the content may be a literal (a number, text, etc.), or may be structured using XML elements, but no assumptions can be made about the structure of the value of a property. In some cases, the value of a property of feature may be another feature.

Properties of a feature may be simple properties or geometric properties. Properties with simple types (e.g., integer, string, float, boolean) are collectively known as simple properties and the properties that are geometry-valued, are known as geometric properties. A feature can have multiple simple properties as well as multiple geometric properties. A feature can be composed of other features. Such a feature is termed as a feature collection. A feature collection has a feature type and thus may have its own distinct properties, in addition to the features it contains.

1.1.1.2 GML Geometry Elements

In accordance with the OGC simple feature model (OGC,2006) in order to express a feature having a spatial property, GML provides the encoding of geometric elements that represents a feature in the spatial domain. The GML encoding of a generic geometry is showed in Fig. 1.

Geometry represent the way a feature is linked to the spatial domain, this link can be represented by one of the classes showed in in Fig. 2 and depicted as follows:

- ***Point***: is defined by a single coordinate tuple.
- ***LineString***: is a special curve that consists of a single segment with linear interpolation. It is defined by two or more coordinate tuples, with linear interpolation between them.

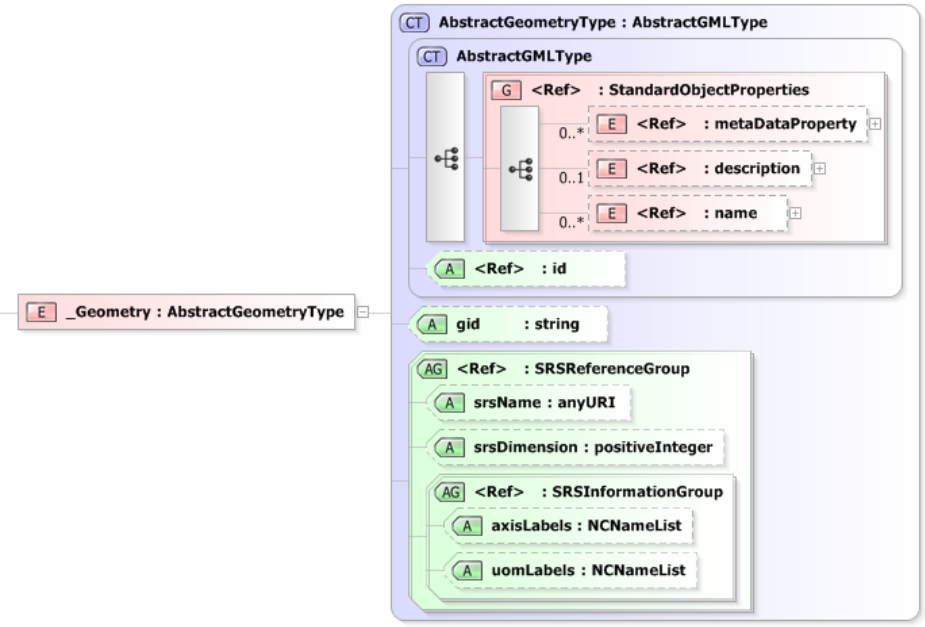

Fig. 1 GML Geometry Encoding

- ***LinearRing:*** is a closed, piece-wise linear path it is defined by four or more coordinate tuples, with linear interpolation between them; the first and last coordinates shall be coincident so that they can form a ring.
- ***Polygon:*** is a connected surface of which the boundary is a set of LinearRings. The boundaries are characterized as interior and exterior boundaries. A Polygon must have at most one exterior boundary and zero or more internal boundaries.
- ***MultiPoint:*** is defined by one or more Points, referenced through pointMember elements.
- ***MultiLineString:*** is defined by one or more LineStrings, referenced through *lineStringMember* elements.
- ***MultiPolygon:*** is defined by one or more Polygons, referenced through polygonMember elements.
- ***MultiGeometry:*** is a geometry collection that includes one or more geometries, referenced through geometryMember elements.

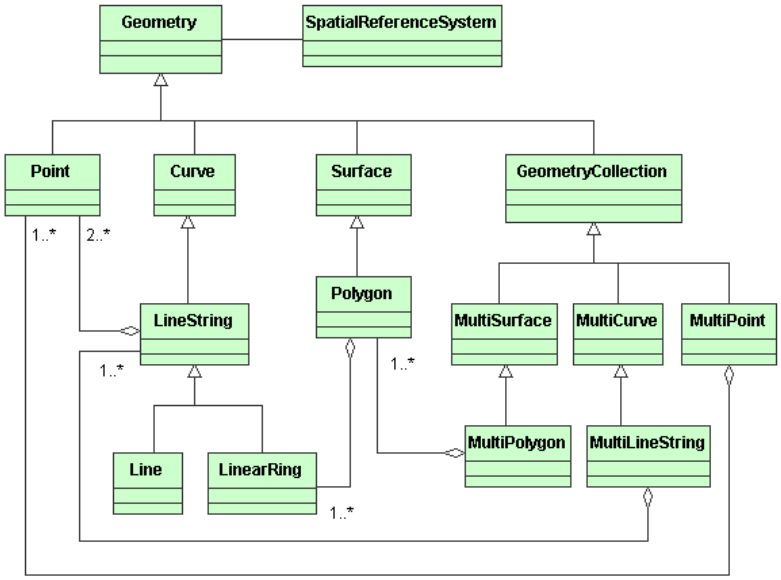

Fig. 2: GML Geometry class hierarchy

1.1.2 KML

Keyhole Markup Language (KML) is an XML-based language schema for expressing geographic annotation and visualization in map applications and 3D "geobrowsers" (Wilson, 2008). It was first developed by Keyhole Inc., which was acquired by Google in 2004. In 2007, Google submitted KML to the Open Geospatial Consortium (OGC). KML was adopted as an OpenGIS standard in 2008 (Wilson, 2008), and the OGC has now the responsibility for maintaining and extending the standard.

KML is focused on visualization of geographic features on map or a globe. The XML language also includes controls of the user's navigation in the sense of where to go and where to look (Wilson, 2008).

The relationship within GML and KML is the same holding within XML and HTML: GML is used to model and exchange geographic data, while KML is used to visualize them on a geobrowser.

KML specifies features (e.g. images, geometries, text etc.), their location in three dimensions, and optionally a preferred location from where to look at them. KML shares common geometry representations and features with GML. The KML schema elements are showed in Fig. 3.

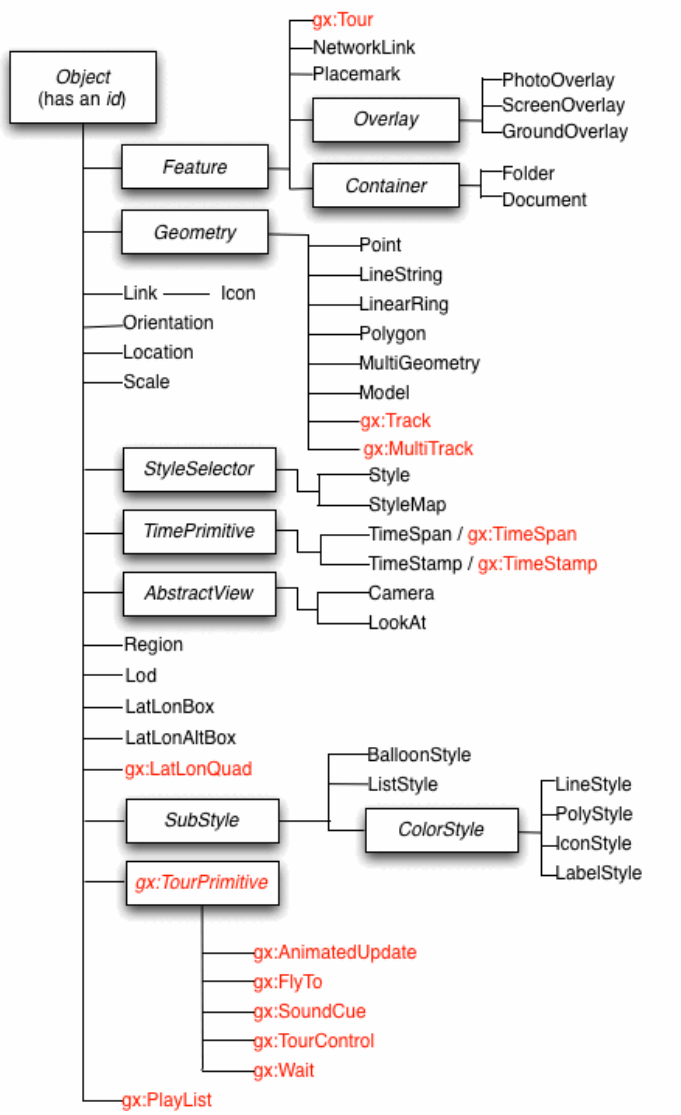

Fig. 3: KML Components

A KML file does not specify coordinate reference system (CRS). It is assumed that longitude and latitude coordinates are defined in WGS84 and altitude in meters above sea level measured from WGS84 EGM96 Geoid Vertical Datum.

KML also defines style rules (element style) for the element described in order to allow the customization of the element showed on the geobrowser.

KML documents and their related images and 3-D objects (if any) may be compressed using ZIP -encoding into KMZ files (Wilson, 2008). This greatly reduces the file size and makes data transfer more efficient, overcoming one of the major criticisms of XML-based structures.

1.2 OGC Web Services.

OGC Web Services (OWS) are defined using open non-proprietary Internet standards, these services are responsible for handling the different kind of operation on geospatial data.

1.2.1 Web Map Service (WMS)

The Web Map Service (WMS) protocol (OGC, 2006) is responsible of dynamically producing maps for georeferenced data from one or more distributed geospatial databases. A WMS request defines the geographic layers and area of interest to be processed, the response is one or more a digital image (JPEG, PNG, GIF) representing a map to be displayed on a web-client.

The WMS standard defines three operations: one returns service-level metadata; another return a map and an optional third operation returns information about particular features shown on a map. Those operations are implemented defining three HTTP requests, respectively: *getCapabilities*, *getMap* and *getFeatureInfo* requests. All these requests could be used in order to create a basic map where the user can identify a feature, get some basic information from it and perform some basic queries and are described as follows.

1.2.1.1 Get Capabilities

The *getCapabilities* is a mandatory WMS operation that returns the service metadata, which is a machine readable (and human-readable) description of the server's information content and acceptable request parameters values in XML format. See Table 1 on how the *getCapabilities* requests are formed.

Table 1: getCapabilities Parameters

Request parameter	Mandatory /optional	Description
VERSION=version	O	Request version
SERVICE=WMS	M	Service type
REQUEST=GetCapabilities	M	Request name
FORMAT=MIME_type	O	Output format of service metadata
UPDATESEQUENCE=string	O	Sequence number or string for cache control

1.2.1.2 Get Map

The *getMap* is a mandatory WMS operation that returns a map as a georeferenced image over the specified area. This is also sent as a HTTP request, see Table 2 on how the *getMap* request are formed.

Table 2: GetMap Request Parameters

Request parameter	Mandatory /optional	Description
VERSION=1.3.0	M	Request version
SERVICE=WFS	M	Service type
REQUEST=GetMap	M	Request name
LAYERS=layer_list	M	Comma-separated list of one or more map layers
STYLES=style_list	M	Comma-separated list of one rendering style per requested layer.
CRS=namespace:identifier	M	Coordinate reference system
BBOX=minx,miny,maxx,maxy	M	Bounding box corners (lower left, upper right) in CRS units
WIDTH=output_width	M	Width in pixels of map picture.
HEIGHT=output_height	M	Height in pixels of map picture.
FORMAT=output_format	M	Output format of map
TRANSPARENT=TRUE|FALSE	O	Background transparency of map (default=FALSE).
BGCOLOR=color_value	O	Hexadecimal red-green-blue colour value for the background color (default=0xFFFFFF).
EXCEPTIONS=exception_format	O	The format in which exceptions are to be reported by the WMS (default=XML).
TIME=time	O	Time value of layer desired.
ELEVATION=elevation	O	Elevation of layer desired.
Other sample dimension(s)	O	Value of other dimensions as appropriate.

The response to a valid *GetMap* request shall be a map of the spatially referenced information layer requested, in the desired style, and having the specified coordinate reference system, bounding box, size, format and transparency.

1.2.1.3 Get Feature Info

The *getFeatureInfo* is an optional request that will allow the user to retrieve information about an object. This request returns the attribute values from a certain location in the image retrieved from the *getMap* request. See Table 3 on how the *getFeatureInfo* request is formed.

Table 3: GetFeatureInfo request Parameters

Request parameter	Mandatory /optional	Description
VERSION=1.3.0	M	Request version
REQUEST=GetFeatureInfo		Request name.
map request part	M	Partial copy of the Map request parameters that generated the map for which information is desired.
QUERY_LAYERS=layer_list	M	Comma-separated list of one or more layers to be queried.
INFO_FORMAT=output_format	M	Return format of feature information (MIME type).
FEATURE_COUNT=number	O	Number of features about which to return information (default=1).
I=pixel_column	M	*i* coordinate in pixels of feature in Map CS
J=pixel_row	M	*j* coordinate in pixels of feature in Map CS
EXCEPTIONS=exception_format	O	The format in which exceptions are to be reported by the WMS (default= XML).

The nature of the *getFeatureInfo* response is at the discretion of the service provider, but it shall refer to the feature(s) nearest to the location selected.

1.2.2 Web Feature Service (WFS)

When the user needs access to the actual data represented on a map instead of an image of it, e.g. in order to modify, create or delete feature, a WFS request has to be performed.

The WFS protocol (OGC, 2010) allows the request and the update of spatial data from a web client. A XML based grammar, named GML (Geographic Markup Language) is used to encode data, but other common GIS format (e.g. SVG,

Shapefile) are also supported.

A request sent as WFS returns the data over a specified geographic area. This differs from the WMS request (see section 1.2.1) that only returns an image over a geographic area. The retrieved data from the WFS request is a file in the eXtensible Markup Language (XML) format. To reduce the amount of information within the retrieved files a filter can be applied to the WFS request to reduce the amount of data within the requested area (see section 1.2.2.7). This filter is transmitted in XML format and it supports the CQL standard, which also is a standard from Open geospatial Consortium.

In order to retrieve and handle the data over a network WFS supports five operations: *getCapabilities*, *DescribeFeatureType*, *GetFeature*, *GetGmlObject*, *Transaction* and *LockFeature*. Depending on the supported operations three class of Web Feature Service can be defined:

1. ***Basic WFS***: it would implement the *GetCapabilities*, *DescribeFeatureType* and *GetFeature* operations.
2. ***XLink WFS***: it would support all the operations of a basic web feature service and in addition, it would implement the *GetGmlObject* operation.
3. ***Transactional WFS***: it would support all the operations of a Basic WFS and in addition, it would implement the *Transaction* operation. Optionally, a transaction WFS could implement the *GetGmlObject* and/or *LockFeature* operations.

In the following subsection, the five WFS operators and the applicable filters will be described.

1.2.2.1 Get Capabilities

The *getCapabilities* operation returns an XML file that describes the data set. It provides all the information about e.g. the feature types, coordinate systems and name of the layers that can be accessed by a WFS request. It also provides all the operations that are supported by a WFS request.

1.2.2.2 Describe Feature Type

The *DescribeFeatureType* operation generates a schema description of feature types available in a WFS implementation. That differs from the *getCapabilities* operation that contains much more information etc. supported operations on the data set. The schema descriptions define how a WFS implementation expects feature instances to be encoded on input (via *Insert* and *Update* requests) and how feature instances will be generated on output (in response to *GetFeature* and *GetGmlObject* requests). The schema of the WFS request is showed in Fig. 4.

The *DescribeFeatureType* operation returns as response an XML schema document that is a valid GML application schema with the information to the user.

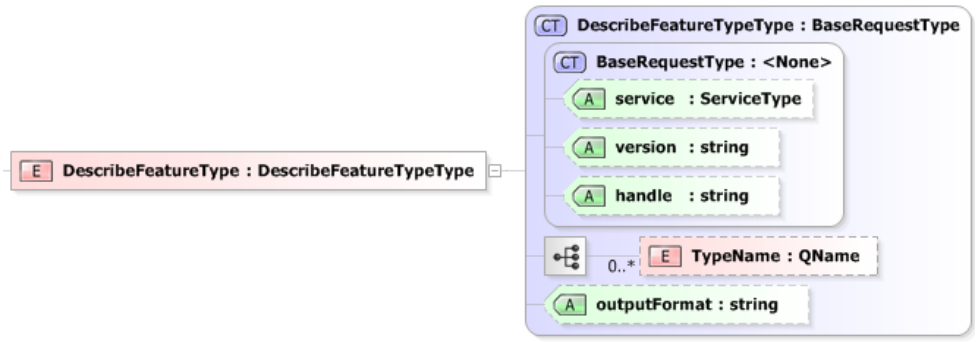

Fig. 4: DescribeFeature request XSD Schema

1.2.2.3 Get Feature

The *GetFeature* operation will send a query to the original data set and then return all data that fulfill these requirements set by the query. The returned data will contain features and it will be distributed in GML format. The schema of the *GetFeature* request is showed in Fig. 5.

The *<Query>* element (Fig. 6) defines which feature type to query, what properties to retrieve and what constraints (spatial and non-spatial) to apply to the feature properties in order to select the valid feature set.

The mandatory *typeName* attribute is used to indicate the name of one or more feature type instances or class instances to be queried.

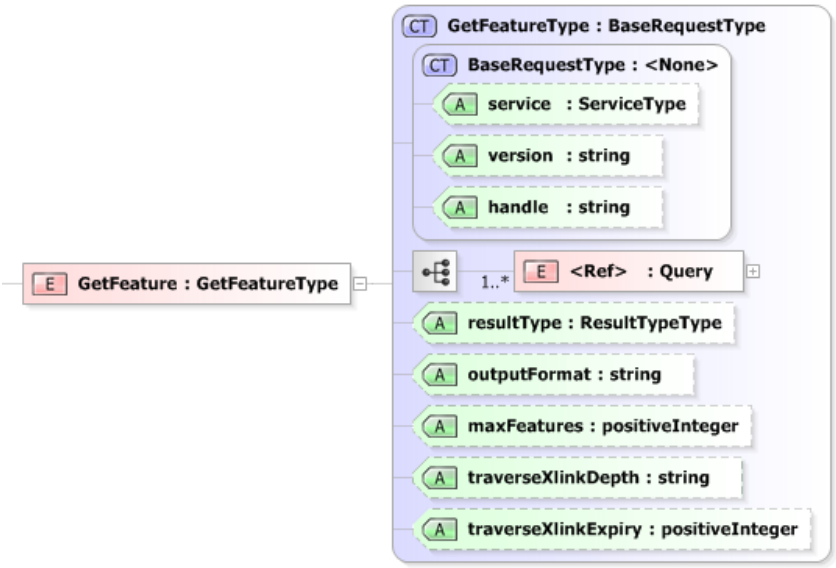

Fig. 5: GetFeature request XSD Schema

The *<Filter>* element can be used to define constraints on a query. It allows to describe both spatial and/or non-spatial constraints defined in (Vretanos, 2010) and as described in section 1.2.2.7.

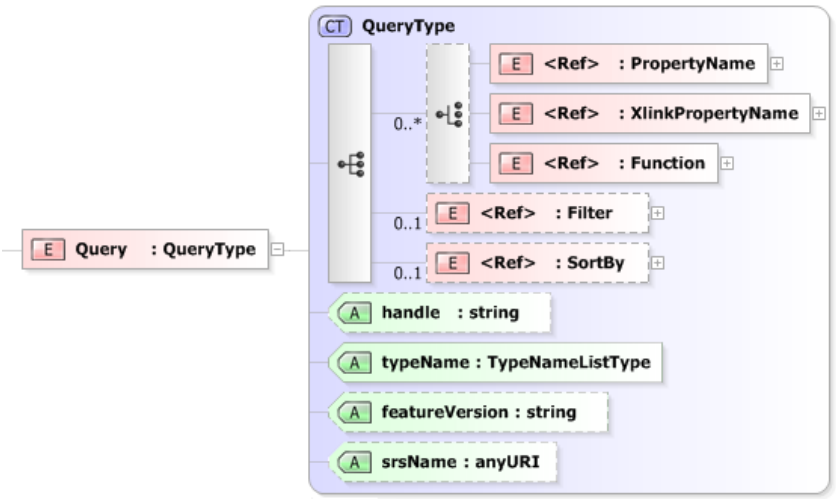

Fig. 6: WFS Query Element

The response to a *GetFeature* request must be valid according to the structure described by the XML Schema description of the feature type. Thus the WFS must report all the mandatory properties of each feature, as well any properties requested through the *<PropertyName>* element. The schema of the *GefFeature* response uses GML and is showed in Fig. 7.

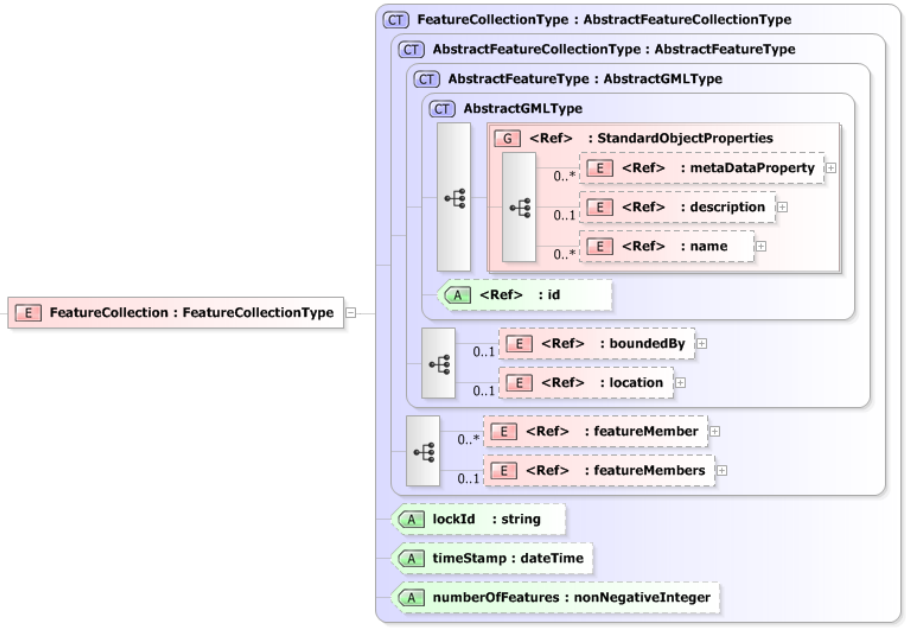

Fig. 7: GetFeature Response xsd schema

1.2.2.4 Get GML Object

The *GetGmlObject* operation allows the user to retrieve element instances depending on their ID. The schema of a *GetGMLObject* request is showed in Fig. 8.

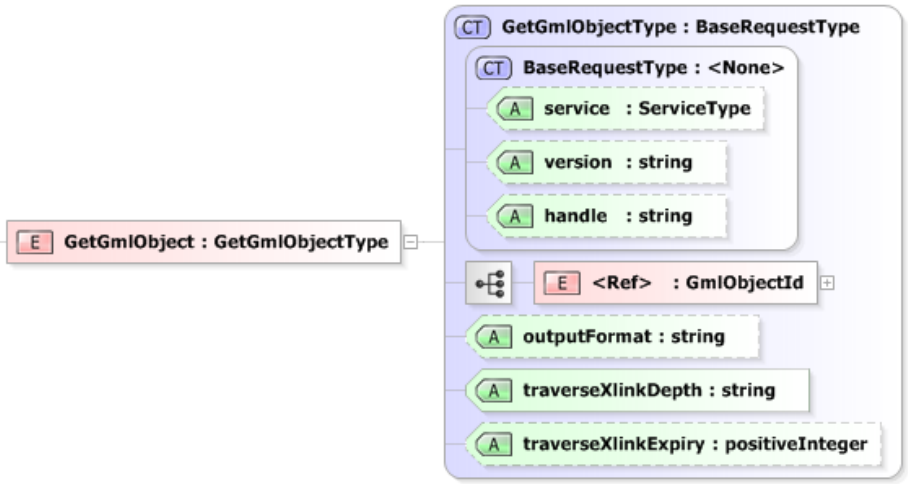

Fig. 8: GetGMLObject Request XSD Schema

The response to a *GetGmlObject* request is the referenced GML element returned as an XML document fragment. This differs from the response to a *GetFeature* request, which returns a complete document containing a *wfs:FeatureCollection*.

1.2.2.5 Lock Feature

The *LockFeature* is an optional operation that locks one or more features in order to ensure consistency. The set of feature to lock can be selected using a filter element. A feature locked with this operation can be modified by the operation allowed by the *Transaction* request (see next section). The *LockFeature* request schema is showed in Fig. 9.

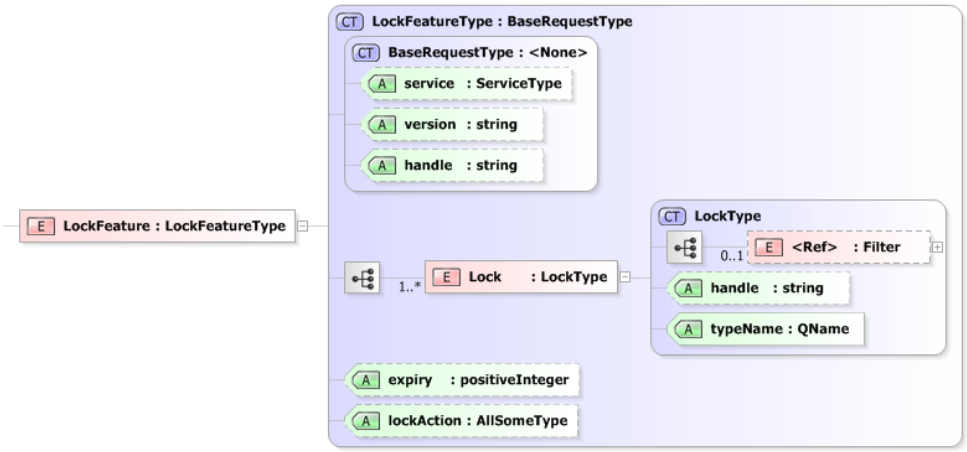

Fig. 9: LockFeature request xsd Schema

The response to a *LockFeature* request is an XML document that will contain a lock identifier that a client application can use in subsequent WFS operations to operate upon the set of locked feature instances.

1.2.2.6 Transaction

The *Transaction* operation supports the creation, deleting and updating operations on geographic data. These operations allow the user to remotely modify a geographical data set. The *create* operation allows the user to add information to the retrieved data. The *delete* operation allows the user to remove information from the retrieved data. The *update* operation transmits the modifications done from the create and delete operation to the source and saves the changes to the original data set. The *Transaction* request schema is showed in Fig. 10.

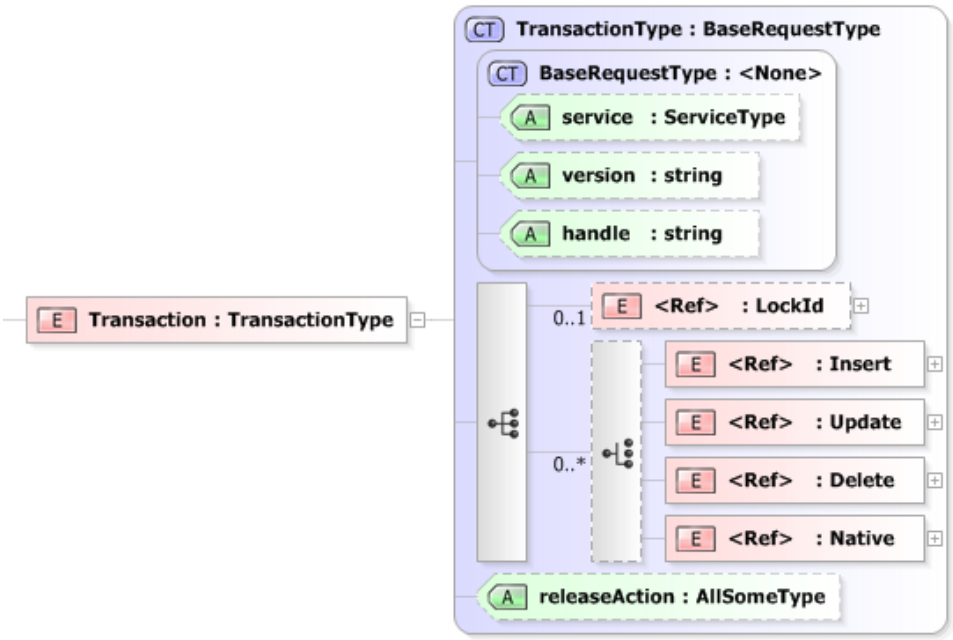

Fig. 10: Transaction Request XSD Schema

The response to a *Transaction* request is an XML document (see Fig. 11) indicating the termination status of the transaction.

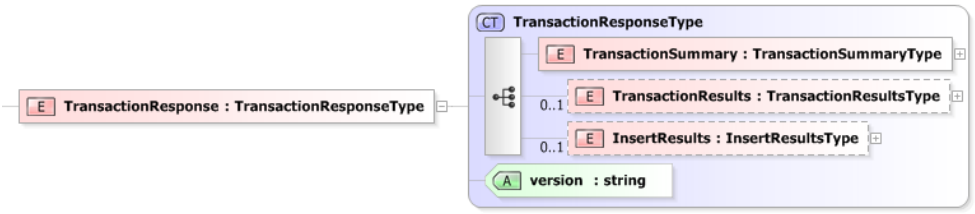

Fig. 11. Transaction Response XSD Schema

1.2.2.7 Filter

A filter is used to identify a subset of resources from a collection whose property values satisfy a set of logically connected predicates. If the property values of a resource satisfy all the predicates in a filter then that resource is considered to be part of the resulting subset (Vretanos, 2010). Fig. 12 shows the Filter xml schema.

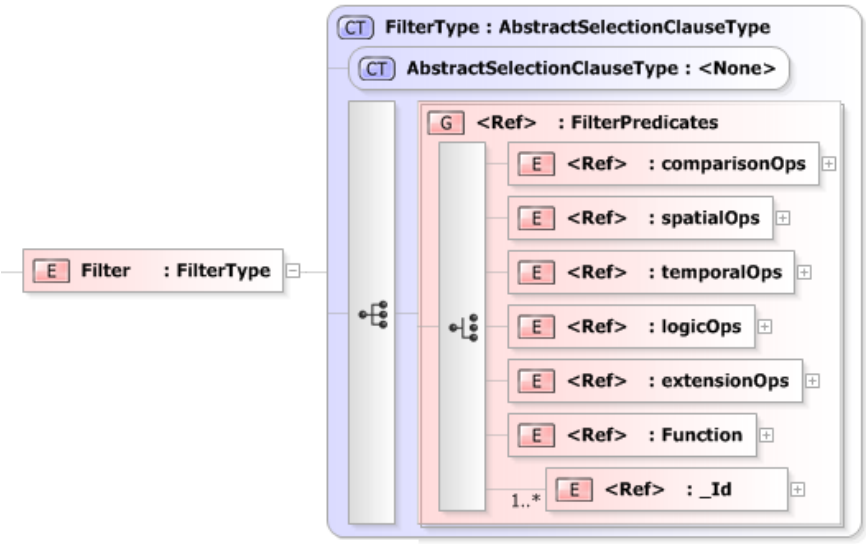

Fig. 12: Filter Operator XSD Schema

The following types of filters are defined:

1. ***Comparison operators***: are used to form expressions that evaluate the mathematical comparison between two arguments. If the arguments satisfy the comparison then the expression evaluates to true. Otherwise the expression evaluates to false. As showed in figure the OGC define the following comparison operators: *PropertyIsLike*, *PropertyIsNull*, *PropertyIsNil*, *PropertyIsBeetwen*.

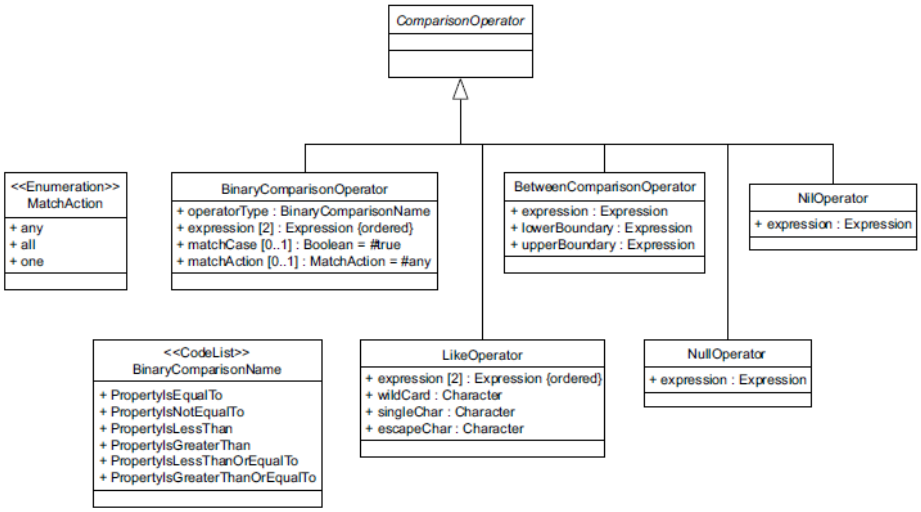

Fig. 13: Filter Comparison Operators (OGC, 2010)

2. ***Spatial Operators***: A spatial operator (see Fig. 14) shall determine whether its geometric arguments satisfy the stated spatial relationship. The operator shall evaluate to true if the spatial relationship is satisfied. Otherwise, the operator shall evaluate to false. The meaning of the defined spatial relationship is exploited in Fig. 15.

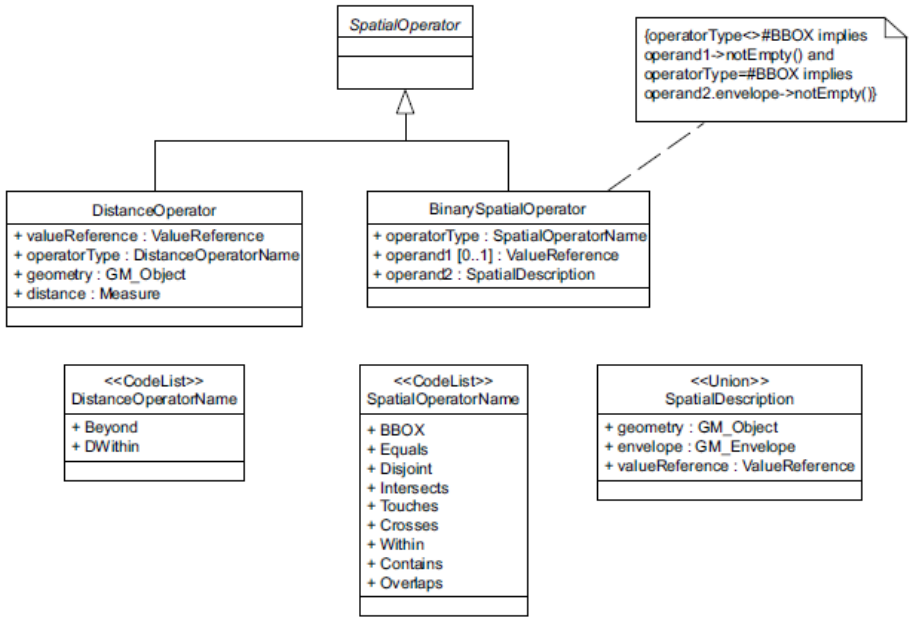

Fig. 14: Filter Spatial Operator (OGC, 2010)

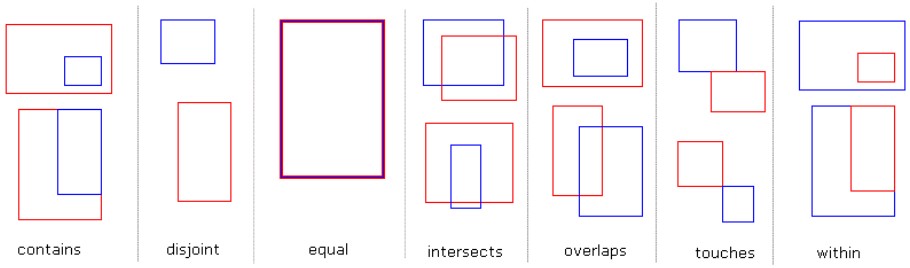

Fig. 15: Filter Spatial Relationship

3. Temporal Operators: A temporal operator determines whether its time arguments satisfy the stated temporal relationship. The operator evaluates to true if the temporal relationship is satisfied. Otherwise, the operator evaluates to false.

4. ***Logical Operators***: A logical operator (i.e. AND, OR, NOT) can be used to combine one or more conditional expressions. The logical operator AND evaluates to true if all the combined expressions evaluate to true. The operator OR operator evaluates to true is any of the combined expressions evaluate to true. The NOT operator reverses the logical value of an expression.

1.2.3 Web Coverage Service (WCS)

The Web Coverage Service (WCS) (OGC, 2010) protocol provides geospatial data as coverage in digital information. The data served by a WCS are grid data, i.e. satellite images, usually encoded in a binary image format, so they cannot be easily displayed by a web client.

1.3 OGC OWS Architecture

OGC suggests a four loosely coupled tiers architecture (OGC, 2005). This OWS architecture is designed for application in which data are voluminous, but can be adapted to any kind of application bypassing un-needed tiers, as indicated by some arrows in Fig. 16. The communication intra and extra tier is done only through open non-proprietary internet standards like HTTPPOST, HTTPGET and SOAP. Follows a brief description of each tier:

- *Clients*: this tier is responsible to handle the interaction with users and to display the request information possibly on a map (i.e. on a Geobrowser).
- *Application Services Tier*. This component contains services designed to support thin client such as web browsers. Its design has the goal of relieving each client directly performing often-needed support functions.
- *Processing Services Tier*. This tier contains services designed to process both feature and image (coverage) data to render on the Geobrowser in the client.
- *Information Management Services Tier*. This tier contains services designed to store and provide access to data and metadata. Is used by invoking web services from the others tiers like WFS, WMS, WCS, etc.

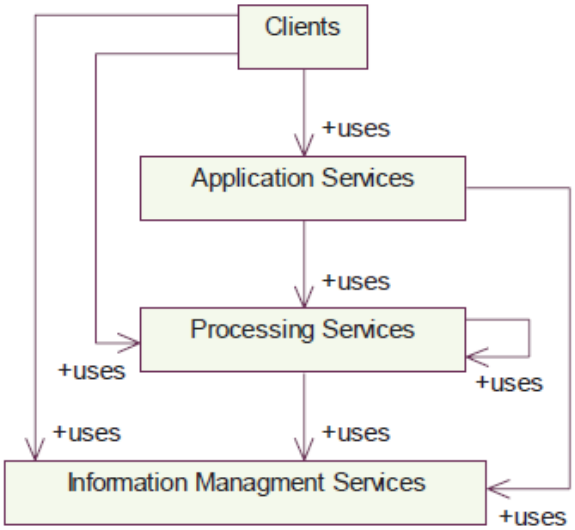

Fig. 16: OWS Web Architecture (OGC, 2005)

1.4 Web GIS Tools

Many tools have been designed with the purpose of offering web GIS services. They can be classified into four class, but often implemented tools belongs to two classes. From the client to the data storage we can distinguish:

- *GIS Client*: these tools are responsible to render information on a navigable 2D or 3D map to the users, and to allow them to interact with displayed data. Tools that fall in this category are Google Maps (Google, n.d.), OpenLayers (OpenLayers, 2011) etc.
- *Map Services*: these tools are able to render raster and vector maps images in a GeoBrowser, implementing the WMS protocol (i.e GeoServer (OSGeo, 2011), MapServer (OSGeo, 2008).
- *GIS Web Server*: these tools offer services to retrieve spatial data, implementing the WFS and WCS protocol (again GeoServer, MapServer)
- *Spatial Data Storage*: are responsible to store and make persistent spatial data, offering spatial types inspired to the OCG Geometry (i.e. Oracle Spatial (Oracle, 2011), Postgis (PostGis, 2011)).

Chapter 2 RDF and the Semantic web

Nowadays the diffusion of information through the World Wide Web (WWW) is more and more increasing. The way data were exchanged between applications and diffused to user changed with the diffusion and availability of internet connection. We passed from the static and hand-filled web pages of the early stage of www, to the data stored in relational or semi-structured (XML) databases and dynamically visualized to users on request. This step that separates the data, stored in an abstract data structure, and their visualization, by dynamically generating web page on request, augmented significantly the amount of information available on the WWW. However, the vast majority of information displayed on the web is built only for human user visual consumption. The web will continue to grow; more people will participate in it, and more every-day procedures will be performed as web applications. This growth of data available needs to be processed by machines in order to retrieve relevant information. This is the goal of the new evolution of WWW: the *Semantic Web*. The *Semantic Web* is an extension of the current web in which information is given well-defined meaning, better enabling computers and humans to work in cooperation (Berners-Lee et al., 2001).

This means that resources as well as relations between resources are characterized in a formal way.

From the Semantic Web point of view, who provides data on the web has to migrate from the from human-only presentation of content to forms accessible also by machine agents by providing meta-data, thus enabling semantic-aware applications to be built on top. There is need for an accepted standard to express those metadata, the same way HTML and XML are accepted standard for document visualization and structure.

The Resource Description Framework (RDF) provides the infrastructure for the expression, the exchange and the extension of metadata. The RDF model specifies the expression of assertions as an oriented and labeled graph.

RDF provides a way to state assertion but their meaning depends on the

understanding of the concepts the RDF statements are made up of. There is a need of a machine-understandable mechanism for the definition of vocabularies which helps to associate a meaning to an RDF statement. RDF Schema (RDFS) consents adding structure and meaning to RDF statement, allowing an elementary structuring of a vocabulary understandable also by machine agents. RDFS has a limited expressivity, but the support tor more complex vocabulary definition such as ontology, is provided by the Web Ontology Language OWL.

In the next of the chapter, details on the RDF, RDFS, and OWL structure and their query language SPARQL will be provided.

2.1 Resource Description Framework (RDF)

The Resource description Framework (RDF) is a W3C recommendation for representing information in the Web (W3C, 2004).

A Uniform Resource Identifiers (URIs) identifies *resources* in RDF. URIs provide globally-unique and resolvable identifiers for entities on the Web. Everything is identifiable by an URI can be described in RDF (e.g. persons, animals, things etc.).

The basic elements of RDF are *statements*, which are *triples* `<subject, predicate, object>` consisting of the resource (the *subject*) being described, a property (the *predicate*), and a property value (the *object*). In a statement, the *subject* and *predicate* must be resource and *object* could be a resource or a literal. A literal is a string of a certain datatype and may only occur as the object of a statement. In some cases, there is the need of describing resources using more complex structures of data than using a literal string or an URI pointer. Anonymous resource, also called blank node, are used for this purpose. A blank node identifier represents such a resource. Follow the formal definition of an RDF-statement (Definition 2.1.1).

Definition 2.1.1 RDF Statement (Triple)

Let be $\mathcal{U}$ the set of resource identified by an URI, $\mathcal{B}$ the set of blank nodes identifiers and $\mathcal{L}$ the set of possible literal values of whatever datatype. Then

$\mathcal{T}$: (s, p,o) ϵ ($\mathcal{U} \cup \mathcal{B}$) $\times$ ($\mathcal{U}$) $\times$ ($\mathcal{U} \cup \mathcal{B} \cup \mathcal{L}$) is an RDF statement (triple).

RDF triples can be visualized as a directed labeled graph, in which *subjects* and *objects* are represented as nodes, and *predicates* as arcs (See Fig. 17).

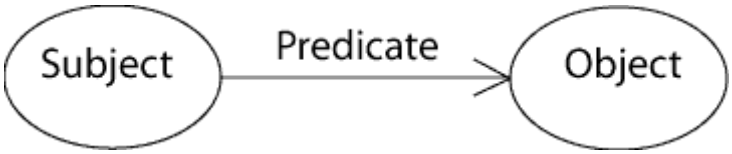

Fig. 17: RDF statment graph representation

Definition 2.1.2 RDF Graph

A set of RDF statements is an RDF Graph:

$$\mathcal{G} = \{\mathcal{T} \mid \mathcal{T} \text{ is an RDF Triple}\}$$

RDF provides formalism, called *reification*, which aims to make statements about other statements. This formalism is useful in order to record information about when statements were made, who made them, or other similar information (this is sometimes referred to as "provenance" information). This is useful for example for trust and authoring issues, knowing who stated a concept, represented by a triple and reified, help to decide whether to trust or not the information contained in the triple.

In RDF reification a blank node symbolizes the statement to be described, while four other statements (*rdf:Statement, rdf:subject, rdf:predicate, rdf:object*) are used to provide the link between the blank node and the statement to be described.

The set of all URI resources and literals in an RDF graph is called vocabulary. In a broader sense, a vocabulary is a set of concepts with a well-understood meaning to make assertions in a certain domain, for example ontology.

There are many way to represent RDF graph. We have seen above a graphical representation that is very readable but is difficult to serialize and parse for a machine. Another format used to represent RDF graph are the N3 format and its derivate Turtle and N-Triple, they are textual line based representation where each line represent a triple and resource are encoded between angular parentheses.

The most prominent serialization format is RDF/XML, the advantages to use

an XML representation are manifold, XML is the standard for the exchanging of data and provides languages to query (XQuery) and transform (XSL) data to represent them.

2.2 RDF Schema (RDFS)

RDF Schema (RDFS) (Brickley & Guha, 2004) provides a standard vocabulary for describing the classes and relationships used in RDF graphs.

Classes represent logical groups of resources, and a member of a class is said to be an instance of the class. The *rdf:type* property is used to define class and property types (e.g., the triple *<C, rdf:type, rdfs:Class>* asserts that *C* is a class). *rdf:type* is also used to denote instances of classes (e.g., *<i, rdf:type, C>* asserts that *i* is an instance of *C*). It is possible to define a class hierarchy using the predicate *rdfs:subClassOf*.

The RDFS vocabulary offers also a way to model properties defying their range and domain using the elements *rdf:Property*, *rdfs:domain*, *rdfs:range*. For example the set of triple in Table 4 asserts that *p* is a property of the class *C* (domain) and has strings as range.

Table 4: RDFS property declaration example

P	rdf:type	rdf:Property
P	rdfs:domain	C
P	rdfs:range	Xsd:#string

It is also possible to define a hierarchy of properties using the predicate *rdsf:subPropertyOf*.

Every RDFS statement can be seen as a Frist Order Logic formula as depicted in Table 5.

Table 5: Frist order Logic formula for RDFS Triples

RDF Triple	FOL formula
<C rdf:type rdfs:Class>	C(i)
<i rdfs:type C>	
<P rdf:type rdfs:Property>	P(a,b)
<a P b>	

<C rdfs:subClassOf D>	∀X (C(X)⇒D(X))
<P rdfs:subPropertyOf R>	∀X ∀Y (P(X,Y)⇒R(X,Y))
P rdfs:domainC	∀X ∀Y (P(X,Y))⇒C(X))
P rdfs:rangeD	∀X ∀Y (P(X,Y)⇒ D(X))

RDFS provides the capability to define ontologies. Ontologies serve to formally specify the semantic of RDF data so that a common interpretation of the data can be shared across multiple applications.

2.3 SPARQL

SPARQL Protocol And RDF Query Language (SPARQL) was defined by the W3C Data Access Working Group in 2004. It defines a query language for RDF Graphs.

The most prominent concept in SPARQL query language is the triple pattern. A triple pattern is a triple <*subject, predicate, object*> where each of the three elements can be a variable and both subject and predicate can be a literal. A collection of triple pattern is called graph pattern.

The result of a SPARQL query on the RDF Graph G is the Sub-Graph of G matching the graph pattern(s) given in input. There are several types of graph patterns:

- *Group graph pattern*: the set of triple pattern are considered in logical AND, they are represented between brackets (“{}”) and concatenated by point (“.”). The result will satisfy all the triples in the group pattern.
- *Union graph pattern*: the set of triple patterns are considered as alternative, they are concatenated by the reserved word UNION. The result will satisfy at least one of the triples in the union pattern.
- *Optional graph pattern*: the set of triple patterns are evaluated optionally, they are concatenated by the reserved word OPTIONAL. If the optional pattern is not satisfied the execution does not stop, but the rest of the triple pattern will be shown.
- *Filter graph pattern*: this pattern is used in order to impose logical, mathematical and other constraints on the triple retrieved. The result will satisfy the filter constraints.

SPARQL provides several query types. The most common is the SELECT query that returns all, or a subset of, the variables bound in a query pattern match. Its syntax is similar to the SQL select and is described in Definition 2.3.3.

Definition 2.3.3 SPARQL SELECT syntax

SELECT $\mathcal{V}$ FROM $\mathcal{U}$ WHERE $\mathcal{P}$

where $\mathcal{U}$ is the URL of an RDF graph $\mathcal{G}$, $\mathcal{P}$ is a SPARQL graph pattern and $\mathcal{V}$ is a tuple of variables appearing in $\mathcal{P}$.

The CONSTRUCT SPARQL query returns an RDF graph constructed by substituting variables in a set of triple templates. Its syntax is described in Definition 2.3.4.

Definition 2.3.4 SPARQL CONSTRUCT syntax

CONSTRUCT $\mathcal{V}$ WHERE $\mathcal{P}$

where $\mathcal{P}$ is a SPARQL graph pattern and $\mathcal{V}$ is a tuple of variables appearing in $\mathcal{P}$.

The ASK query returns a boolean indicating whether a query pattern matches or not. Its syntax is described in Definition 2.3.5.

Definition 2.3.5 SPARQL ASK syntax

ASK $\mathcal{P}$

where P is a SPARQL graph pattern.

The DESCIBE query returns an RDF graph that describes the resources found. Its syntax is described in Definition 2.3.6.

Definition 2.3.6 SPARQL Describe syntax

DESCRIBE $\mathcal{V}$ WHERE $\mathcal{P}$

where $\mathcal{P}$ is a SPARQL graph pattern and $\mathcal{V}$ is a tuple of variables appearing in $\mathcal{P}$.

SPARQL uses post-filtering clauses which allow, for example, to order (ORDER BY clause), or to limit (LIMIT and/or OFFSET clauses) the answers of a query.

2.4 OWL

The OWL Web Ontology Language is designed for use by applications that need to process the content of information instead of just presenting information to humans. OWL facilitates greater machine interpretability of Web content by providing additional vocabulary along with a formal semantics (McGuinness & Van Harmelen, 2004).

OWL enrich RDF(S) model by providing vocabulary terms to express some concepts and relationships. For example OWL introduce two kind of properties (sub-property of *rdf:Property*):

- *owl:DatatypeProperties*, that define properties having as range RDF literals and XML Schema datatypes;
- *owl:ObjectProperties*, that define properties having as range other class instances.

OWL allow also to introduce restriction on the cardinality of a defined property, by using the elements *owl:minCardinality* and *owl:maxCardinality*.

More complex elements of owl allow to define transitivity, symmetric, functional and inverse property (respectively *owl:transitiveProperty*, *owl:symmetricProperty*, *owl:functionalProperty*, *owl:inverseProperty*).

Depending on what OWL elements are used and their instantiation, it is possible to classify OWL in three increasingly expressive sublanguages:

- *OWL Lite:* it supports property and class hierarchies and simple constraints. For example, while it supports cardinality constraints, it only permits cardinality values of 0 or 1.
- *OWL DL:* its name is due to its correspondence with the Description Logics. OWL DL includes all OWL language constructs, but they can be used only under certain restrictions (for example, while a class may be a subclass of many classes, a class cannot be an instance of another class). It guarantees the maximum expressiveness while retaining computational completeness and decidability.
- *OWL Full* is meant for users who want maximum expressiveness and the

syntactic freedom of RDF with no computational guarantees. It supports all owl vocabulary elements without restrictions.

OWL Full can be viewed as an extension of RDF, while OWL Lite and OWL DL can be viewed as extensions of a restricted view of RDF. Every OWL (Lite, DL, Full) document is an RDF document, and every RDF document is an OWL Full document, but only some RDF documents will be a legal OWL Lite or OWL DL document (McGuinness & Van Harmelen, 2004).

2.5 RDF(S)/OWL Modeling Vs. Standard Modeling

The prominent contribution of the presented thesis is the definition of a meta-model for representing data having Spatio-Temporal feature. RDF, RDFs and OWL seem to be the perfect candidate to express and model meta-data and their instances.

RDF modeling is easy to extend and to dynamically adapt to user needs, this feature would be difficult to achieve in the relational database model were the data have a predefined structure with simple record-type, and the schema is fixed and difficult to extend.

The graph structure of RDF statements allows modeling both meta-data that application data using the same formalism using the same query language (SPARQL) to retrieve information. Also the abstract graph structure of RDF allows a concept modeling and querying that is unambiguous and independent from the representation, which is not true for the tree-based XML structure. Let us make an example, suppose that user wants to model the concept "*Tolkien written the Lord of the Rings*". There are several XML representations of this concept, showed in Table 6(a, b, c). There are also many formats to represent an RDF triple, as showed in showed in Table 6d (N-Triple) e Table 6 (RDF/XML).

Table 6: XML and RDF Representations

	XML Reprepsentation
a	<Book> <Author> J.R.R. Tolkien </Author> <Title> The Lord of the Rings </Title> </Book>

b	<Author> <Name> J.R.R. Tolkien </Name> <Books> <Book> <title> The Lord of the Rings </title> </Book> </Author>
c	<BOOK title="The Lord of the Rings"> <AUTHOR> J.R.R. Tolkien </AUTHOR> </BOOK>
	RDF Triple
d	<authors:Tolkien> <hasWritten> "The Lord of the Rings"
e	<rdf:Description abuot="#authors:Tolkien"> <hasWritten> The Lord of the Rings </hasWritten> </rdf:Description>

Now, let be the case that another user wants to know who wrote the book "The lord of the Rings". To retrieve this information from an XML datasource, one has to know the representation schema and write the appropriate XQuery/XPath (see Table 7(a, b, c)). On the contrary the SPARQL query (see Table 7d) remains independent from the RDF encoding because both the model and the language are based on the abstract graph data structure.

Table 7: XQuery and Sparql

	XQUERY
a	/Book[/Title/Text()="The Lord Of Rings"]/Author
b	/Author[/Book/Book/title/text()="The Lord of Rings"]/Name
c	/BOOK[@title="The Lord Of Rings"]/Author
	SPARQL
d	Select ?x WHERE ?x <hasWritten> "The Lord of the Rings"

Chapter 3 Related Work

In this chapter, significant related work on spatio-temporal data modeling and spatio-temporal visualization will be illustrated.

3.1 Data Model

The capabilities of any information system largely rely on the design of its data model. A rigorous data model must be able to manage and foresee any spatio-temporal queries and analytical methods to be performed (Yuan, 1996).

Many approaches on modeling spatio-temporal data adopt the triad model (Peuquet, 1994), i.e. data are represented with reference to the *where* (spatial reference), *what* (object attributes) and *when* (time reference). So a spatio-temporal phenomena could be seen under three dimension, respectively: spatial, descriptive and temporal. What changes the efficiency and the expressiveness of a model is the way spatial and temporal models are defined and integrated.

One of the first attempt to manage spatio-temporal data was made in (Worboys, 1994), with a model able to handle both spatio-temporal phenomena like the changing of regional structure of administrative areas over time, and the information about network development and the land ownership changing over time. Other works extend Entity-Relationship (ER) model (Tryfona et al., 2003) or the Object Oriented (OO) model defining some Abstract Data Types (ADT) (Parent et al., 1999) aiming at describing entities, properties and relationships of spatio-temporal data. Authors focus on the orthogonally representation of space and time. They provide ADTs (Abstract Data Type) for space and time object, and define object type, relationship and constraint on this kind of object.

In (Pelekis et al., 2005) a comparative overview of some existing models in terms of temporal, spatial, spatio-temporal semantic and queries capabilities is provided.

Another OO model was proposed in (Camossi et al., 2003). This work extends the ODMG model including classes and literal types in order to model time and space and authors introduce spatial and temporal granularities. They define

spatial and temporal parametric types to represent the spatial, temporal and spatio-temporal dimension of a feature, and they define a set of spatial primitives (very similar to the OGC ones). Both space and time are partitioned in granules (i.e.: day, week, year for time; ms, dm, km for space). Granules are related by the "finer-than" relation (and its inverse "coarser-than" relation) that specifies if a granule is more specific that another one. Multigranularity could introduce inconsistencies while arranging or converting data from a granule to another more or less specific. In a more recent work (Bertino et al., 2009) authors discuss and propose model to avoid this problem. The (Camossi et al., 2003) approach is similar to the one proposed in this thesis, we define a set of property instead of parametric types depending on the fact that we use RDFS instead of extending ODGM. The proposed model manages standard spatial properties instead of self-defined ones, and also allows user to define own spatial and temporal granularity instead of to be forced in some pre-defined granularities.

Other works aim at modeling spatio-temporal data from a database point of view, proposing Spatio-Temporal Database Management System (STDBMS). In (Nadi & Delavar, 2005) authors present a model based on five database tables storing separately information about descriptive attributes, time, space and spatio-temporal event. The data retrieval is based on a three access level. This model attempt to represent the three dimensions of data independently but manages only object that change their spatial extension, and it is limited to the temporal operator and representation available on a database. In (Lohfink et al., 2010) an OO model for STDBMS based on time versioning is proposed. Here time is treated as an attribute of the spatial objects, and this limits the temporal representation and exploration.

In (Innerebner et al., 2007) authors present a web architecture based on OGC standards in order to offer a web service for spatio-temporal data. However this solution is based on a proprietary STDBMS, only moving-objects are managed and does not offer many facilities to exploit time, limiting the user in the temporal exploration. As showed in section 1.3 the Open Geospatial Consortium (OGC) suggests best practice for a Web-GIS architecture, defining four loosely coupled

tiers architecture (OGC, 2005). The here presented model completely fulfill this best practice.

3.1.1 Space, Time and Ontologies

As seen in previous section, many approaches on modeling spatio-temporal data adopt the triad model proposed in (Peuquet, 1994). The RDF/OWL data model proposed by W3C and used for the proposed CMS seems perfectly to model this threefold nature of data. In RDF a resource is represented by a triple `<Subject, Predicate, Object>`, each triple represents the instance of an association between the things denoted by the nodes that it links (W3C, 2004).

The current state of art is rich of proposals of spatial, temporal and spatio-temporal domain RDF models and ontologies.

The work of (Gutierrez et al., 2007) introduces the concept of temporal RDF graphs and defines their query language. They use the RDF mechanism of reification (see section 2.1) in order to add the temporal dimension to a resource. (Hobbs & Pan, 2006) also provided RDF/OWL ontology for describing the temporal content of Web pages and the temporal properties of Web services.

In (Spaccapietra et al., 2004) a review of the current RDF/OWL spatial ontologies is provided. In (Kolas et al., 2005) authors propose a five layer architecture of spatial ontologies types in order to build an interoperable geospatial semantic system:

1. *Base geospatial ontology*: the core geospatial knowledge vocabulary and knowledge structure.
2. *Feature data source ontology*: an ontological view of WFS data.
3. *Geospatial Service ontology*: knowledgebase discovery and execution of all registered geospatial services.
4. *Geospatial filter ontology*: the integration of geospatial relationships into the queries.
1. *Domain ontology*: a knowledge representation that is organized, customized, and aligned with a specific domain and/or user.

In (Lieberman et al., 2006) the Open Geospatial Consortium (OGC) defines the

first standard spatial ontology for the geospatial semantic web. It is a porting of GML language and is the de facto standard in order to represent spatial information in RDF/OWL. OGC together with the W3C are defining an extension of SPARQL, namely GeoSPARQL (OGC, 2012), in order to query spatial data expressed in RDF. Actually GeoSPARQL is only a draft and there are not tools supporting this query language.

The work of (Perry et al., 2007) is one of the first attempts to model spatial, temporal and descriptive properties using RDF. Authors use the (Gutierrez et al., 2007) temporal RDF graph and define spatial and spatio-temporal upper-level ontologies. They also define and implement a set of operators, providing a framework that relies and extends the Oracle DBMS. The upper level ontology for the thematic dimension propose a binary classification between feature that changes or not over time (*Occurents* vs. *Continuants*), and consequentially if they changes a spatial property (*Spatial Occurent* vs. *Non Spatial Occurent*). The model presented in this thesis does not link the spatio-temporal event to the type of feature, considering a feature as a general object that can have spatial, temporal, descriptive and/or spatio-temporal properties. We believe that this approach best suits the triad model (Peuquet, 1994). Furthermore, the temporal domain proposed in this thesis allows a more complex temporal representation and querying.

In (Batsakis & Petrakis, 2010) authors propose SOWL, an extension of OWL, to represent qualitative and quantitative spatial information employing the RCC-8 topological relations, cardinal direction relations, and distance relations. They include this relationship in the model and use a SWRL rules implemented in a Pellet reasoner to infer spatial relations between entities. Another ontology based model for spatio-temporal data is presented in (Lyell et al., 2011). Authors introduce the concept of spatio-temporal coordinate, using *Gml:Point* for the spatial reference and a time-stamp for the temporal one. In order to develop Spatio-Temporal Ontologies using existing standards it is a good start point, but the choice of this primitives limits the kind of objects that can be represented and the spatial and temporal exploration.

3.1.2 Spatio-Temporal Classification

Several different forms of spatio-temporal data types and applications are available in real world and the current state of art offers many ways to classify them (Asproth et al., 1995), (Nadi & Mahmoud, 2003), (Kisilevich, 2005), (Pfoser & Tryfona, 1998).

As stated in (Nadi & Mahmoud, 2003) from temporal point of view, there are two types of objects:

- *Static*: objects that may not change in a short period of time, but that will change in the long period (Asproth et al., 1995) as cartographic maps, roads, etc.
- *Dynamic*: objects that may changes some of its feature (spatial or descriptive) in a short period of time.

Temporal dynamic objects can be still classified according to the dynamic aspects of spatial information (Nadi & Mahmoud, 2003):

- Objects that change their shape over time.
- Objects that change their position over time.
- Objects that change their descriptive features attribute over time.
- Any combination of the above changes.

In (Kisilevich, 2005) authors classify spatio-temporal data according to three dimensions, generating five classes of objects described in Fig. 18.

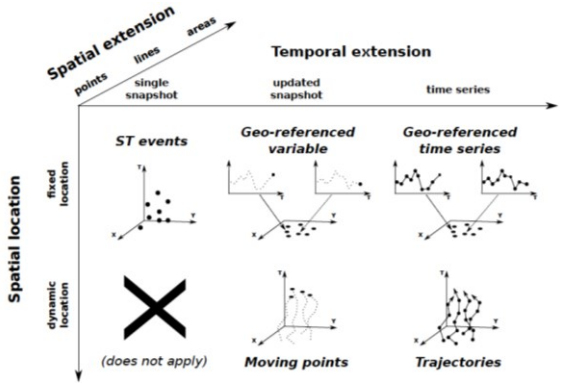

Fig. 18: Spatio-temporal classification used in (Kisilevich, 2005)

The temporal dimension considers three type of changes: an object that does not evolve at all (static snapshot), an object that changes status and record only its most recent value (update snapshot) or all the changes history is kept (time

series).

The spatial dimension describes whether the objects considered are associated to a static or a dynamic location. The spatial extension considers the spatial geometry of the object: point, line or shape.

Furthermore a classification of spatiotemporal application, based on the data they manage is given in (Pfoser & Tryfona, 1998):

- applications dealing with objects that change their position over time
- applications involving objects that change their shape over time
- applications dealing with objects which integrate the above two behaviors;

As stated in (Asproth et al., 1995), most of the phenomena in the real word are dynamic in nature, so classifying object on their static or dynamic nature could be useless. For example the changes of a road or of a map, seen static, could be view as an object that changes its shape referring to the dynamic spatial classification in (Nadi & Mahmoud, 2003). Furthermore the distinction among the temporal changes (snapshot, time series) and the spatial extension (point, line, shape) in (Kisilevich, 2005) could be useful for clustering purpose, but for data modeling this difference could be generalized.

3.2 Visualization

In this section will be described related work on spatio-temporal visualization and geovisualization.

3.2.1 Spatio-Temporal Visualization

Adequate spatio-temporal data visualization techniques improve human's mental analysis abilities allowing users to "see" knowledge inside data (Camossi et al., 2003).

In a spatio-temporal analysis, users interact with information in a different manner, discovering, analyzing and exploring information into a unique interoperable environment (e.g. (MacEachren & Kraak, 2001), (Andrienko et al., 2007)). To allow this kind of knowledge acquisition, user needs tools to support

data exploration to validate and/or reformulate hypothesis. These tools are fundamental for the analysis and exploration of data in the assisted process using and they should provide adequate spatio-temporal visualization techniques and a high level of interaction capabilities, a fundamental feature for the analysis and exploration of data (MacEachren et al., 2004).

In (Andrienko, 2003) authors provide a classification of existing visualization techniques for exploratory analysis of spatio-temporal data. They extend Pequet's classification scheme of data components users are interested in (Where, When, What) (Peuquet, 1994) using the notion of reading levels individually applied to spatial, temporal and spatio-temporal data (Koussoulakou & Kraak, 1992). This approach generates four categories of spatio-temporal queries. For each category, the authors suggest the best visualization technique, like Map iteration, Map animation, time windowing, space time cube (see Fig. 19) and so on.

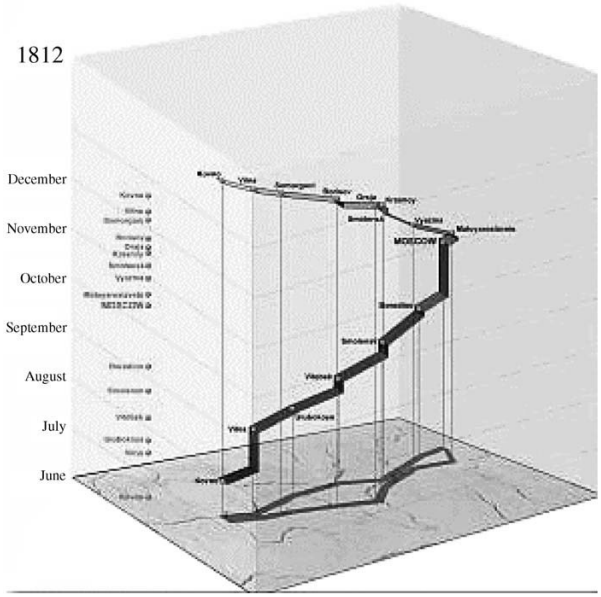

Fig. 19: Representation of Minard's Map in space–time cube (Andrienko, 2003)

In (Lee et al., 2005) authors illustrate a tool based on ArcGis to investigate Historic site changes. In their interface (see Fig. 20) animation maps, time-life bar and 3D model view of the History site are used. Users can navigate only by time, stopping, forwarding or rewinding the animation map. The maps, the time life bar and the 3D model are not interactive.

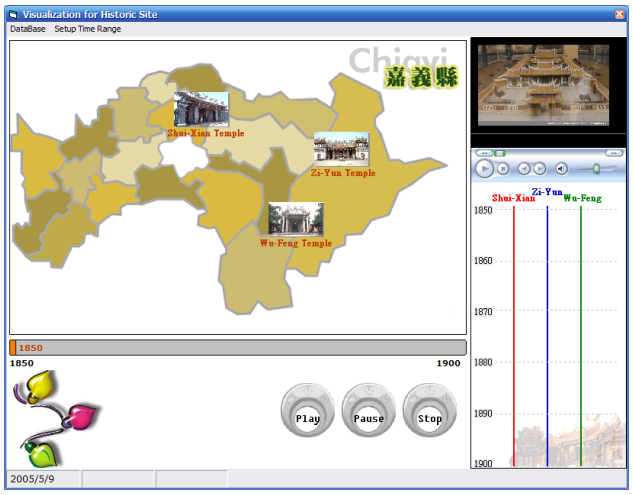

Fig. 20: (Lee et al., 2005) web interface prototype

Stefanakis in (Stefanakis, 2008) presents a prototype educational framework for modeling, analyzing and visualizing the Ancient Greek Mythology. He deals with events without any absolute time reference which are related each other by topological relations. He proposes a web interface (see Fig. 21) that shows "*Myths*" on a map. In this interface temporal navigation is missing and granularity of space and time is missing as well.

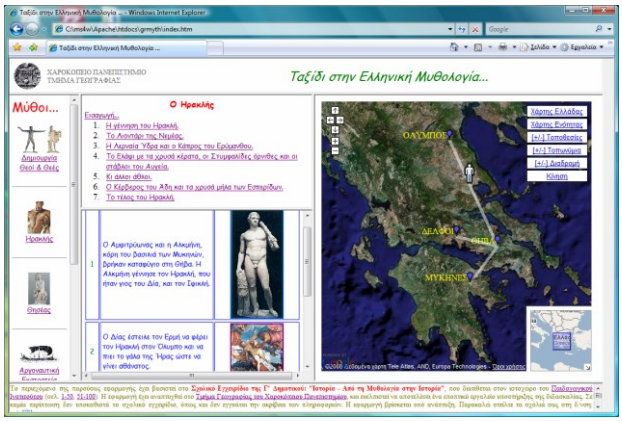

Fig. 21: (Stefanakis, 2008) Web Application.

In (Becker et al., 2009) authors combines OGC standards and SVG SMIL to produce animated maps for moving objects data.

In (Bertolotto et al., 2007), authors present an interface for browsing the outcome of a data mining process on spatio-temporal datasets. The interface allows the decision maker to interact both on the spatial dimension (through an

interactive map), both on the temporal one (through a slider). However, differently from the proposal, in that work the temporal aspect is much simpler, being limited to a fixed granularity.

3.2.2 GeoVisualization

As stated by McCarren "*Geovisualization integrates approaches from visualization in scientific computing (ViSC), cartography, image analysis, information visualization, exploratory data analysis (EDA), and geographic information systems (GISystems) to provide theory, methods, and tools for visual exploration, analysis, synthesis, and presentation of geospatial data*" (MacEachren & Kraak, 2001).

One of the most effective geovisualization techniques is thematic mapping. A thematic map is a color-coded maps that describes the geographic distribution or numerical values or descriptive of a variable. They reveal the geographical connections of data to which they refer (Shekhar & Xiong, 2008).

Jaques Bertin established a graphic system of visual variables, which represents an universally recognized theory of the cartographic transcription of geographical information (Koch, 2001). The intent of Bertin was to define basic visual symbols in order to define a standard way to represent information (see Fig. 22). As the basic unit he uses the "Mark" and defines a series of characteristics of this unit that can be modified in order to differentiate the representation (position, size, shape, color).

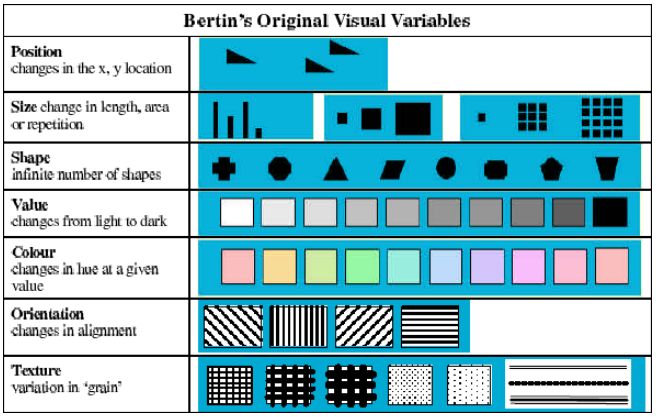

Fig. 22: Bertin Visual Variables

Bertin's system has been subsequently modified by various cartographers, for example works of (Robinson et al., 1995)and (Slocum et al., 2009)defines four types of visual variables (point, line, area and volume) and for each of them they provide the appropriate thematic mapping techniques.

A *point visual variable* refers to a particular location in space, and is used when the geographical phenomena being mapped is located at a place or is aggregated to a given location (MacEachren, 1979). Differentiation among point symbols is achieved by using visual variables, like size, color and shape. Thematic maps using point symbols are:

- *Dot maps*: here one dot represents a unit of some phenomena, and dots are placed at locations where the phenomenon is likely to occur (Slocum et al., 2009). See Fig. 23.

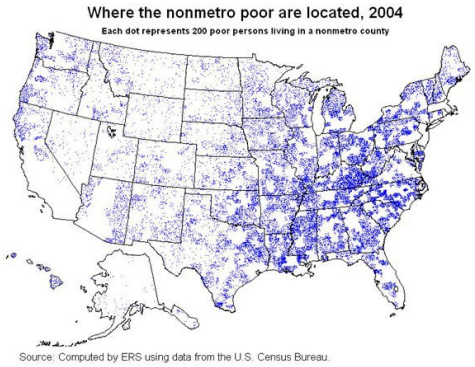

Fig. 23: Dot Map (Wikipedia, 2013)

- *Proportional symbol Map*: in this kind of map a symbol is used to represent the data on the map. The size of the symbol varies according to the value of the datum referenced (Sandvik, 2008)(see Fig. 24).

Fig. 24: Proportional Symbol Map (Sandvik, 2008)

The *line visual variables* are used to indicate connectivity or flow, equal values along a line and boundaries between unlike areas (MacEachren, 1979). The shape, color or form (e.g. solid vs dotted) of line symbols changes according to the value of the data they represent. Common thematic maps using line symbols are:

- *Flow maps*: this maps utilise lines of differing width to depict the movement of phenomena between geographical locations (Slocum et al., 2009), see Fig. 25.

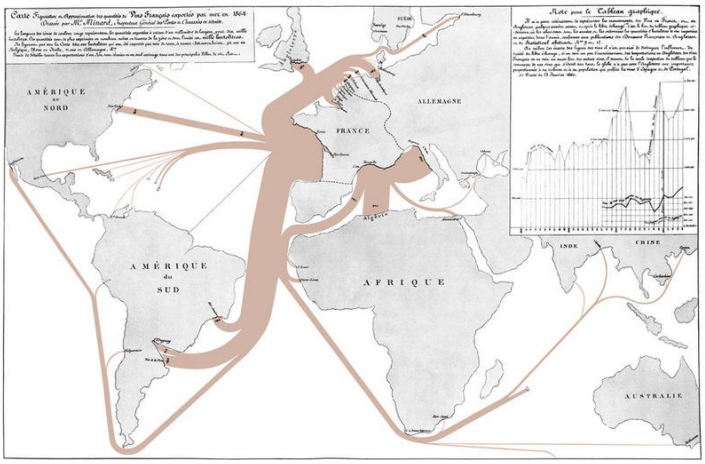

Fig. 25: Flow Map (Wikipedia, 2013)

- *Isarithmic maps*: this maps depict smooth continuous phenomena, like rainfall or barometric pressure (Slocum et al., 2009), see Fig. 26.

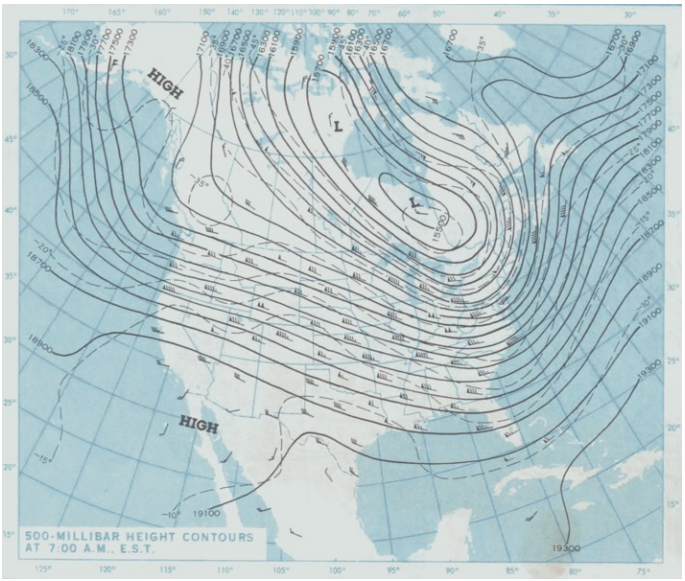

Fig. 26: Isarithmic map (Wikipedia, 2013)

The area visual variable are used to assign a characteristic or value to a whole area on a map. Bertin visual variables used are color, texture and perspective height (Slocum et al., 2009). The most commonly employed map is the *choropleth map*: it is a map in which the geographical areas are shaded in proportion to the value of the data displayed on the map, as for example the density of population. Is a good way to see how the values of a given vary according to region. (Wikipedia, 2013), see Fig. 27.

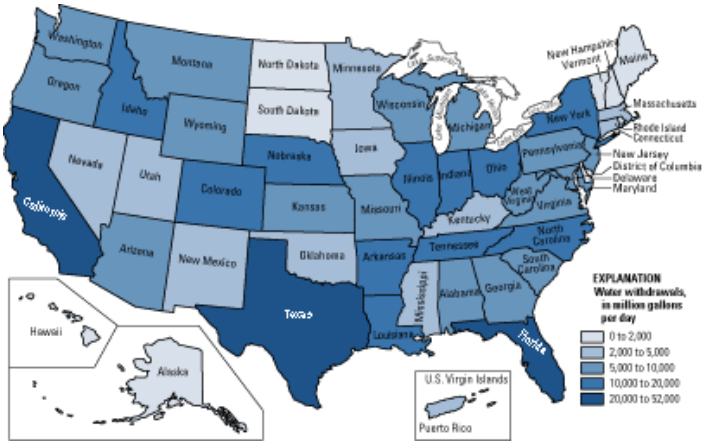

Fig. 27: Choropleth map (Wikipedia, 2013)

The volume visual variable uses the x and y coordinates to locate the point on the map and the z coordinate to represent the data value, as can be seen in a prism map (see Fig. 28).

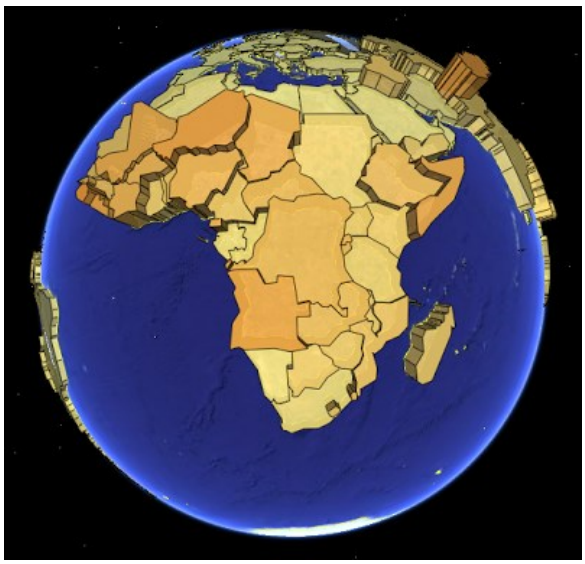

Fig. 28: Prism Map (Sandvik, 2008)

More recent works merges text-based information visualization techniques with geovisualization. In (Slingsby et al., 2007) authors use the Tag Cloud technique in order to visualize spatial related text as shown in Fig. 29. Tag clouds are a visualization technique developed for assisting in this process by summarizing the relative importance of tags (Hassan-Montero & Herrero-Solana, 2006). Each tag is displayed, usually in alphabetical order, at a size according to some measure of its prominence.

Fig. 29: Tag Cloud-map

In (Slingsby et al., 2008) authors uses the *Tree Map* technique (Johnson & Shneiderman, 1991) in order to visualize spatio-temporal data (see Fig. 30). Tree Map are a rectangular, space-filling approach for visualizing hierarchical data. They use 2D visualization of trees where the tree nodes are encapsulated into the area of their parent node. The size of the single nodes is determined proportionally in relation to all other nodes of the hierarchy by an attribute of the node (InfoVis, 2013).

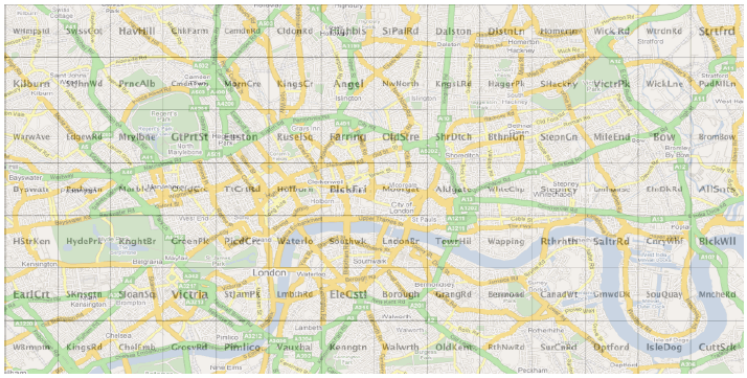

Fig. 30: Tree-map (Slingsby et al., 2008)

Part II
Proposed Work

Chapter 4 Data Model

The capability of any information system largely relies on the design of its data model. A rigorous data model must be able to manage and foresee any required spatial, temporal and spatio-temporal queries and analytical methods (Yuan, 1996). In the following sub-section will be described how the data-model manages:

- the abstract meta-model that allows users to describe features having spatial, temporal, spatiotemporal and descriptive properties;
- the temporal domain, using an original model to handle user defined contexts, granularities and qualitative temporal references;
- the spatial domain, using a model inspired by the OGC standards able to manage semantic, absolute and uncertain spatial references;
- the integration of space and time properties, managing the various types of spatio-temporal data.

4.1 Meta-Model

The underlying idea proposed in this work consists to provide a framework that allows users to model spatial, temporal and spatio-temporal features regardless their specific application domain. Starting from the triad model (Peuquet, 1994) we will consider our data as just having a set of properties, regardless their semantics meaning.

The RDF data model proposed by W3C seems to suit perfectly this purpose. As illustrated in Part IChapter 2 a resource is represented by a triple *<Subject, Predicate, Object>.* Each triple represents a statement of a relationship between the things denoted by the nodes that it links (W3C, 2004). Therefore a triple represents a link (predicate) between two nodes *(subject, object)*; the set of all triples generates an *RDF Graph*.

An object/feature instance in the triad model can be seen as a set of triple where the *subject* is the feature itself, the *predicates* are the different properties

(spatial, temporal and descriptive) and the *objects* are the corresponding property values.

Using RDF Schema concepts it is possible to model a generic feature defining classes and properties in a way very similar to the OO design.

One of the more important advantages of using RDF data model is that the model and the data have the same data structure (graph) and one can query both of them with the same query languages (SPARQL), even if the specific data model is unknown. This is a very important feature in order to be independent from the specific domain application, because we can refer to a property as a general predicate, which will be instantiated and personalized by user needs.

Unfortunately, SPARQL does not (still) support spatial and temporal operators. Some works introduce spatial and temporal query algebra, but none of them are at the present time a W3C recommendation. Also the state of art models does not provide support to the thematic and hierarchical stratification of the spatial and temporal domain proposed in this thesis.

To overcome this limitation this work provides a data model specification for spatial, temporal and spatio-temporal domains. The "semantic" entities are modeled using RDF/OWL structures while the specific operators are implemented using ad hoc data structures and/or defined standards.

In order to allow users to define a generic Feature, an RDF(S)/OWL meta-model is designed, that allows the definition of spatial, temporal, spatiotemporal and descriptive properties. To represent a generic application feature two concepts are defined:

Definition 4.1.1 CMSFeature

CMSFeature is an *rdfs:Class* that represent a generic class in the CMS.

Definition 4.1.2 CMSProperty

CMSProperty is a *rdf:Property* that represents a generic CMS property. It has as domain a *CMSFeature*.

CMSProperty is an abstract concept, the model defines four sub-properties,

with adequate range and domain that allows the definition of the feature dimension.

Definition 4.1.3 temporalProperty

A *temporalProperty* is a sub-property of *CMSProperty* and allows linking a *CMSFeature* to an element of the temporal domain (i.e. *Period-Event*, defined in section 4.2.2).

Definition 4.1.4 spatialProperty

A *spatialProperty* is a sub-property of *CMSProperty* and allows linking a *CMSFeature* to an element of the spatial domain (i.e. *SpatialReference*, defined in section 4.3.2).

Definition 4.1.5 spatioTemporalProperty

A *spatialProperty* is a sub-property of *CMSProperty* and allows relating a *CMSFeature* to a spatiotemporal phenomenon (see section 4.3.4.2).

Definition 4.1.6 descriptiveProperty

A *descriptiveProperty* is a sub-property of *CMSProperty* and represents the descriptive dimension of a *CMSFeature*. This link can be expressed by linking the *CMSFeature* to a simple-type (string, numeric, etc.) or to a complex-type represented by another *CMSFeature*.

The Meta-Model provides an ontology of descriptive property to express simple and complex types, as showed in the Meta-Model graphical representation in Fig. 31.

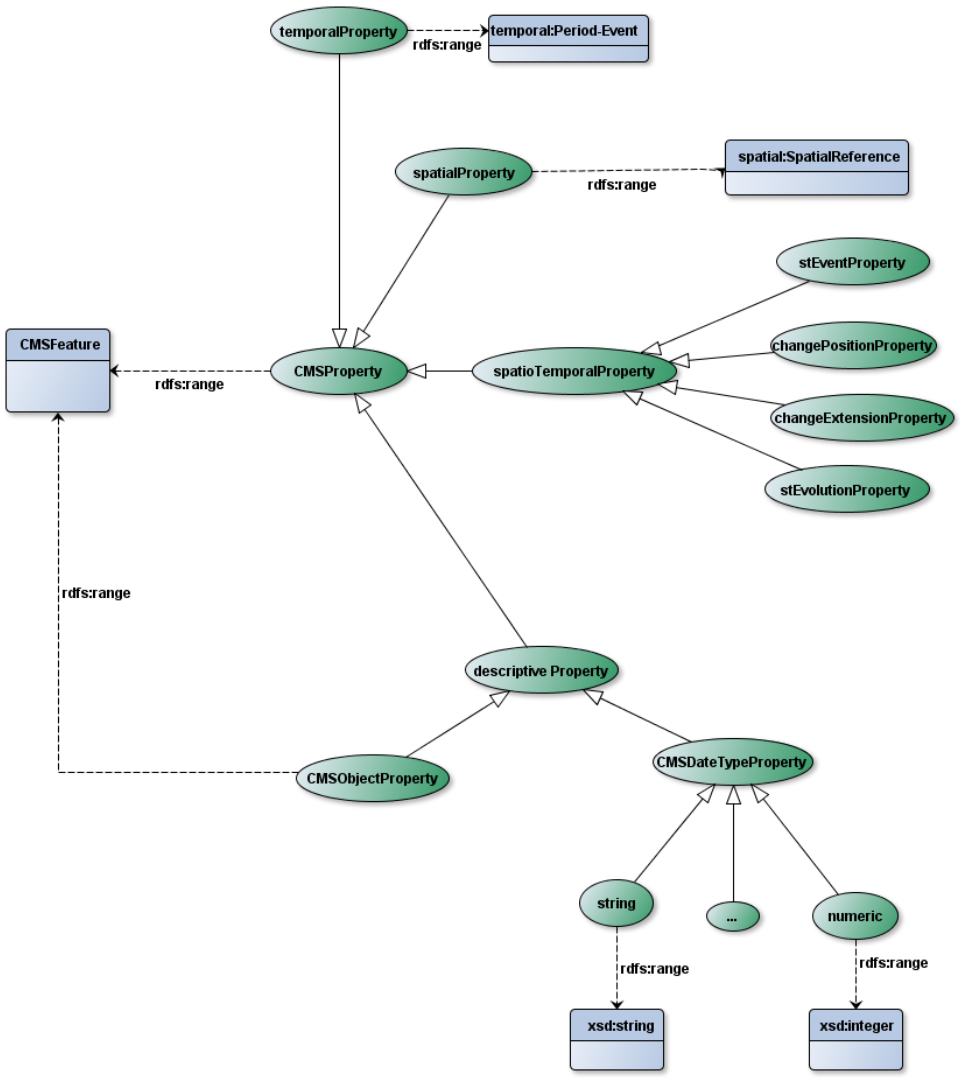

Fig. 31: Meta-Model Schema

4.1.2 RDF Model

The concepts defined in the previous section are described in RDF model, generating the schema showed in Fig. 31 and listed in Table 8.

Table 8: RDF Meta-Model

CMSFeature	
Type	rdf:Class
CMSProperty	
Type	owl:objectProperty

Domain	CMSFeature
temporalProperty	
Type	owl:objectProperty
SubPropertyOf	CMSProperty
Domain	CMSFeature
Range	temporal:PeriodEvent
spatialProperty	
Type	owl:objectProperty
SubPropertyOf	CMSProperty
Domain	CMSFeature
Range	spatial:SpatialReference
spatioTemporalProperty	
Type	owl:objectProperty
SubPropertyOf	CMSProperty
Domain	CMSFeature
Range	spatial:SpatialReference
descriptiveProperty	
Type	rdf:Property
SubPropertyOf	CMSProperty
Domain	CMSFeature
CMSObjectProperty	
Type	owl:objectProperty
SubPropertyOf	CMSProperty
Domain	CMSFeature
Range	CMSFeature
CMSDateTypeProperty	
Type	Rdf:Property
SubPropertyOf	descriptiveProperty
Domain	CMSFeature
stringProperty	
Type	owl:DatetypeProperty
SubPropertyOf	CMSDateTypeProperty
Domain	CMSFeature
Range	xsd:#string

numericProperty	
Type	owl:DatetypeProperty
SubPropertyOf	CMSDateTypeProperty
Domain	CMSFeature
Range	xsd:#integer

4.1.3 Meta-Model Customization

The feature and properties declared by user and managed by the model are an instance of the Meta-Model presented called *Application Schema*.

In order to define own feature users have to declare a *rdf:Class* that is a subclass of *CMSFeature* and create properties that are sub-properties of the one defined above. As an example, suppose that a user wants to model the *Litterary Source* feature. She/he defines a *Feature* having a string property "*title*", an object property "*author*" that points to another *CMSFeature*, a spatial property "*published in*" and a temporal property "*publish date*". Table 9 shows the *Application Schema* generated for this feature.

Table 9: Meta-Model Customization

```
 PREFIX my: somedomainuri
 PREFIX rdf, rdfs, tadaima, owl, spatial, temporal, st.
my:Author          rdfs:type             owl:Class
                   rdfs:subClassOf       CMSFeature
…
my:LiterarySource  rdfs:type             owl:Class
                   rdfs:subClassOf       CMSFeature
my:title           rdfs:type             owl:DateTypeProperty
                   rdfs:subPropertyOf    stringProperty
                   rdfs:Domain           my:LiterarySource
                   rdfs:Range            xsd:string
my:hasAuthor       rdfs:type             owl:ObjectTypeProperty
                   rdfs:subPropertyOf    CMSObjectProperty
                   rdfs:Domain           my:LiterarySource
                   rdfs:Range            my:Author
my:publishPlace    rdfs:type             owl:ObjectTypeProperty
```

	rdfs:subPropertyOf	CMSSpatialProperty
	rdfs:Domain	my:LiterarySource
	rdfs:Range	spatial:SpatialReference
my:publishDate	rdfs:type	owl:ObjectTypeProperty
	rdfs:subPropertyOf	CMSTemporalProperty
	rdfs:Domain	my:LiterarySource
	rdfs:Range	temporal:PeriodEvent

4.2 Temporal Domain

Managing quantitative and qualitative temporal reference is a well-known problem in the temporal database literature ((Koubarakis, 1994), (Chaudhuri, 1988)) and in Cultural Heritage field (Doerr et al., 2004). Theoretical works on temporal domains (Bettini et al., 1996) introduce the concept of temporal granularity, with the aim to study the expressiveness power and the decidability properties of navigational operators on stratified temporal domains.

The temporal model presented in this thesis merges the qualitative and quantitative aspect of time references using the de facto standard model (Doerr et al., 2004) with the granularity concepts studied in theoretical works (Bettini et al., 1996), adding the possibility to organize the temporal domain in thematic contexts, each of them with its granularity and the possibility of making quantitative and qualitative temporal references.

An application temporal domain is a (partially) ordered set of temporal events quantitatively or qualitatively referenced to an absolute temporal axis (from now on, a timeline). The elements on this timeline can assume two kinds of forms: Point (instant) and Set (Interval).

Features having (spatio) temporal nature have a (time) reference to these temporal events. It can happen that some kinds of data do not always present precise purely quantitative time reference (e.g. a point or an interval on the timeline). Temporal Events could be qualitatively referenced to others by ordering relations (before, after etc.) or a Temporal Event may hinge on others.

Moreover, experts would make different partitions on timeline based on a

context of interest and a notion of temporal granularity, and associate temporal events to these different partitions.

It is clear that the basic element of this complex domain, that gives the temporal reference to a feature, should contain information about the partition it belongs to and the qualitative or quantitative link to the temporal axis.

The formal definitions of the concepts illustrated above follows.

4.2.1 Thematic contexts and Time Granularity.

The temporal domain can be organized in *thematic context*. Let us call $Cont=\{C_1, C_2, \ldots, C_n\}$ the set of thematic contexts defined by users. Each context has a finite number of layers, which represents the temporal granularity. Let us call $Layers_C=\{L_1, L_2, \ldots, L_m\}$ with $C \in Cont$ the set of the layers defined for the context *C*.

Formally, a time domain is given by the set of pairwise disjoint pairs $(T_{C,L}, \leq_L)$, where $T_{C,L}$ is a non-empty set of *Temporal-Entities* in a context *C*, *L* is a layer in the context *C* and $\leq_L$ is a partial linear order relation on $T_{C,L}$. The ordering relation holds on *Temporal-Entities* belonging to the same Layer. The definition of time granularity is inspired by (Bettini et al., 1996). For a context *C*:

- L_1: contains *Instant* and atomic *Temporal Interval*.
- L_i with $i \geq 2$: contains the remaining not atomic *Temporal-Entities*.

In the model of time considered in this work, the temporal entities are *Period-Event* objects. A *Period-Event* is an event or a time interval in a specific thematic context and it is qualified by a given level of granularity.

4.2.2 Quantitative and Qualitative Temporal Reference.

In the assumed model the temporal reference of an object is given by the abstract concept *Period-Event*.

If the temporal reference is quantitative, then the *Period-Event* has a *temporal_extension* property that refers to an abstract object called *Temporal-Entity*. A *Temporal-Entity* could assume three forms:

1. a point extension, called *Temporal Instant*, indicated by *t*;

2. an interval extension, called *Temporal Interval,* indicated as *i=[s,t]* and defined by a start temporal reference *s* and a final temporal reference *t* where *s* and *t* could be *Temporal-Entities*. *N:B a Period-Event is a Temporal-Entity itself, in this model is possible to define interval in terms of instant, interval and quantitative period events.*
3. a pivot extension, that models the temporal events that hinge on other events, called *Temporal Offset* and defined by the triple *i=(off_ante, e, off_post)* where *e* is a *Temporal-Entity*, *off_ante* is the number of instants *i* preceding *e* and *off_post is* the number of instants *i* following *e*. A *Temporal Offset* can be mapped to a Temporal Interval.

If the temporal reference is qualitative, then the *Period-Event* object has a *temporal_relationship* property that refers to another *Period-Event* object in the same *Context*. The *temporal_relationship* properties are a subset of the Allen Temporal Property (Allen, 1991)*: after, before, during, contains, overlaps, equal.* The mapping of the original property to the subset used in the model is shown in Table 10.

Table 10: Allen' Properties mapping

Original Allen Property	Mapping to "reduced" Allen Property
Before	Before
Meet	Before
After	After
Met by	After
Contains	Contains
Started by	Contains
Ended by	Contains
During	During
Starts	During
Ends	During
Overlaps	Overlaps
Overlapped by	Overlapped By
Equal	Equal

4.2.3 Rdf Model

The structure of the adopted time model has been designed using RDFS/OWL data model (see Fig. 32).

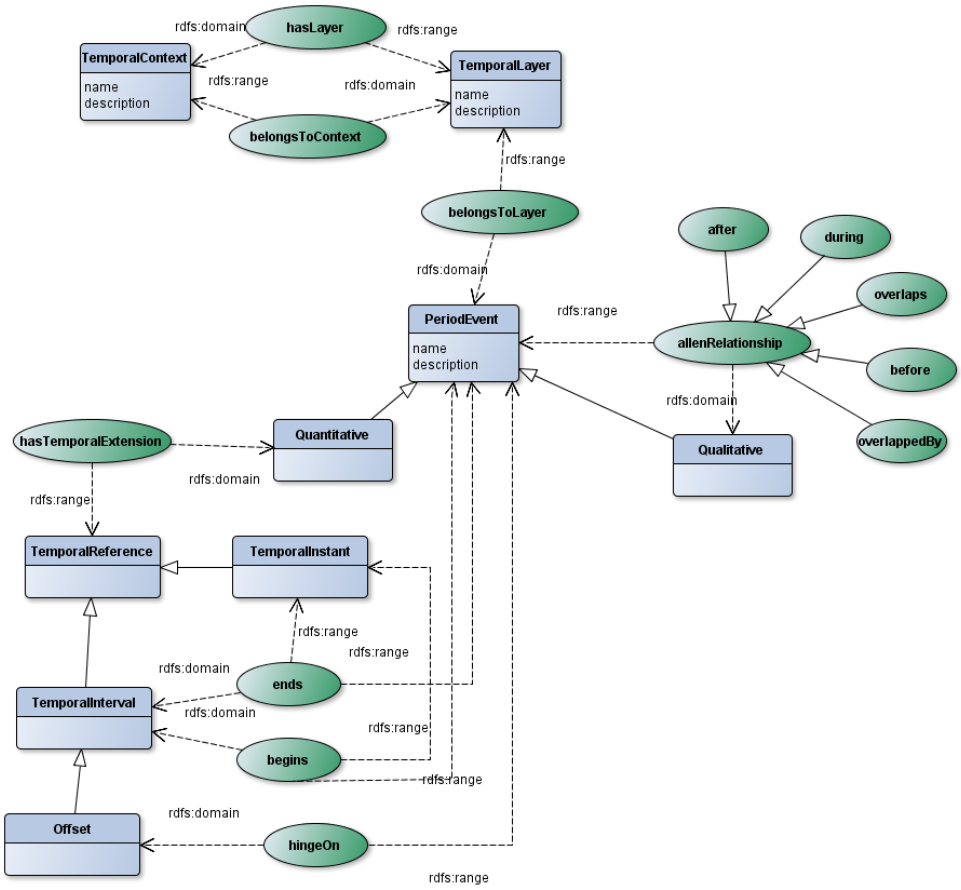

Fig. 32: Temporal Model schema.

The concepts of *TemporalContext* and *TemporalLayers* are represented by their respective classes. Users can define own hierarchy order between *TemporalLayers* in the same *TemporalContext* specifying the properties *greaterThan* and *finerThan*.

The concepts of qualitative and quantitative reference are represented by the *Period-Event* class and its subclasses *Quantitative* and *Qualitative*.

It follows (see Table 11) a brief description of the RDF classes and their properties.

Table 11: Temporal Domain RDF Schema

TemporalContex		
Type	rdf:Class	
Properties	***name***	
	Type	owl:DatetypeProperty
	Domain	TemporalContex
	Range	xsd:#string
	description	
	Type	owl:DatetypeProperty
	Domain	TemporalContex
	Range	xsd:#string
	hasLayer	
	Type	owl:ObjectProperty
	Domain	TemporalContext
	Range	TemporalLayer
TemporalLayer		
Type	rdf:Class	
Properties	***name***	
	Type	owl:DatetypeProperty
	Domain	TemporalLayer
	Range	Xsd:#string
	description	
	Type	owl:DatetypeProperty
	Domain	TemporalLayer
	Range	xsd:#string
	belongsToContext	
	Type	owl:ObjectProperty
	Domain	TemporalLayer
	Range	TemporalContex
	isFinerThan	
	Type	owl:ObjectProperty
	Domain	TemporalLayer
	Constraints	TemporalLayer
	isGreatherThan	

	Type	owl:ObjectProperty
	Domain	TemporalLayer
	Constraints	TemporalLayer
PeriodEvent		
Type	rdf:Class	
Properties	***name***	
	Type	owl:DatetypeProperty
	Domain	PeriodEvent
	Range	xsd:#string
	description	
	Type	owl:DatetypeProperty
	Domain	PeriodEvent
	Range	xsd:#string
	belongsToLayer	
	Type	owl:ObjectProperty
	Domain	PeriodEvent
	Range	TemporalLayer
Quantitative		
Type	rdf:Class	
SubclassOf	PeriodEvent	
Properties	***hasTemporalExtension***	
	Type	owl:objectProperty
	Domain	Quantitative
	Range	TemporalReference
TemporalReference		
Type	rdf:Class	
TemporalInstant		
Type	rdf:Class	
SubclassOf	TemporalReference	
TemporalInterval		
Type	rdf:Class	
SubclassOf	TemporalReference	
	begins	

	Type	owl:objectProperty
	Domain	TemporalInterval
	Range	TemporalReference or Quantitative
		ends
	Type	owl:objectProperty
	Domain	TemporalInterval
	Range	TemporalReference or Quantitative
Offset		
Type	Rdf:Class	
SubclassOf	TemporalInterval	
		hingeOn
	Type	owl:objectProperty
	Domain	Offset
	Range	PeriodEvent
		offsetAnte
	Type	owl:DatatypeProperty
	Domain	Offset
	Range	xsd:#integer
		offPost
	Type	owl:DatatypeProperty
	Domain	Offset
	Range	xsd:#integer
Qualitative		
Type	Rdf:Class	
SubclassOf	PeriodEvent	
		allenRelationship
	Type	owl:objectProperty
	Domain	Qualitative
	Range	PeriodEvent

4.2.4 Allen Temporal Inconsistence

The temporal relationships defined in (Allen, 1991) and used to link a period event to another by a qualitative temporal reference, are related one to the other by the transitivity property as showed in Fig. 33.

B r2 C / A r1 B	<	>	d	di	o	oi	m	mi	s	si	f	fi
"before" <	<	no info	< o m d s	<	<	< o m d s	<	< o m d s	<	<	< o m d s	<
"after" >	no info	>	> oi mi d f	>	> oi mi d f	>	> oi mi d f	>	> oi mi d f	>	>	>
"during" d	<	>	d	no info	< o m d s	> oi mi d f	<	>	d	> oi mi d f	d	< o m d s
"contains" di	< o m di fi	> oi di mi si	o oi dur con =	di	o di fi	oi di si	o di fi	oi di si	di fi o	di	di si oi	di
"overlaps" o	<	> oi di mi si	o d s	< o m di fi	< o m	o oi dur con =	<	oi di si	o	di fi o	d s o	< o m
"over-lapped-by" oi	< o m di fi	>	oi d f	> oi mi di si	o oi dur con =	> oi mi	o di fi	>	oi d f	oi > mi	oi	oi di si
"meets" m	<	> oi mi di si	o d s	<	<	o d s	<	f fi =	m	m	d s o	<
"met-by" mi	< o m di fi	>	oi d f	>	oi d f	>	s si =	>	d f oi	>	mi	mi
"starts" s	<	>	d	< o m di fi	< o m	oi d f	<	mi	s	s si =	d	< m o
"started by" si	< o m di fi	>	oi d f	di	o di fi	oi	o di fi	mi	s si =	si	oi	di
"finishes" f	<	>	d	> oi mi di si	o d s	> oi mi	m	>	d	> oi mi	f	f fi =
"finished-by" fi	<	> oi mi di si	o d s	di	o	oi di si	m	si oi di	o	di	f fi =	fi

Fig. 33: The Transitivity Table for the 12 temporal relationship (ommitting =) (Allen, 1983)

The propagation of these transitivity properties generates a graph of temporal relationship that in some cases could be inconsistent. In Fig. 34 dot arrows represent the relationship inferred by transitivity from the declared ones (represented by continuous arrows).

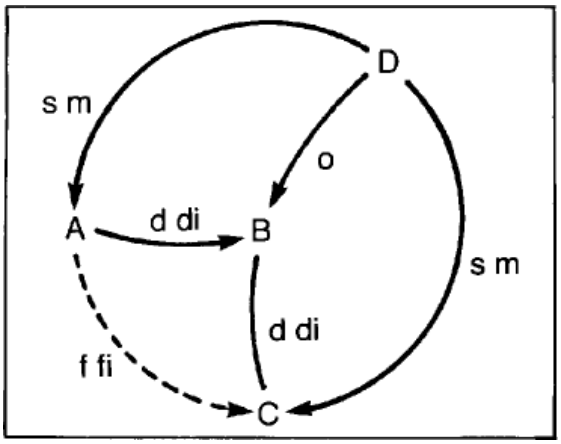

Fig. 34: An Inconsistent Labeling (Allen, 1983)

Allen proposed an algorithm to identify and prevent this inconsistence, also optimized and implemented by (Vilain et al., 1989) and (van Beek & Manchak, 1996). It has been proved (Allen, 1983) that the algorithm is correct but not complete, due to the nature of the problem itself.

In the temporal model proposed in this thesis the inconsistence can be introduced by user in the definitions of qualitative temporal definition chain. See the example in Fig. 35 user define three qualitative *Period-Event* declaring the temporal relationship represented by the continuous line. The transitivity propagation of those properties, represented by the dotted line, make the graph inconsistent. To overcome this limitation we check on request the consistent of the temporal graph using the Allen Algorithm.

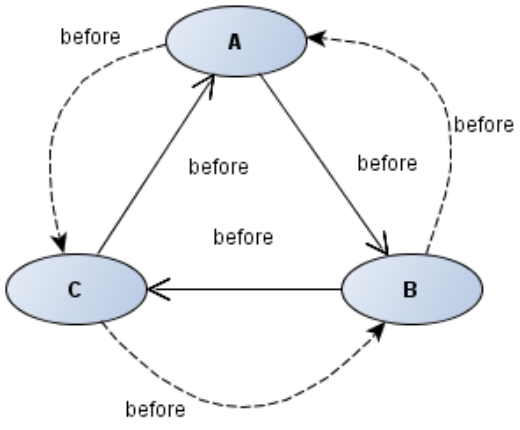

Fig. 35: Time Inconsistence Example

In order to simplify the retrieval of temporal information we introduce a constraint on the chain of temporal qualitative relationships that does not affect the expressivity of the proposed model.

Definition 4.2.1 Chain constraint

A chain of qualitative temporal relationship between *Period-Event* **must** end with a quantitative *Period-Event*.

4.2.5 Temporal Operators Implementation

As previously said SPARQL does not (yet) provide support for temporal queries. To overcome this limitation we implement the temporal operator on quantitative period event using an ad hoc data structure: a *Temporal Interval Tree (TIT).*

This data structure is based on the Interval Tree (Cormen et al., 2009), a particular red and black tree able to manage intervals. This data structure guarantee insertion and search operation in O(log *n*) time, where *n* is the number of nodes.

4.2.5.1 Interval Tree data structure

The state of art offers various data structure in order to manage temporal intervals for example (Cormen et al., 2009) and (De Berg et al., 2008). In this work the (Cormen et al., 2009) implementation is used and extended. The formal definition of interval, overlap and interval trichotomy follows.

Definition 4.2.2 Interval

An ***interval*** is an ordered pair of real numbers [$t1$, $t2$], with $t1 \leq t2$. An interval [$t1$, $t2$] is represented as an object i, with fields $low[i] = t1$ (the ***low endpoint***) and $high[i] = t2$(the ***high endpoint***) (Cormen et al., 2009) .

Definition 4.2.3 Overlap

Let be i, and i' intervals. We say that i and i' ***overlap*** if $i \cap i' \neq \emptyset$, that is, if $low[i] \leq high[i']$ and $low[i'] \leq high[i]$.

Definition 4.2.4 Interval Trichotomy

Any two intervals *i* and *i'* satisfy the ***interval trichotomy***; that is, exactly one of the following three properties holds:

1. *i* and *i'* overlap,
2. *i* is to the left of *i' (i.e., high[i]< low[i']),*
3. *i* is to the right of *i' (i.e., high[i']< low[i]).*

The underlying data structure is a red-black tree in which each node *x* contains an interval *int*[*x*] and the key of *x* is the low endpoint, *low*[*int*[*x*]], of the interval.

In addition to the intervals themselves, each node *x* contains a value *max*[*x*], which is the maximum value of any interval endpoint stored in the subtree rooted at *x*. Is proven (Cormen et al., 2009) that updating this additional information cost *O(1)* time, so insertion and deletion can still be performed in *O*(lg *n*) time on an interval tree of *n* nodes.

Definition 4.2.5 Definition IntervalTreeNode

An *IntervalTreeNode* is a basic element of an *Interval Tree*. It is composed by the following field:

- *left:* pointer to the left child of the node;
- *right:* pointer to the right child of the node;
- *int[begin, end]:* interval (Definition 4.2.2) that act as key for the ordering relationship;
- *element:* information about the object that the nodes represents.

A new search operation is defined namely INTERVA-SEARCH (T,i), with T being am Interval Tree and I an interval.

Definition 4.2.6 Interval-Search(T,i)

The Interval-Search(T,i) operation returns a pointer to an element x in the interval tree *T* such that *int*[*x*] **overlaps** (Definition 4.2.3) interval *i*, or the sentinel *nil*[*T*] if no such element is in the set.

4.2.5.2 Temporal Interval Tree

In order to manage the described temporal domain, the (Cormen et al., 2009) interval tree has to be configured and extended. The implementation described in the following paragraph will be called *Temporal Interval Tree* (TIT).

TIT takes as input an RDF-Graph based on the temporal domain schema (see Fig. 32) and produces a forest of interval tree able to manage Allen's temporal operator on the Quantitative Period Event.

The definition of interval seen for an interval tree (Definition 4.2.2) and the concept of "overlaps" differs a little from the ones described by Allen and used in the temporal domain model. To avoid confusion we redefine the concepts of interval, interval trichotomy and the operation of Interval-Search in terms of Allen property as follows.

Definition 4.2.7 Interval

An ***interval*** is an ordered pair of link to the temporal axes, called **datation**, [t_1, t_2], with t_1 **before** or **equal** $t2$. An interval [t_1, t_2] is represented as an object i, with fields $low[i] = t_1$ (the ***low endpoint***) and $high[i] = t_2$ (the ***high endpoint***).

Definition 4.2.8 Interval Trichonomy

Any two intervals i and i' satisfy the ***interval trichotomy***; that is, exactly one of the following properties holds:

1. i **overlaps** i',
2. i **overlapped by** i',
3. i **contains** i',
4. i **during** i',
5. i **equal** i',
6. i is to the left of i' (i.e., i **before** i'),
7. i is to the right of i' (i.e., i **after** i').

Where **overlaps**, **overlapped by**, **contains**, **during**, **equal**, **before** and **after** are the temporal relationships defined in (Allen, 1991).

Definition 4.2.9 Interval-Search

INTERVAL-SEARCH(*T, i*) returns a List of elements *x* in the interval tree *T* such that *int*[*x*] **overlaps** or **overlapped** by or **contains** or **during** or **equal** interval *i*, or the sentinel *nil*[*T*] if no such element is in the set.

The basic element of a Temporal Interval Tree is a *TemporaTreeNode (see)*. It contains all the information about the *Quantitative Period-Event* it represents and exploits its link to the timeline. It has a reference to a list of the qualitative period events (*YellowTreeNode* in Definition 4.2.11) that links to it.

Due to the constraint stated in Definition 4.2.1, it is always possible in this model link a qualitative period event to a quantitative one, but, as shown by the temporal transitivity matrix in (Allen, 1983) the Allen temporal relationship matrix, it is not always possible infer the relationship. Therefore a *YellowTreeNode* represents a *Qualitative Period-Event* and contains information both about the Period-Event it is directly related and the recursively retrieved quantitative period-event at the end of the qualitative relationships chain.

An element of TIT, *TemporalTreeNode*, represents a quantitative period event, this structure contains information about period event *uri*, *context*, *layer* and obviously *begin* and *end temporal references*. It also contains a reference to the elements, *YellowTreeNode*, representing the *Qualitative Period-Event* that (recursively) links to the *Quantitative Period-Event* it represents. Due to the constraint stated in Definition 4.2.1, it is always possible in this model link a qualitative period event to a quantitative one, but, as shown by the temporal transitivity matrix in (Allen, 1983) the Allen temporal relationship matrix, it is not always possible infer the relationship.

Definition 4.2.10 TemporalTreeNode

A *TemporaTreeNode* is the basic element of a Temporal Interval Tree. It is an Interval Node (Definition 4.2.5) where its *element* field is structured as:

- *uriPe*: reference to the quantitative *Period-Event* it represents
- *isLinkedBy: list of YellowTreeNodes representing the Qualitative Period-Events that recursively point to it.*

Definition 4.2.11 YellowTreeNode

A *YellowTreeNode* is structure representing a *Qualitative* Period-Event and is composed by the following fields:

- *uriPe:* reference to the qualitative Period-Event it represents,
- *directRefersTo:* reference to the Period-Event it is directly linked,
- *allenRelationship:* temporal relationships holding between the two *Period-Events* in *uriPe* and *directRefersTo*
- *refersTo:* link to the quantitative *Period-Event* it has to recursively points for the constraint in *Definition 4.2.1.*

Therefore TIT provides for each thematic context in the temporal domain a full temporal ordering for the *Quantitative* Period-Event and supports the identification of the link to the temporal axis for the *Qualitative* Period-Event .

For example, referring to Fig. 36 we can say that:

- B **before** or **overlaps** or **contains** or **equal** A
- C **after** or **overlapped by** or **during** A

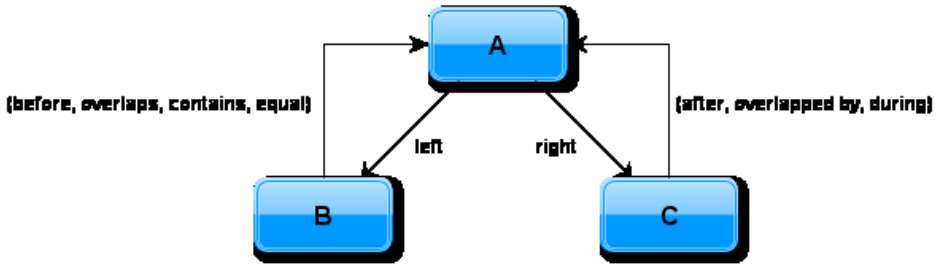

Fig. 36: Temporal relation between nodes of Temporal Interval Tree

It follows the description about how mapping the RDF model to the presented data structure.

4.2.5.3 Temporal Interval Forest Creation

The pseudo-code of the Temporal Interval Forest creation is showed in Table 12. The first step is to create a TIT for each thematic context in the domain (Table 12, line 02-12).

Table 12: Temporal Interval Forest creation algorithm

```
   function createTemporalForest(RDFGraph G){
01    Map<TemporalContext, IntervalTree> temporalForest;
02    foreach TemporalContext C in getContext(G) {
03      Map<PeriodEvent, TemporalTreeNode> mapNode;
04      mapNode=getQuantitativePEasTreeNode(G,C);

05      foreach YellowTreeNode yn in getQualPEasYellowNodes(G,C);
06         TemporalTreeNode tN=  mapNode.get(yn.elem.refersTo)
07         tn.elem.isLinkedBy.add(yn);
08         map.update(tn);

09       TemporalIntervalTree tree= new IntervalTree();
10       foreach TemporalTreeNode node in mapNode
11           tree.insert(node);
12       temporalForest.put(C, tree);}}
```

The first step of the TIT creation is to retrieve all *Quantitative* Period-Events from the RDF graph (Table 12, line 03-04). In the presented RDF temporal model, a *Quantitative* Period-Event could be linked to the temporal axis directly or indirectly, we have the following cases:

1. PE linked to a *Temporal Instant*.
2. PE linked to a *Temporal Interval* and:
    a. begin is a *Temporal Instant* or *a Period-Event*;
    b. end is a *Temporal Instant* or a *Period-Event*;
    c. Any combination of **a** and **b**.

Furthermore, in order to retrieve the link to the temporal axis, i.e. the temporal instant, and needed for filling the field *int[begin, end]* (see Definition 4.2.5 and Definition 4.2.10) the following graph is generated (Fig. 37):

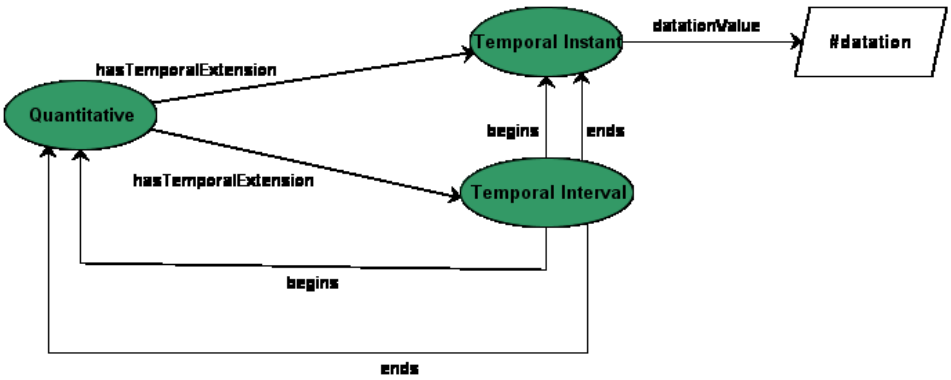

Fig. 37: Quantitative Temporal Reference Graph

The temporal references are recursively resolved during the creation of the temporal tree. This operation benefits from the RDF data structure described in the previous paragraph. Using SPARQL and its property paths simplify the retrieval of the temporal reference effectively linked to the temporal axis. The SPARQL query used to resolve the quantitative temporal reference is showed in Table 13.

Table 13: Sparql query resolving Quantitative Period Event temporal reference

```
//INPUT= $CONTEXT_URI: Uri of a TemporalContext

#Define temporal:beginPath=
((temporal:hasTemporalExtension/temporal:beginsWith>+)+/(<temporal:hasTem
poralExtension>){0,1}
#Define temporal:endPath=
((temporal:hasTemporalExtension/temporal:endsWith>+)+/(<temporal:hasTempo
ralExtension>){0,1}

SELECT  DISTINCT ?pe ?begin  ?end
 WHERE { ?pe rdf:type temporal:Quantitative .
         ?pe temporal:peBelongsToContext <$CONTEXT_URI> .

//Recursively retrieval of begin and end reference
   OPTIONAL {?pe temporal:beginPath ?dat . "
              ?dat rdf:type temporal:TemporalInstant .
              ?dat temporal:datationValue ?begin.

             ?pe temporal:endPath ?datE .
             ?datE rdf:type temporal:TemporalInstant .
             ?datE temporal:datationValue ?endDatationValue .
             } .
//Direct retrieval of begin and end reference
  OPTIONAL { ?pe temporal:hasTemporalExtension ?ti .
              ?ti rdf:type temporal:TemporalInstant .
               ?ti temporal:datationValue ?begin .
               ?ti temporal:datationValue ?end .
               }
       }
```

At this step the retrieved *Period-Events* are transformed in *TemporalTreeNodes* having link to *Qualitative Period-Event* set to null. In order to simplify and to speed up the filling of the missing information, the creation of the TIT is postponed to the retrieval of the *Qualitative Period-Event*, and at this step a Hash-Table of *TemporalTreeNode* is maintained in memory.

As stated before, in order to manage *Qualitative Period-Events* special tree nodes, the *YellowTreeNodes*, are created. Those nodes contain information about the *Qualitative* Period-Event and they are linked to the first *Qualitative* PE (*TemporalTreeNode*) they refer to. In the presented model a qualitative *Period-*

Event could refer to other qualitative period event, generating the graph shown in Fig. 38 with the constraint that the chain of *Qualitative* PE must end with a *Quantitative* one (see Definition 4.2.1).

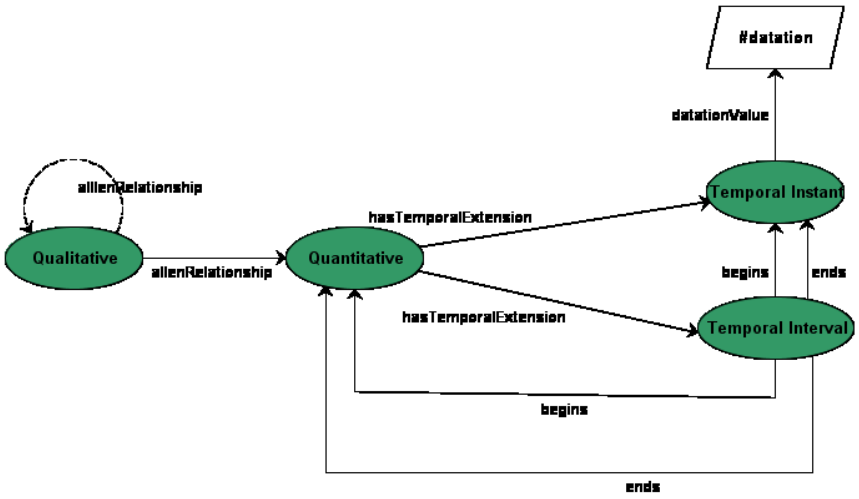

Fig. 38: Qualitative temporal reference graph

Also in this case the RDF structure and the SPARQL property path simplify the retrieval of the reference to the *Quantitative Period-Event* (Table 12, line 05) as shown by the SPARQL query in Table 14.

Table 14: Sparql query for qualitative period-event reference

```
#Define temporal:qualitativePath:= (temporal:before | temporal:after |
temporal:overlaps | temporal:overlappedBy | temporal:during |
temporal:contains)+

SELECT  DISTINCT ?pe ?namePe ?refTo ?directPe
  WHERE { ?pe rdf:type temporal:Qualitative
          ?pe temporal:peBelongsToContext <Context.uri> .
          ?pe temporal:PEname ?namePe .
          ?pe temporal:qualitativePath ?refTo .
          ?refTo rdf:type temporal:Quantitative .
           ?allenProp rdfs:subPropertyOf temporal:allenProperty.
           ?pe ?allenProp  ?directPe }";
```

The qualitative reference are translated in *YellowTreeNode* and linked to the corresponding *TemporalTreeNode* (Table 12, line 05-08).

At this point the information in the temporal HashMap is complete and it is possible to insert the *TemporalTreeNode* in a *TemporalIntervalTree* (Table 12, line 09-11). Finally the *TemporalIntervalTree* created is added to the *Temporal Interval Forest* (Table 12, line 12).

4.3 Spatial Domain

Concepts of thematic context and granularity levels are very familiar also in the spatial domain: the geovisualization theory provides various thematic mapping techniques for managing user thematic contexts and the concept of (granularity) layer is very common in GIS application.

There are many ways to link a feature in a spatial domain, i.e. provide a *spatial reference*. For example a user can give an absolute and precise reference by providing coordinates in terms of latitude and longitude, or by drawing geometry on a geo-referenced map. An element in the spatial domain can be referenced also in an abstract semantic way for example by name (e.g.: Giuseppe Garibaldi (subject) was born in *Nice* (spatial reference)).

It can happen that the spatial reference of an object is not well known, but it could be expressed by a qualitative relationship with another yet specified semantically place (e.g. object A is located at north of place X; is cross place Y, is 10 km far from place Z) or to an absolute reference (geometry) (e.g. object b is contained in the geometry: <(1,1);(2,2);(3,3);(4,4)>).

The spatial domain model proposed in this work aims at managing thematic and granularity partition of the spatial domain and to provide user a flexible way to create spatial elements providing semantic, absolute and qualitative reference.

4.3.1 Thematic context and granularity

A spatial domain is a two or three dimensional geo-referenced area where the problem takes place.

As for the temporal domain, it can happen that users need to organize this domain in more than one *thematic contexts*. For example assuming that our problem is taking place in Italy (the spatial domain), user could make partition of this region on the base of *Political Boundaries*, *Historical Domination*, *Geographical* etc. Each of these *thematic context* can have different granularity levels, e.g. *Layers*. For example user can hierarchically partition the context "*Political boundaries*" in layers: "*Region, District, City*", and use a different

partition for the context "*Historical Domination*" for example "*Kingdom, Republic, Dukedom, Principality*" etc.

The most prominent concept in this domain is the notion of *Place*: abstract entity with semantic meaning, having an *Id*, a *name* and a *description* and that could be represented on the georeferenced spatial domain. The OGC describes how to represent an entity on a georeferenced spatial domain by defining a hierarchy of *Geometries* that allows to draw points, lines, polygons and their collections on the map and that are expressed in terms of geographical coordinate (latitude, longitude and optionally altitude).

The same semantic concept of a *Place* can be represented in different way according to:

- the thematic context/layer it belongs (e.g the place Naples can have different boundaries depending if users are exploring the *Political* or the *Geographical* context);
- the time of validity (e.g. the change of boundaries of a *Place* over time).

This last case will be treated as a *Spatio-Temporal Object* (in particular the *change shape* object) as described in 4.4.1 .

Formally the spatial domain is a subset of R^2 or R^3. This domain can be organized in *thematic context*. Let us call $SpatialCont=\{SC_1, SC_2, \ldots, SC_n\}$ the set of *thematic context* defined by users. Each context has a finite number of layers, that represents the spatial granularity, let us call $SpatialLayers_{SC}=\{SL_{sc,1}, SLx_{sc,2}, \ldots, SL_{sc,m}\}$ with $SC \in SpatialCont$ the set of the layers defined for the context *SC*.

Let be *Place* the abstract representation of an abstract object, having a *name*, a *description* and an *id*. User can associate georeferenced *Geometry* element to a *Place* according to layer and/or the time of validity it belongs. This association generates the object *Place-Instance* represented by the tuple $PI=<P, SL_{SC,X}, Geom, PE>$ where *P* is a *Place*, $SL_{SC,X}$ a *SpatialLayer* in the context *SC*, *Geom* an *OGC Geometry* and *PE* a reference to a *Period-Event*.

A *SpatialLayer* $SL_{C,X}$ is an unordered set of *Place-Instances* $SL_{C,X}=\{PI_1, PI_2,\ldots PI_n\}$ with *C* belonging to *SpaltialCont*, $SL_{C,X}$ belonging to $SpaltialLayers_C$ and $PI_1 \ldots PI_n$ being *Place-Instance*.

4.3.2 Spatial Reference

A *Spatial Reference* represents the way an object is linked to the spatial domain. There are many ways to link a feature in a spatial domain:

- *Semantic*: user relates the object spatial reference in an abstract *semantic* way for example by named entities (e.g.: Giuseppe Garibaldi (*subject*) was born (*spatial property*) in *Nice* (*spatial reference*)). ;
- *Absolute*: user can give an *absolute* and precise reference by providing coordinates in terms of latitude and longitude, or by drawing geometry on a geo-referenced map or by choosing a specific instance of a *Place* (i.e. *PlaceInstance*);
- *Uncertain*: it might be the case that the spatial reference is not well known, but it could be however expressed by a qualitative relationship with another yet specified semantic place or to an absolute reference (e.g. object A is located at north of place X, object B crosses POINT(a,b) etc..).

In the presented meta-model it is possible to link a *CMSFeature* to the spatial domain by a *spatialProperty*. In order to manage the three types of *SpatialReference* we define a hierarchy of abstract *rdf:Class* depicted in the section 4.3.3 .

4.3.3 RDF Model

The concepts of *Context* and *Layers* are represented by their respective classes. Users can define own hierarchy order between *Layers* in the same *Context* specifying the properties *greatherThan* and *finerThan (see* Fig. 39).

In order to manage the three different types of *SpatialReference*, in the proposed model we define an abstract class named *SpatialReference* having three subclasses: *SemanticReference*, *AbsoluteReference* and *UncertainReference.* As detailed in Table 15 the class *Place* is a *SemanticReference,* while the classes *PlaceInstance* and *ogc:Geometry are AbsoluteReference*.

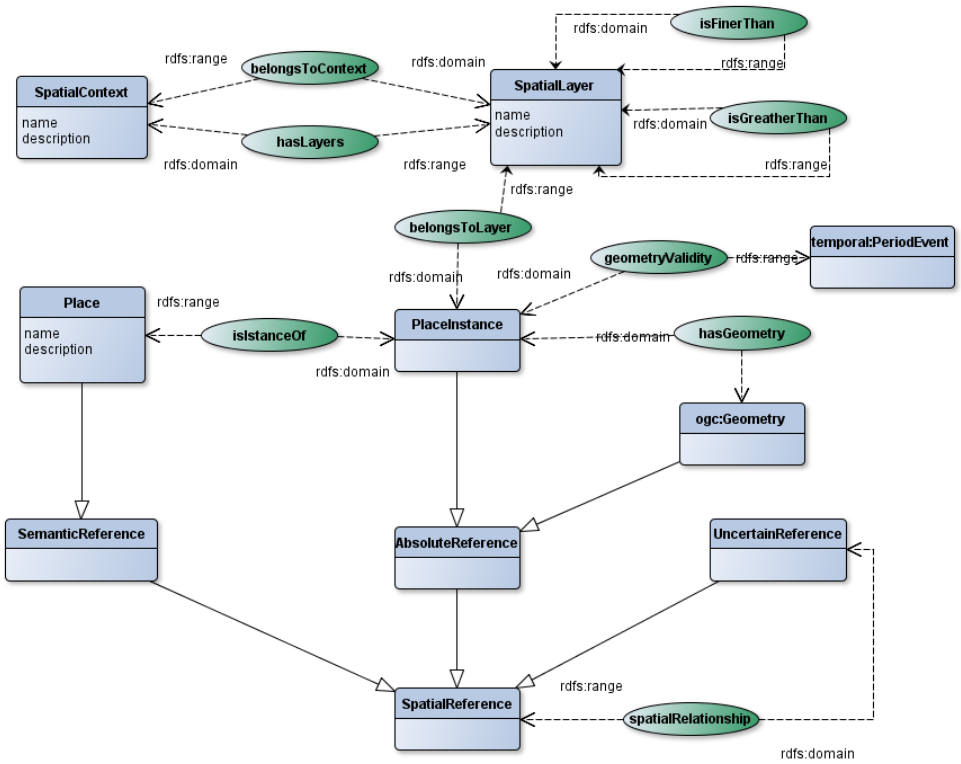

Fig. 39: Spatial Model

An *Uncertain Spatial Reference* links **qualitatively** a *Feature* to an *Absolute* or *Semantic Spatia Reference* by means of a spatial relationship. The spatial relationship allowed are defined in (ISO, 2003): *equals, disjoint, intersect, touches, crosses, within, contains, overlaps, Dwithin*.

Table 15: Spatial Domain RDF model

SpatialContex		
Type	rdf:Class	
Description	This class represents a Spatial Thematic Context	
Properties	***name***	
	Type	owl:DatetypeProperty
	Domain	SpatialContext
	Range	xsd:#string
	description	
	Type	owl:DatetypeProperty

	Domain	SpatialContext
	Range	xsd:#string
		hasLayer
	Type	owl:ObjectProperty
	Domain	SpatialContext
	Range	SpatialLayer
SpatialLayer		
Type	rdf:Class	
Description	This class represents a granularity Layer of a Thematic Context	
Properties		***name***
	Type	owl:DatetypeProperty
	Domain	SpatialLayer
	Range	Xsd:#string
		description
	Type	owl:DatetypeProperty
	Domain	SpatialLayer
	Range	xsd:#string
		belongsToContext
	Type	owl:ObjectProperty
	Domain	SpatialLayer
	Range	SpatialContext
		isFinerThan
	Type	owl:ObjectProperty
	Domain	SpatialLayer
	Constraints	SpatialLayer
		isGreatherThan
	Type	owl:ObjectProperty
	Domain	SpatialLayer
	Constraints	SpatialLayer
SpatialReference		
Type	rdf:Class	
Description	This abstract class represents the way a *CMSFeature* can be linked to the spatial dimensions.	
SemanticReference		
Type	rdf:Class	

SubclassOf	SpatialReference	
Description	This abstract class represents the way a *CMSFeature* can be linked to the spatial dimension by a semantic association. It is implemented by its subclass Place.	
AbsoluteReference		
Type	rdf:Class	
SubclassOf	SpatialReference	
Description	This abstract class represents the way a *CMSFeature* can be linked to the spatial dimension by a semantic association. It is implemented by its subclass Place.	
Place		
Type	rdf:Class	
SublcassOf	SemanticReference	
Description	This class allows the semantic representation of a spatial entity.	
Properties	***name***	
	Type	owl:DatetypeProperty
	Domain	Place
	Range	xsd:#string
	description	
	Type	owl:DatetypeProperty
	Domain	Spatial
	Range	xsd:#string
PlaceInstance		
Type	rdf:Class	
SubclassOf	AbsoluteReference	
Description	This class allows associating semantic places to an absolute spatial reference by providing a layer, geometry and optionally a time of validity.	
Properties	***isInstanceOf***	
	Type	owl:objectProperty
	Domain	PlaceInstance
	Range	Place
	belongsToLayer	
	Type	owl:objectProperty
	Domain	PlaceInstance
	Range	SpatialLayer

	hasGeometry	
	Type	owl:objectProperty
	Domain	PlaceInstance
	Range	ogc:Geometry
	geometryValidity	
	Type	owl:objectProperty
	Domain	PlaceInstance
	Range	temporal:PeriodEvent
ogc:Geometry		
Type	rdf:Class	
SubclassOf	AbsoluteReference	
Description	This class is the one defined by the OGC Standard. Making it a subclass of *AbsoluteReference* allows user to use a standard geometry to link a *CMSFeature* in the spatial dimension.	
UncertainReference		
Type	rdf:Class	
SubclassOf	SpatialReference	
Description	This class express a uncertain spatial entity by relating an Absoulute or a Semantic reference qualitatively by means of a (ISO, 2003) spatial relationship.	
	spatialRelationship	
	Type	owl:objectProperty
	Domain	UnceratinReference
	Range	SemanticReference or AbsoluteReference
	Declared	equals, disjoint, intersect, touches, crosses, within,
	SubProperties:	contains, overlaps, withinDistance

4.3.4 Spatial Operator Implementation

Querying spatial data is assuming a relevant importance in the semantic web field. The OGC has defined an RDF Schema (Brickley & Guha, 2004) that implements the nowadays standard to model and express spatial data (ISO, 2003). There is also a proposal of standard to query this RDF data, namely GeoSPARQL (OGC, 2012).

At present time GeoSPARQL is a draft and there are few tools that allows spatial query on a RDF model and they are mainly based on proprietary standard.

The presented model stores the spatial information using the ontology defined in (OGC, 2012) in order to be compliant to GeoSPARQL definition and to be ready for its future implementation. The spatial operators are actually implemented with standard OGC protocols (i.e. WFS) using a geospatial web server relying on a spatial database.

A WFS request operates on a geo-layer that contains spatial and optionally descriptive data. A WFS request can query the field of a *geo-layer* using filters that could perform logical, descriptive and spatial operations. A WFS response is a *FeatureCollection*: a set of *FeatureMembers* that represents the *geo-layer* field.

As showed in section Part I1.2.2 there are several types of WFS requests and filters, it follows some definitions on the available operators and their WFS implementation. We distinguish two types of spatial operators: domain operators and spatial reference operators.

4.3.4.1 Spatial domain Operator

The spatial domain operators allow the navigation and querying of the hierarchical and stratified spatial domain. The operators that retrieve semantic information from the model, e.g. the *SpatialThematicContext* list or the *SpatialLayer* list, are fully implemented by SPARQL queries. For the operators that needs a spatial operation or which have to retrieve spatial information a WFS request has to be performed.

Hence the spatial model elements (*Place*, *PlaceInstance*, *Geometry*) are also stored in tables of a spatial DB such as PostgresSQL as shown in Fig. 40. In order to support the integration with the RDF Spatial model (Fig. 39) we store in the DB tables also the spatial domain elements URIs.

The normalized relational tables are then organized in a flat and aggregated view (*spatial_domain_view*) which is linked to a spatial web server. This view becomes a geo-layer that it is possible to query through the standard protocol WFS using a spatial web server (namely GeoServer).

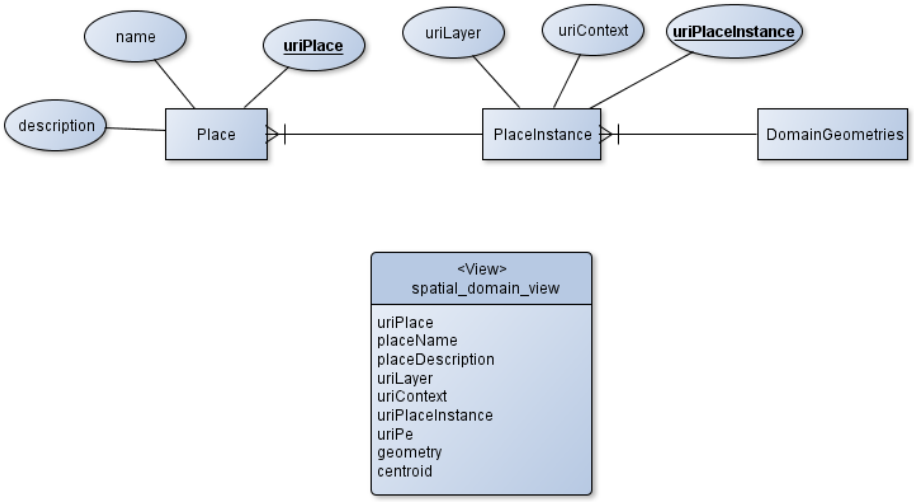

Fig. 40: Spatial Domain DB schema

Definition and implementation details of some of the spatial domain operators as follows.

Definition 4.3.1 getSpatialThematicContext(RDF graph)

This operator takes as input an RDF *graph* and retrieves a collection of *SpatialThematicContext* of the defined spatial domain.

Definition 4.3.2 getSpatialLayers(SpatialContext tc)

This operator takes as input *SpatialContext* and retrieves a collection of its *SpatialLayer* ordered by decreasing granularity.

The operators in Definition 4.3.1 and Definition 4.3.2 are fully implemented by SPARQL queries.

Definition 4.3.3 getSpatialLayerElements(SpatialLayer sl)

This operator takes as input the uri of a *SpatialLayer* and returns a collection of *PlaceInstances* belonging it.

Definition 4.3.3 implementation acts on the geo-layer *SpatialDomainGeoLayer* and performs the *getFeature* WFS request (Table 16) applying the filter propertiesEqualTo on the geo-layer field uriLayer.

Table 16: WFS Request retrieving Spatial Layer elements

```
<wfs:GetFeature service="WFS" version="1.1.0"  >
  <wfs:Query typeName="tadaima:spatial_domain_view">
   <ogc:Filter>
      <ogc:PropertyIsEqualTo>
        <ogc:PropertyName>layer</ogc:PropertyName>
        <ogc:Literal>$sl.uri</ogc:Literal>
      </ogc:PropertyIsEqualTo>
    </ogc:Filter>
    </wfs:Query>
</wfs:GetFeature>
```

Definition 4.3.4 getChilds(PlaceInstance pi)

This operator receive as input a *PlaceInstance pi* and retrieve the collection of the its child. A child of a *PlaceInstance pi* is a *PlaceInstance c* such that:

- c.geometry is **within** pi.geometry and
- c.layer is **finer than** pi.layer and **¬∃** l **ϵ** SpatialLayer such that l **is finer than** pi.layer and l **is greater** than c.layer.

This operator retrieves with a SPARQL query the *PlaceIstance* geometry and the *uri* of the child layer. Than it performs a WFS *getFeature* request (Table 17) that retrieves all the elements that are **within** the input geometry and that belongs to one of the child layer retrieved.

Table 17: getChild(placeInstance) WFS Request

```
<wfs:GetFeature service="WFS" version="1.1.0"  >
  <wfs:Query typeName="tadaima:spatial_domain_view">
   <ogc:Filter>
    <ogc:and>
      <ogc:Within>
         <ogc:PropertyName> geometry </ogc:PropertyName>
          $PI.geometry
      </ogc:Within>
           <ogc:PropertyIsEqualTo>
        <ogc:PropertyName>layer</ogc:PropertyName>
        <ogc:Literal>$childLayer.uri</ogc:Literal>
      </ogc:PropertyIsEqualTo>
    </ogc:and>
   </ogc:Filter>
  </wfs:Query>
</wfs:GetFeature>
```

4.3.4.2 Spatial Reference Operator Implementation

Those operators allow the retrieval of the *Feature* instances involved in a spatial property and that satisfies an absolute, semantic or uncertain spatial

constraint.

Therefore each feature instance having a spatial reference (*Semantic*, *Absolute and Uncertain*) is also stored in tables of the spatial DB such as shown in Fig. 41. In order to support the integration with the descriptive dimension of users declared *Features* modeled in RDF, we store in the DB tables the URI of the feature instance, the URI of the *Feature* class and di URI of the spatial property. Those URIs actually act as foreign keys between the relational database and the RDF data model.

As for the spatial domain element, the normalized relational tables are then organized in two flat and aggregated views, one for the quantitative spatial reference and the other one for the uncertain reference, which are linked to a spatial web server. Each view becomes a layer that it is possible to query through the standard protocol WFS.

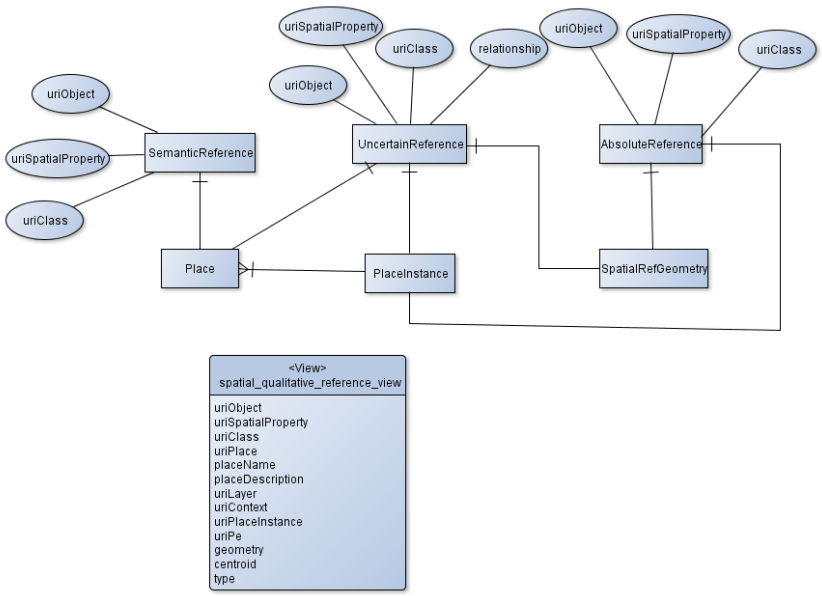

Fig. 41: Spatial Reference DB implementation

4.4 Spatio-Temporal Domain

In the real world applications a lot of spatio-temporal phenomena exist and there are many ways to classify them.

The first assumption made in this integration model is that a spatio-temporal phenomenon has separately both spatial and temporal properties and, moreover, must have a meaning in conjunction.

The second assumption is that we can distinguish among four types of spatio-temporal phenomena using a spatio-temporal classification, inspired by previous works ((Asproth et al., 1995), (Nadi & Mahmoud, 2003), (Kisilevich, 2005)) and depicted as follows (see also Fig. 42).

4.4.1 Spatio-Temporal Classification

The classification proposed considers the temporal dimension to be dynamic and based on two binary dimensions:

- *Spatial Position*: whether the object changes its spatial location or not over time.
- *Spatial Extension*: whether the object changes his shape (geometry) or not over time.

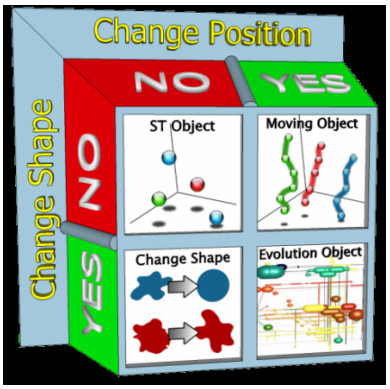

Fig. 42: Spatio-Temporal Classification

From this two dimensions four classes of spatio-temporal phenomena are obtained (Fig. 42):

- *ST-Event*: The objects belonging to this class change neither their position nor shape over time, but that could change their descriptive attribute.
- *Change Position* (or *Moving Object)*: the objects belonging to this class can change their position but not their shape over time, like a car during a trip.

- *Change Spatial Extension (or Change Shape)*: the objects belonging to this class can change their shape (geometry) but not their position, i.e. a building construction.
- *ST-Evolution*: this class covers the object changing their shape and position over time, like the migration of a flock of birds.

We believe that a model implementing this classification could cover all the spatio-temporal aspects of data in common applications, and could be used by all the types of spatio-temporal application described in (Pfoser & Tryfona, 1998).

4.4.2 Spatio-Temporal Reference

A feature involved in a Spatio-Temporal Phenomena has a Spatio Temporal reference. A Spatio-Temporal reference is identified by a quadruple:

```
<TRef, SPositionRef, SExtensionRef, DescriptiveRef>
```

where:

- *TemporalReference* (TR) is a pointer to the Period-Event id (let us call it idPE).
- *SpatialPositionReference* (SPR) can be a *SpatialReference* previously defined in the spatial model. It represents the spatial position of the object.
- *SpatialExtensionReference* (SER) can be or a Geometry in the OGC model (Open Geospatial Consortium, 2003) (Line, Polygon, MultiGeometry etc.). It represents the spatial shape of the object.
- *DescriptivePropertyReference* (DPR) is an optional component; it is a meta-data pointer to the descriptive property involved in the Spatio-Temporal phenomena. The descriptive property must belong to the *ObjectReference* on which the spatio-temporal reference relationship relies on.

The semantic of this quadruple depends on the type of ST phenomena. Depending on that, an object can have one or more *ST-Reference*, also *Position* and *Extension* reference could be mandatory or optional. The *DescriptionReference* is always optional. It follows the formal definitions of the specific Spatio-Temporal reference types.

Definition 4.4.1 ST-Event Reference

An ST-Event Reference is a *SpatioTemporal Reference* where *TemporalReference* is mandatory and at least one of Spatial Position Reference or Spatial Extension Reference is mandatory.

An object representing a ST-Event phenomenon has one and only one ST-Event Reference.

Definition 4.4.2 Change Position Reference.

A Change Position Reference is a SpatioTemporal Reference where Temporal Reference and SpatialPosition Reference are mandatory.

An Object representing a Change Position phenomenon has two or more Moving Object References ordered by time.

Definition 4.4.3 Change Spatial Shape Reference

A Change Spatial Shape Reference is a SpatioTemporal Reference where Temporal Reference and SpatialExtension Reference are mandatory.

An Object representing a Change Spatial Shape Phenomena has two or more Change Spatial Shape References ordered by time.

Definition 4.4.4 *ST-Evolution Reference.*

An ST-Evolution Reference is a SpatioTemporal Reference where TemporalReference, Spatial Position Reference and SpatialExtension Reference are mandatory.

An Object representing a ST-Evolution Phenomena has two or more ST-Evolution References ordered by time.

4.4.3 RDFS/OWL Model

To represent a Feature involved in some spatio-temporal phenomena there are essentially two approaches:

1. Define a Type/Class for each kind of ST Phenomena, describing their

special attribute and operator and force a Feature instance to belong to one of this defined Type/Class.

2. Define special ST property, with special domain, range and cardinality constraint, allowing a generic feature to declare a ST property.

In author opinion the second approach seems to suit better the triad model (Peuquet, 1994). In order to allow the definition of spatio-temporal properties assignable to feature, four RDF classes, defining the ST-Reference seen in previous paragraph have been created. As we will have seen in Definition 4.1.5 this classes defines the range of a spatio-temporal property assignable to a generic Feature (see Fig. 43).

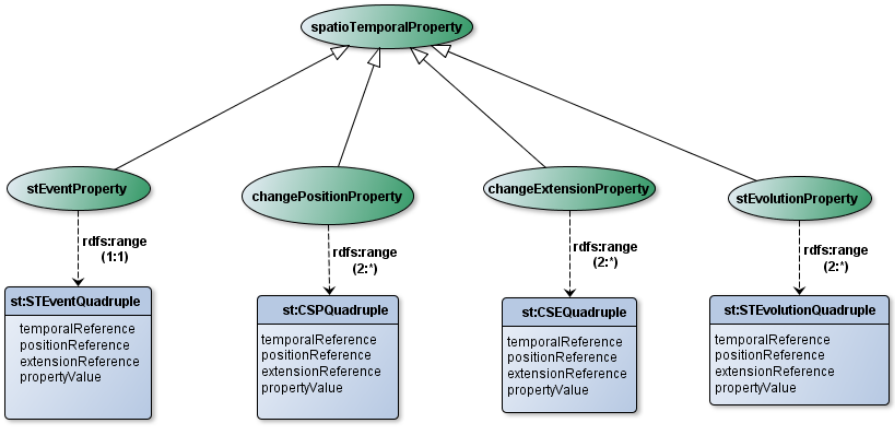

Fig. 43: Spatio-Temporal properties RDF schema

Details on the RDF model representing those spatio-temporal properties are depicted in Table 18 followed by the definition of the ST-reference quadruple classes in Table 19.

Table 18: SpatioTemporal Properties RDF

ST-EventReference	
Type	owl:objectProperty
SubPropertyOf	spatioTemporalProperty
Domain	CMSFeature
Range	ST-EventQuadruple
Constraints	minCardinality=1, maxCardinality=1
ChangePositionReference	
Type	owl:objectProperty

SubPropertyOf	spatioTemporalProperty
Domain	CMSFeature
Range	CSP-Quadruple
Constraints	minCardinality=2, maxCardinality=*
ChangeSpatialExtensionReference	
Type	owl:objectProperty
SubPropertyOf	spatioTemporalProperty
Domain	CMSFeature
Range	CSEQuadruple
Constraints	minCardinality=2, maxCardinality=*
ST-EvolutionReference	
Type	owl:objectProperty
SubPropertyOf	spatioTemporalProperty
Domain	CMSFeature
Range	STEvQuadruple
Constraints	minCardinality=2, maxCardinality=*

Table 19: Spatio Temporal Properties range (quadruple)

ST-EventQuadruple		
Type	Rdf:Class	
Properties	***STE-TemporalReference***	
	Type	owl:objectProperty
	Domain	ST-EventQuadruple
	Range	temporal:PeriodEvent
	Constraints	minCardinality=1, maxCardinality=1
	STE-SpatialPositionReference	
	Type	owl:objectProperty
	Domain	ST-EventQuadruple
	Range	spatial:AbsoluteReference
	Constraints	minCardinality=1, maxCardinality=1
	STE-SpatialExtensionReference	
	Type	owl:objectProperty
	Domain	ST-EventQuadruple
	Range	ogc:Geometry
	Constraints	minCardinality=0, maxCardinality=1

	STE-DescriptivePropertyValue	
	Type	owl:DateTypeProperty
	Domain	ST-EventQuadruple
	Constraints	minCardinality=0, maxCardinality=1
CSP-Quadruple		
Type	Rdf:Class	
Properties	***CSP-TemporalReference***	
	Type	owl:objectProperty
	Domain	CSP-Quadruple
	Range	temporal:PeriodEvent
	Constraints	minCardinality=1, maxCardinality=1
	CSP-SpatialPositionReference	
	Type	owl:objectProperty
	Domain	CSP-Quadruple
	Range	spatial:AbsoluteReference
	Constraints	minCardinality=1, maxCardinality=1
	CSP-SpatialExtensionReference	
	Type	owl:objectProperty
	Domain	CSP-Quadruple
	Range	ogc:Geometry
	Constraints	minCardinality=0, maxCardinality=1
	CSP-DescriptivePropertyReference	
	Type	owl:DateTypeProperty
	Domain	CSP-Quadruple
	Constraints	minCardinality=0, maxCardinality=1
CSE-Quadruple		
Type	Rdf:Class	
Properties	***CSE-TemporalReference***	
	Type	owl:objectProperty
	Domain	CSE-Quadruple
	Range	temporal:PeriodEvent
	Constraints	minCardinality=1, maxCardinality=1
	CSE-SpatialPositionReference	
	Type	owl:objectProperty
	Domain	CSE-Quadruple

	Range	spatial:AbsoluteReference
	Constraints	minCardinality=0, maxCardinality=1
		CSE-SpatialExtensionReference
	Type	owl:objectProperty
	Domain	CSE-Quadruple
	Range	ogc:Geometry
	Constraints	minCardinality=1, maxCardinality=1
		CSE-DescriptivePropertyValue
	Type	owl:DatatypeProperty
	Domain	CSE-Quadruple
	Constraints	minCardinality=0, maxCardinality=1
STEv-Quadruple		
Type	Rdf:Class	
Properties		***STEv -TemporalReference***
	Type	owl:objectProperty
	Domain	STEv-Quadruple
	Range	temporal:PeriodEvent
	Constraints	minCardinality=1, maxCardinality=1
		STEv -SpatialPositionReference
	Type	owl:objectProperty
	Domain	STEv-Quadruple
	Range	spatial:AbsoluteReference
	Constraints	minCardinality=1, maxCardinality=1
		STEv -SpatialExtensionReference
	Type	owl:objectProperty
	Domain	STEv-Quadruple
	Range	ogc:Geometry
	Constraints	minCardinality=1, maxCardinality=1
		STEv -DescriptivePropertyValue
	Type	owl:DatatypeProperty
	Domain	STEv-Quadruple
	Constraints	minCardinality=0, maxCardinality=1

4.4.4 Spatio-Temporal Operator Implementation

The spatio-temporal operators allow the retrieval of the feature involved in a spatio-temporal phenomenon.

It follows definition of the main spatio-temporal operator managed in the proposed model.

Definition 4.4.5 getEvents(rdf:property st)

This operator receives as input an *rdf:property* that is sub-property of *stEventProperty*. It returns a collection of *ST-EventsReference* quadruples (Definition 4.4.1) involved in the ST-EventProperty *st* given as input.

Definition 4.4.6 getMovements(CMSFeature object, rdf:property cp)

This operator receives as input an instance of a user defined *CMSFeature* (*object*) and an *rdf:property* that is sub-property of c*hangePositionProperty (cp)* where cp.domain=object.getClass*.* It returns a collection of *Change Posistion Reference* quadruples (Definition 4.4.2) ordered by *TemporalReference*, involved in the ChangePositionProperty *cp* given as input.

Definition 4.4.7 getExtensionChanges(CMSFeature object, rdf:property ce)

This operator receives as input an instance of a user defined *CMSFeature* (*object*) and an *rdf:property* that is sub-property of *changeExtensionProperty (ce)* where cp.domain=object.getClass*.* It returns a collection of *Change Spatial Extension Reference* quadruples (Definition 4.4.3) ordered by *TemporalReference*, involved in the c*hangeExtensionProperty ce* given as input.

Definition 4.4.8 getEvolutions(CMSFeature object, rdf:property ev)

This operator receives as input an instance of a user defined *CMSFeature* (*object*) and an *rdf:property* that is sub-property of *stEvolutionProperty (ev)* where cp.domain=object.getClass*.* It returns a collection of *ST-Evolution Reference* quadruples (Definition 4.4.4) ordered by *TemporalReference*, involved in the c*hangeExtensionProperty ev* given as input.

Some of the defined spatio-temporal operators need a temporal ordering on the results, requiring their integration with the Temporal Interval Tree.

Furthermore the above defined operators can be combined with spatial filters; e.g. user wants to retrieve *ST-Event* in a selected area or bounding-box. To implement this operation a WFS has to be performed, so each feature instance having a spatio-temporal property is also stored in tables of the spatial DB (see Fig. 44). The defined tables reflect the *rdf:classes* implementing the spatio temporal quadruple reference. In order to support the integration with the descriptive dimension of users declared *Features* modeled in RDF, we store in the DB tables the URI of the feature instance, the URI of the *Feature* class and di URI of the spatial property as seen for the spatial-properties. The normalized DB is organized in four flat views (*st_event_view, change_position_view, change_extension_view* and *st_evolution_view*) that will feed four geo-layers in order to allow the implementation of the spatio-temporal operators with spatial filters through a WFS request.

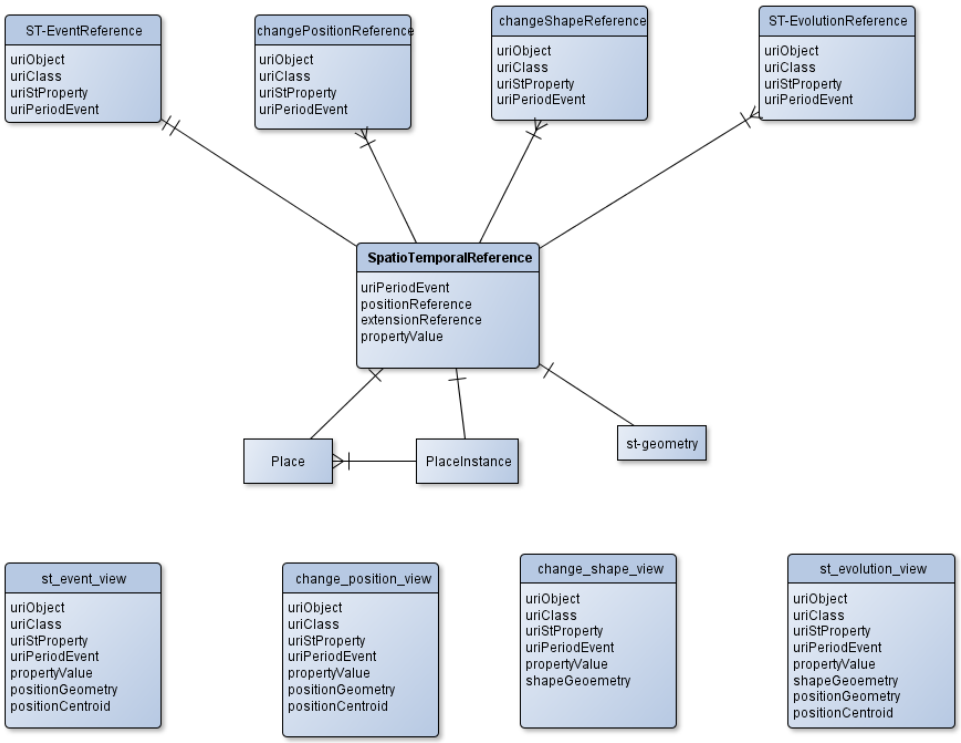

Fig. 44: Spatio-Tempora Reference DB schema

Chapter 5 CMS Prototype: Tadaima.

Building a web oriented spatio-temporal application requires to deal with a lot of technologies and tools that requires high technical skills that common users, like for example the cultural heritage experts, rarely have. The complex model underlying an ad hoc spatio–temporal built system makes the data entry and the personalization of a data exploration interface technically involved. This is a recurrent problem in computer science, let us consider for example the development and growth of blogging activities where experts in different fields can diffuse their advices and opinions in the web without taking care of the technical aspects about databases and web technologies just using adequate tools like Web Content Management System (for example, Wordpress, Joomla etc.).

In Cultural heritage field this problem is relevant, it is very important to store and publish data to improve and to spread knowledge, but these tasks are not easy due to the complex nature of data. The idea presented in this thesis is to provide to user, and in particular for this prototype to the Cultural heritage expert, a Content Management System that has a twofold goal:

1. to allow the common users (i.e. cultural heritage experts) to build from scratch a web application in order to store and to present spatio-temporal data;
2. to offer to end users an interaction interface that allows to independently interact with the three dimensions of data: spatial, temporal and descriptive.

As for the most number of CMSs, two classes of users are defined: the *Admin-User* that models data, inserts and updates them and chooses the visualization metaphor, and the *End-User*, that visualizes and explores the available data.

In the following subsection the back-end and front-end interfaces and operations are presented as they are applied to a significant case study in the Cultural Heritage field.

5.1 Back-End

The *Admin-Users* interact with the back-end interface of the CMS. This interface allows them to create a new data schema by defining features and their properties, inserting and editing data and to personalize the way End-Users visualize them. The operations allowed to the *Admin-User* are depicted in the following sections.

5.1.1 Configuration

This section allows to the *Admin-User* to set the access parameters for the underlying engines like the relational database (PostgreSQL), the spatial database (PostGIS, (PostGis, 2011)), the spatial web server (GeoServer, (OSGeo, 2011)) etc., and to add other users and their access privileges.

The fresh installation of *Tadaima CMS* requires the settings of the access parameters to PostgreSQL and GeoServer, and automatically creates the tables, the views and the geo-layers needed by the application in order to be executed (see Fig. 45).

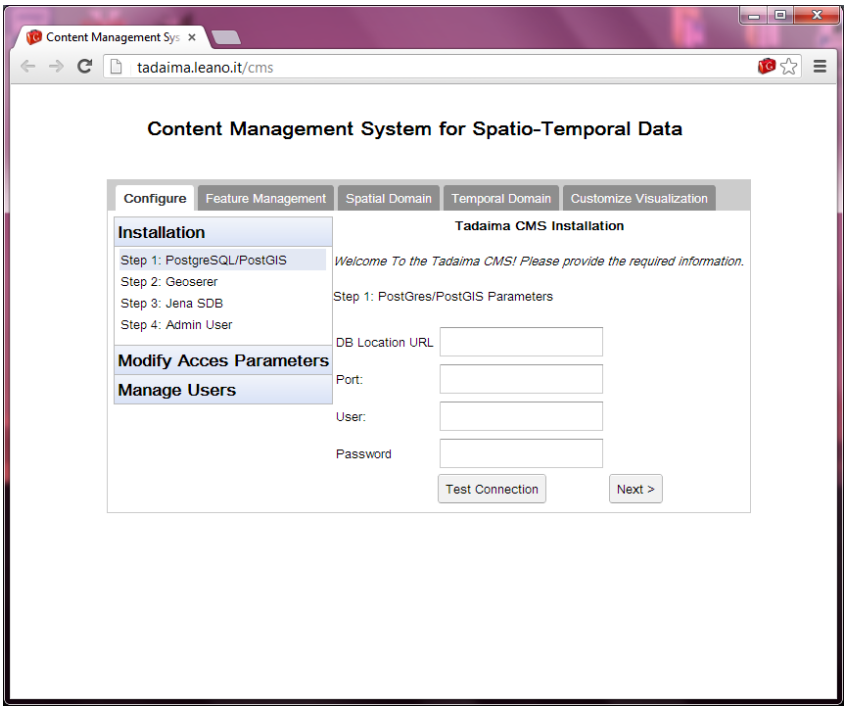

Fig. 45: CMS Installation Steps

5.1.2 Feature Management

In this panel the *Admin* can define her/his data model, adding and editing the objects instance of the defined *Features*.

5.1.2.1 Features Creation

The features creation section has the responsibility of creating an *Application-Schema* that suits the needs of the *Admin-User* and that is compliant with the CMS meta-model. *Admin-Users* can create new kind of *Features* alternatively by using a guided interface as illustrated in Fig. 46, by importing a RDF/OWL file or by selecting and configuring some pre-defined class of *Feature* available in the CMS library (populated with classes commonly used in Cultural Heritage applications).

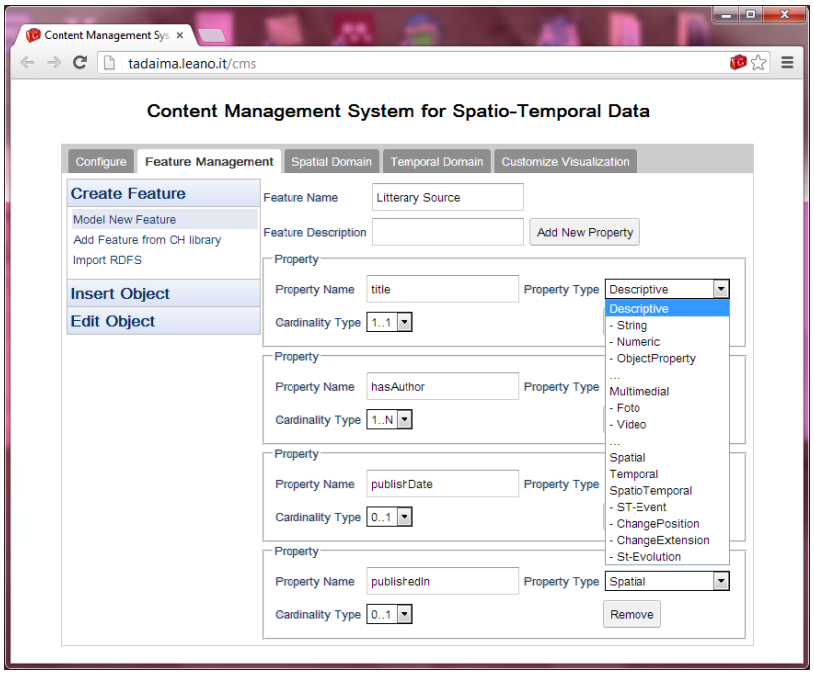

Fig. 46: Feature Creation: Model Designer User Interface

The architecture of this section is showed in Fig. 47. A user defined *Feature* is a *subclass* of *CMSFeature* belonging to the meta-model. Users can add as many properties as they need. The *Model-Designer* component allows the user to set name of a property, its type (choosing from those defined in the meta-model) and its cardinality constraint. The defined property becomes a *subproperty* of the ones defined in the *meta-model* and, if selected, the *owl:mincardinality* and

owl:maxcardinality constraints are instantiated. The *Meta-Model Manager* component takes as input the choice made by the user through the *Model-Designer* and the meta-model and produces as output the (RDFS) *Application-Schema*.

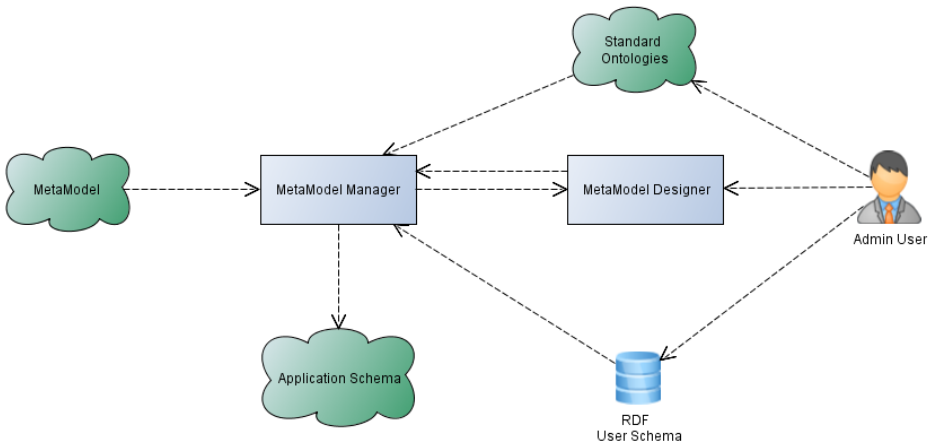

Fig. 47: Feature Creation Diagram

As an example, consider Fig. 46 where the *Admin-User* models the *Litterary Source* feature. She/he defines a *Feature* having a string property "*title*", an object property "*author*", a spatial property "*published in*" and a temporal property "*publish date*". Table 9 shows the *Application-Schema* generated for this feature.

Admin-User could also import a pre-defined OWL or RDFS file, the *Model-Manager* will try to map the inserted schema into one compatible with *the meta-model* (for example mapping each *rdfs:Class*/*owl:Class* in a *CMSFeature*) and asks to the *Admin-User* to make corrections.

Tadaima CMS also offers a set of predefined *Features* and their properties able to represent some well-known Cultural Heritage artifacts, like *Literary Source*, *Bibliographic Reference*, etc.

5.1.2.2 Object Creation-Editing

This admin interface panel allows users to insert and edit instances of the feature created as described in the section 5.1.1 . As showed in Fig. 48 the *Application-Schema Manager* uses the *Application-Schema* to automatically generate the input interfaces for each kind of user defined feature.

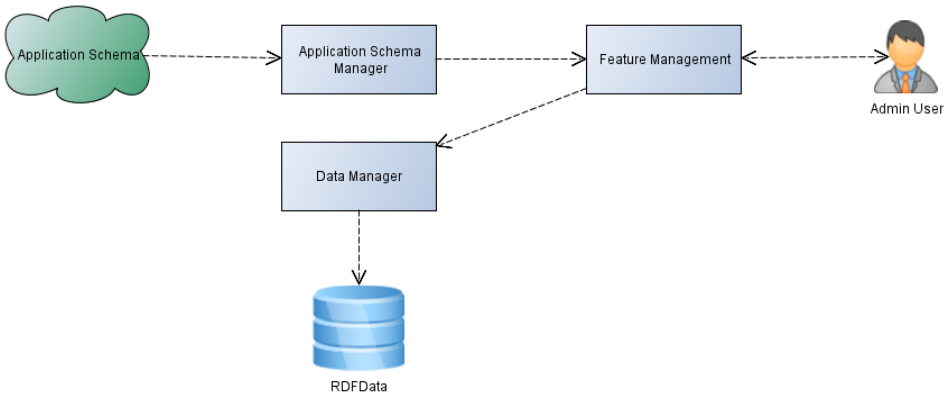

Fig. 48: Feature Editing Diagram

In Fig. 49 it is showed the inserting interface for the *Literary Source* feature described in Table 9. This component also manages the storage of the data inserted/edited by users.

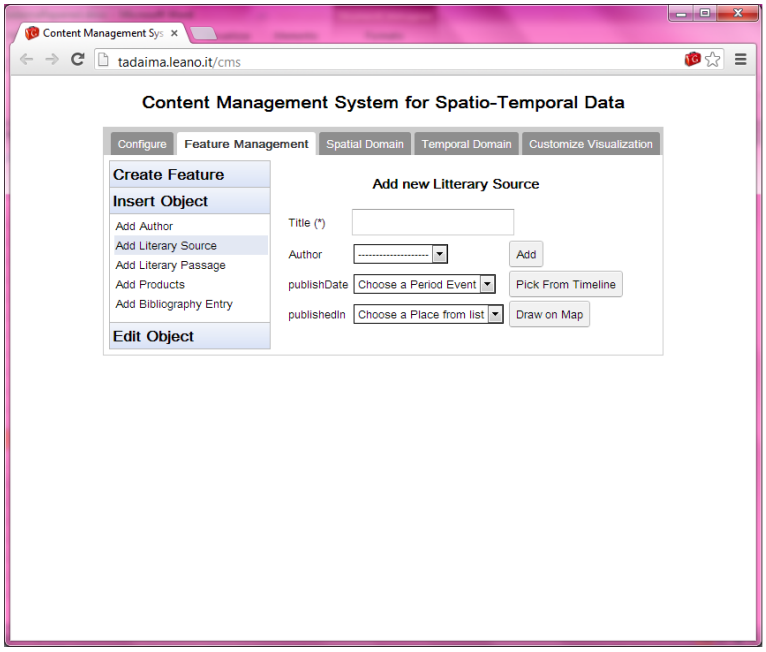

Fig. 49: Auto-Generate Interface for inserting-editign LiterarySource feature

5.1.3 Temporal Domain Management

In this panel the *Admin-User* can define and populate the Temporal Domain elements. As showed in Fig. 50 she/he can create *ThematicContext* and *Temporal-Layer*.

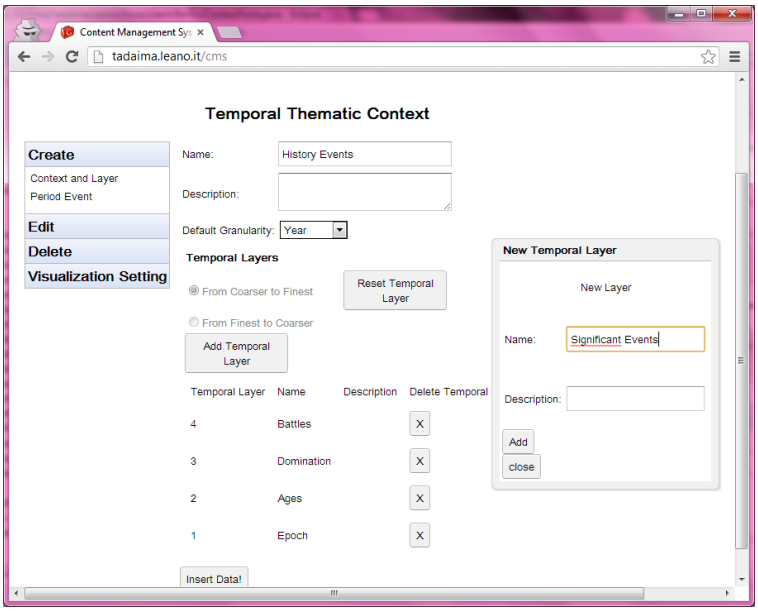

Fig. 50: Temporal Thematic Context and Layers Creation

This section also allow the creation of *Quantitative* and *Qualitative Period-Events* as showed in Fig. 51.

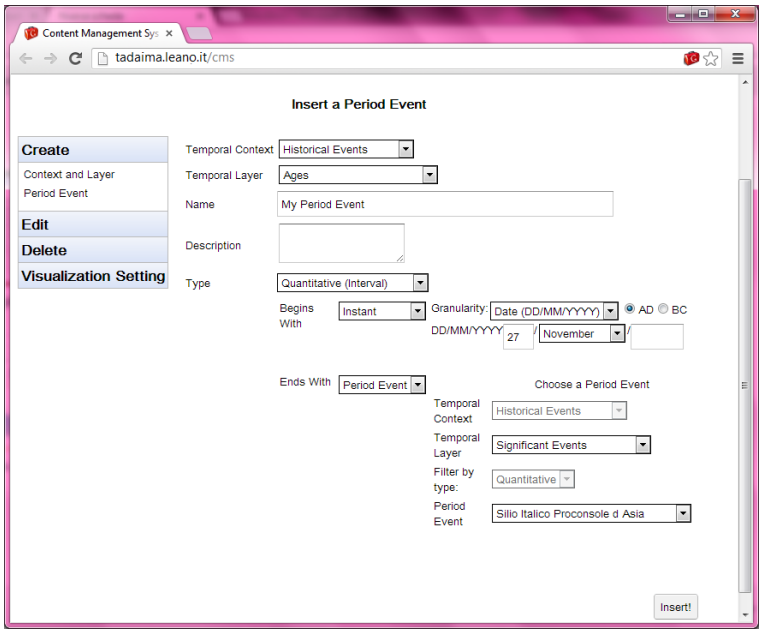

Fig. 51: Insert Quantitative Period-event

Admin-User can also personalize some timeline visualization details (see section 5.2.2.1), like the order of *Thematic-Context*, the default one and the color of the *Temporal-Layer*.

5.1.4 Spatial Domain Management

In this panel, the *Admin-User* can define the Spatial Domain elements and populate those possibly exploiting graphical interactions with a map. As for the temporal domain, she/he can create *Thematic-Context* and Spatial-*Layer,* also she/he can create (semantic) *Places*.

Admin-User can also associate geometries to *Places* (e.g. creating *Place-Instances*) by interacting with a map or by importing GML or KML files (see Fig. 52).

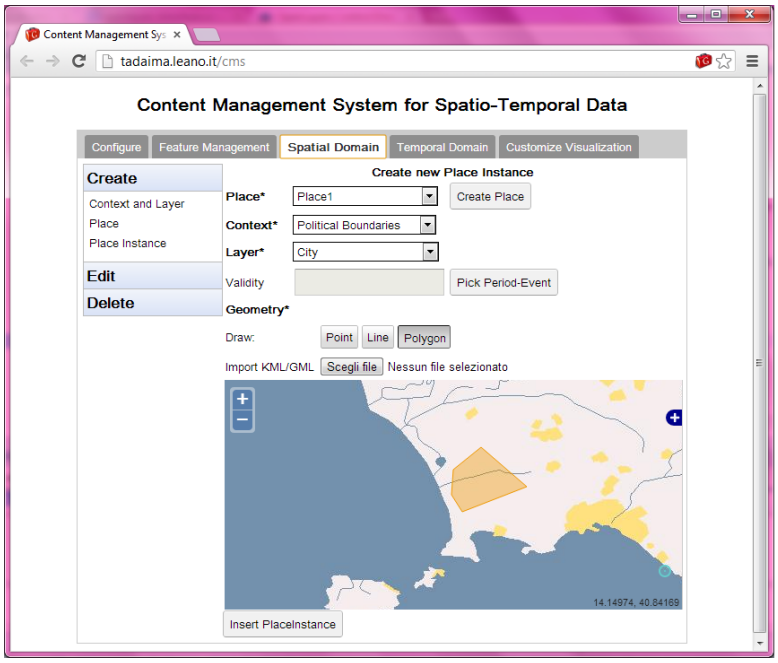

Fig. 52: Place Instance creation interface

The data inserted by the user are stored in RDF and, for the spatial part, in the spatial DB (PostGIS) in order to be available for querying them by the *Spatial Web Service* (GeoServer).

5.1.5 Visualization Personalization

In this panel the *Admin-User* can associate visual metaphor to the defined features in order to control their visualization in the front end.

For each spatio-temporal data type, several visualization metaphors have been proposed in the literature. We choose a set of spatio-temporal metaphor for

each kind of Spatio-Temporal phenomena inspired by (Andrienko, 2003) where it is suggested the appropriate visualization according to the spatial nature of data.

For example for the *Features* representing an *ST-event* phenomenon (see 4.4.1) users can choose to visualize data according the "*Cluster*" visual metaphor, where objects are grouped together in the same cluster (represented by a circular icon on the map) depending on their spatial distance, the radio icon is directly proportional to the cluster size. User can personalize several aspect of the cluster visualization, e.g. the color of the icon, the criteria of the cluster strategy, the properties that could act as filters, etc.

From the architectural point of view, the *Visualization Personalization Manager* allows *Admin-Users* to choose among the CMS defined collection of spatio-temporal metaphor to visualize their data. As showed in Fig. 53 the component takes as input the *Application-Schema* and automatically suggests the metaphor for a *Feature* basing on the nature of its defined property. This component produces as output a *Visualization Personalization Schema* used by the front-end to visualize data to end users.

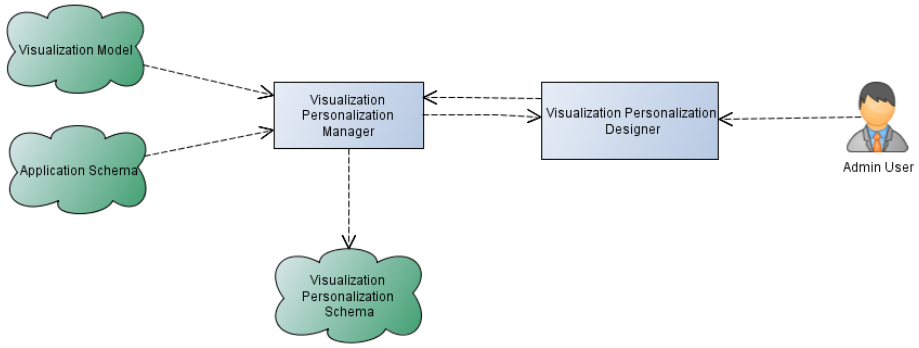

Fig. 53: Visualization Personalization Diagram

5.2 Front-End

End-Users interact with the CMS Front-End component. The main goal of this component is to present spatial, temporal, textual and multimedia information merged together and allowing users to easily interact with them. In the next subsections more details about the front-end architecture and the visualization strategy will be depicted.

5.2.1 Front-End Architecture

Our main goal is to realize a flexible architecture, where each module is characterized by a loose coupling, in order to achieve the independence from data storage and visualization tools.

In particular, the proposed architecture relies on three-tier architecture, composed by a data storage at the back end, a business logic layer, and a visualization layer at the front end, arranged as shown in Fig. 54. As suggested by the OGC, to achieve modularization all the communication among or intra modules are carried out through standard open protocols and exchanging data formats. The CMS architecture extends our work presented in (Cutugno et al., 2012). The three tiers of the proposed architecture are detailed in the following subsections.

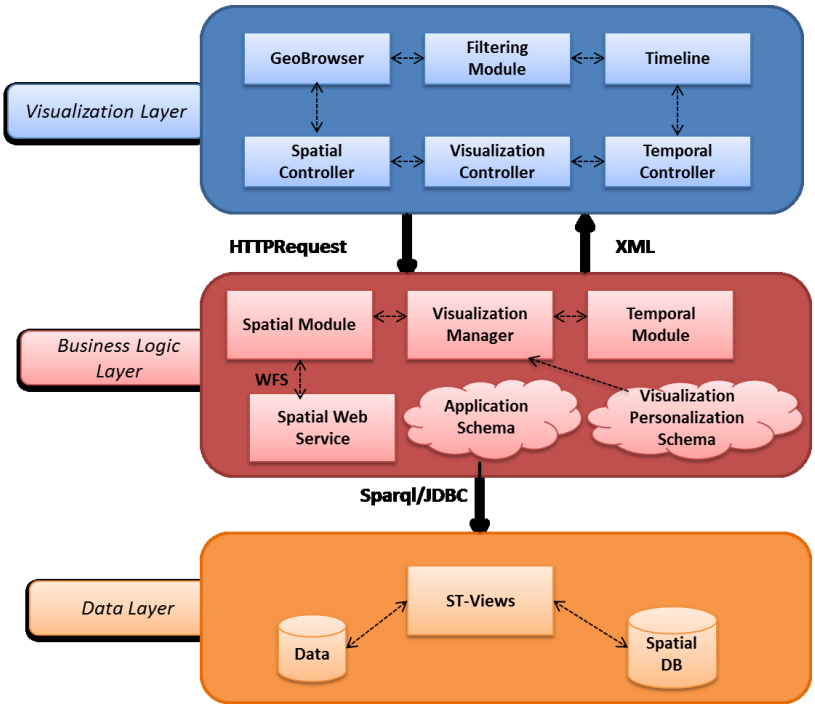

Fig. 54: Front-End Architecture

5.2.1.1 Data Layer

This component is responsible of storing and making persistent the application data. It is composed by a RDF-triple store (namely Jena SDB) that make persistent the RDF data generated by the application; by a standard DMBS

and a Spatial DBMS (in our case PostgreSQL and PostGIS) implementing the DB view of the model described in section 0. This layer communicates with the upper one by means of standard protocols like SPARQL and JDBC.

5.2.1.2 Business Logic Layer

The layer is responsible for performing the operations required by the user and integrating data belonging to the heterogeneous data sources in a single data structure, to be fed to the *Visualization Layer*. This tier is composed of three main modules (*SpatialModule, TimelineModule* and V*isualizationManager)*, make use of OGC-compliant web-services for spatial operation (*SpatialWebService*) and is responsible to create and query data respecting the CMS data-model.

- *VisualizationManager: t*his component is responsible for interpreting and managing the operations received by the *Visualization Tier.* It dispatches the spatial and temporal request to the respective module and merges the results, completing them with the descriptive information produced by the heterogeneous SPARQL (Prud'hommeaux & Seaborne, 2008) queries that feeds the upper layer.
- *SpatialModule*. This module performs the spatial query requested by the user invoking a *SpatialWebService* through the standard protocol (WFS).
- *TimelineModule*. This component implements all the temporal operator defined in (Allen, 1983). It is responsible of performing the temporal query requested using the Temporal Interval Tree (see section 4.2.5.2) and returns the results to the *VisualizationManager*.
- *SpatialWebService*. This component implements the OCG compliant web services (WFS, WCS, WMS), and it works with standard protocol like HTTP, SOAP etc. We adopt Geoserver (OSGeo, 2011) as our *SpatialWebService*.

5.2.1.3 Visualization Layer

The *Visualization Layer* (VL) applies the choices made by the admin user in the *Visualization Personalization* step. It takes as input the *Application-Schema*, the

Visualization Personalization Schema and the *data* and produces as output the visualization metaphor. It also implement the required interaction and synchronization among the different component of the interface.

The main goal of this layer is to present spatial, temporal, textual and multimedia information merged together and offered to the users, furthermore we aim at notifying the underlying layers with the query they performed. *VL* uses a *Geobrowser* and a *Timeline* visual component in order to allow the spatial and temporal exploration and to make them independent. The descriptive data dimension is represented on the *Geobrowser* or/and in the *descriptive panel* (a more detailed description of the interface will be presented in the section 5.2.2).

This Layer is composed by four components:

1. *VisualizationController (VC).* It is responsible of handling events on the user interface and of notifying the *Business Logic Tier* about the operation the user wants to do. The extra-tier communication uses the HTTP/XML protocol, the intra-tier one uses an XML data format: as a response the *VC* receives XML data to refresh its visual components.
2. *SpatialController*. It is able both to render user selected information onto a map and to interpret and notify the spatial query to the *VC*.
3. *TimelineController*. The component implements the visual metaphor that allows an easily navigation and interaction with the hierarchal and stratified time dimension.
4. *FilteringModule*. The module is responsible of applying the descriptive filters given by the user, to notify the user operations to the underlying layer and to display the descriptive dimension of data.

5.2.1.4 Exchange Protocol

The architecture here proposed uses standard files and protocols (WFS, GML) for the communication intra and extra module, in order to be independent from storage and visualization tools.

The user interaction is handled by the *Visualization Layer (VL)*, it interprets

and intercepts the user query and notifies the *Business Logic Layer (BL)* performing an *HTTPRequest*. At this point the *BL* switches the request to the specific component. In particular, the spatial manager invokes a *Spatial Web Service* using the WMS standard protocol receiving a KML file as response. The *BL* uses *meta-model* and *Application Schema* information and invokes the data layer too in order to retrieve the descriptive data information from the triple-store using SPARQL queries. This response is merged with the KML from the spatial component and with the response from the temporal component building a new XML file. This XML is given as response to the *VL*. Filtering operations on the descriptive dimension are made using XQuery (W3C, 2010) on the merged XML file. An example of interaction is shown in Fig. 55.

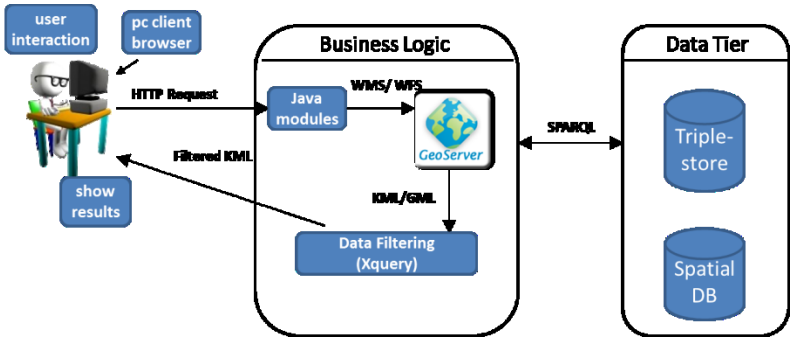

Fig. 55: Exchange Protocol

5.2.2 Front-End Interface

The Front-End Interface extends our work presented in (Cerasuolo et al., 2012), making it fully configurable by users through the *Visualization Personalization Designer*. Its design was inspired by two basic principles:

1. *The triad model* (Peuquet, 1994): an object can be seen under three dimensions: spatial, temporal and descriptive. Both the underlying model and the visualization metaphor reflect the threefold nature of data.
2. *Shneiderman's visual information seeking mantra* (Shneiderman, 1996); it is an (iterative) three step approach to data searching: overview first, zoom and/or filter, and details on demands.

The proposed interface offers an overview of the data in a spatial domain and gives the ability to users to filter them by temporal queries using the timeline, or by applying filter on the descriptive dimension. Requested details of a selected object are showed in a separate panel. This interface allows users to navigate among the three dimensions: spatial, temporal and descriptive. The components are fully synchronized and they use standard protocols and data format to communicate.

As showed by the mockup in Fig. 56, the developed user interface is composed of the four basic components, integrated in a web page and depicted in the following subsections.

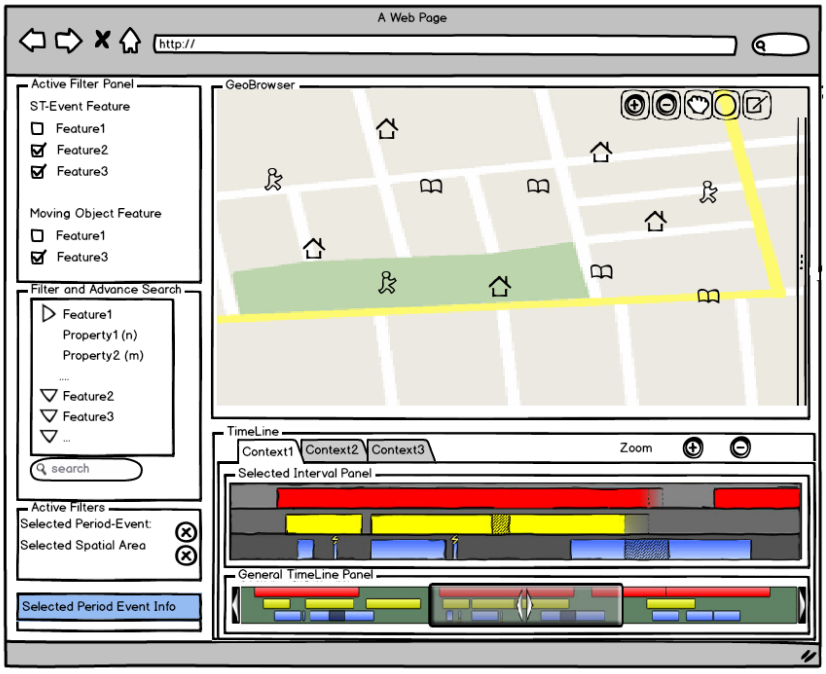

Fig. 56: Front-End interface mock-up

5.2.2.1 Timeline: Temporal Domain Visualization

The timeline is the distinctive element of the proposed interface, allowing users to interact with the temporal. The temporal model defined in section 4.2 introduces some original features, allowing users to navigate information by thematic *Context*, granularities *Layers* and *Period-Events*. Therefore, an ad hoc temporal visualization metaphor is required.

The proposed *Temporal Visualization Component* (TVC) provides scalable temporal information, allowing users to dynamically choose the temporal detail level that best matches their search criteria. Through the TVC it is possible to operate a targeted selection of temporal data, reducing the information overload by filtering data onto the temporal dimension.

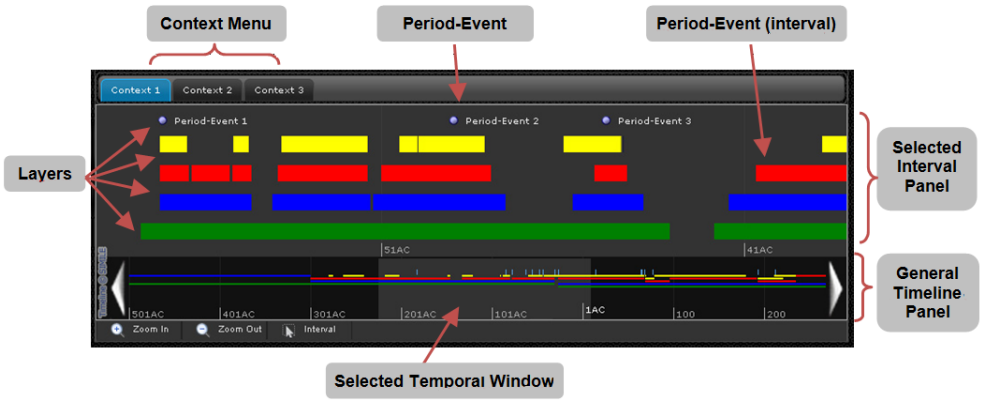

Fig. 57: Timeline Mock-Up

The proposed temporal visualization metaphor is shown in Fig. 57. The *Context Panel* has a selectable tab for each defined thematic temporal context: by choosing one of them, only the corresponding temporal data (layers and period-events) are retrieved.

The temporal navigation interface is structured in two panels (described in the following subsections):

1. the *General TimeLine Panel*: it offers a global temporal overview and allows users to select a temporal interval of interest;
2. the *Selected Interval Panel*: it gives a detailed view of the temporal interval selected into the General Time Panel.

General TimeLine Panel

The aim of this panel is to offer the same functionality of an *Overview Map*, i.e. to show the location of the current view respect to a wider and more general context. Typically this interaction metaphor is used in bi-dimensional spaces, such as maps or images, to give an immediate insight of the rendered portion of space, given a wider domain. The *General TimeLine Panel* has the same goal, in the

temporal domain, which is mono-dimensional. We choose a representation based on a scrollbar-like interaction (see Fig. 58), where the whole time span is displayed, and a semitransparent selector is used to indicate the temporal frame currently considered.

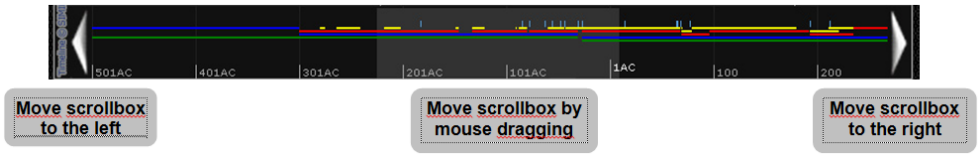

Fig. 58: Timeline General Panel Actions

This timeline should be the main temporal selector for users, allowing them to obtain a fast temporal navigation. Besides the usual interactions, other than the typical ones (moving the scrollbox and clicking the arrow buttons), we have envisioned also the possibility to directly resize the scrollbox itself, to change the covered time span. This can be achieved by dragging the borders of the scroll box, till the desired size.

Selected Interval Panel

The *Selected Interval Panel* offers a detailed view of the temporal span selected into the *General TimeLine Panel*. This panel has a twofold goal: the first one is to display all temporal information available for the selected time span, and the second one is to allow users to select only some temporal elements of interest (i.e. period-events, intervals or specific instants), in order to further filter the data shown into the geobrowser.

The panel has a bar for each *Layer* belonging to the chosen *context* and shows the *Period-Events* distribution over the selected temporal window. *Period-Events* are differently shaped based on their temporal extension (see Fig. 59):

- the temporal interval ones are represented by rectangles having a width directly proportional to the temporal length;
- the temporal instant ones are represented by blue circles.
- If *a Period-Event* is (recursively) related to a *Qualitative* one, its name is highlighted by a different character type and after its name there

are the number, between parenthesis, of *Qualitative Period-Event* related to it. By clicking on the number a balloon shows the list of the *Qualitative Period-Event* it is related.

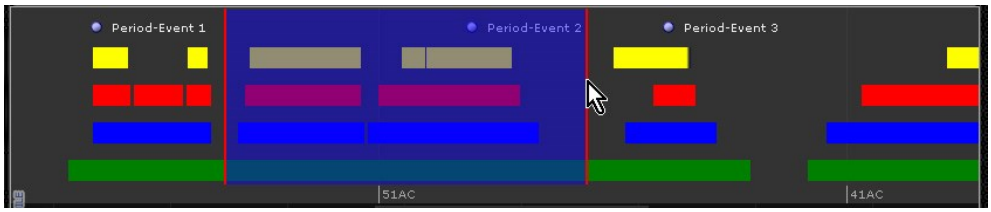

Fig. 59: Timeline Selection Panel: Interval selection

Users can interact with this panel by selecting a *Period-Event* just clicking on it, or by selecting an interval of interest, drawing a box onto the time span of interest (see Fig. 59). The action of selecting a span causes a complex query: this retrieves not only the data that is strictly related to the selected span, but also all the data that are related to the span by overlapping and inclusion relationship over *Period-Events*.

5.2.2.2 Spatial Component

The *Geobrowser* and the *Spatial Query Panel* represent the *spatial visualization component*.

The Geobrowser is responsible of rendering data on a map, allowing users to explore the spatial dimension. Users can navigate through the map by selecting areas of interest and activating the (spatial) thematic layers existing into the dataset.

The *Spatial Query Panel* is responsible of interacting with the geobrowser in order to allow users to perform spatial queries (i.e. visualizing objects nearest to a point in a circle area defined by a chosen radius) or by drawing the interested area on the map.

5.2.2.3 Descriptive Filters and Visualization

Two panels represent the descriptive visual component. The first one (Fig. 60a) is the *Active Filters Panel*: it helps users to focus on the descriptive dimension by selecting the information layer to be visualized on the map. They can choose

the type of the object to visualize and they can filter them by some feature, chosen by the *Admin-User* in the *Visualization Personalization* step, by activating and disabling layers, allowing an easily data crossing. It allows also users querying data by textual search and to remove selected spatial and temporal filter. Activating a filter on this panel automatically updates the spatial component (map) according to the selected temporal filter.

The second one (Fig. 60b) is the *Detail Panel* it is showed on demand by clicking on an object on the map; it contains detailed information on the object selected and, if present, it shows multimedia content (video, photos etc.). The *Admin-User* can choose what details to show and how in the *Visualization Personalization* step.

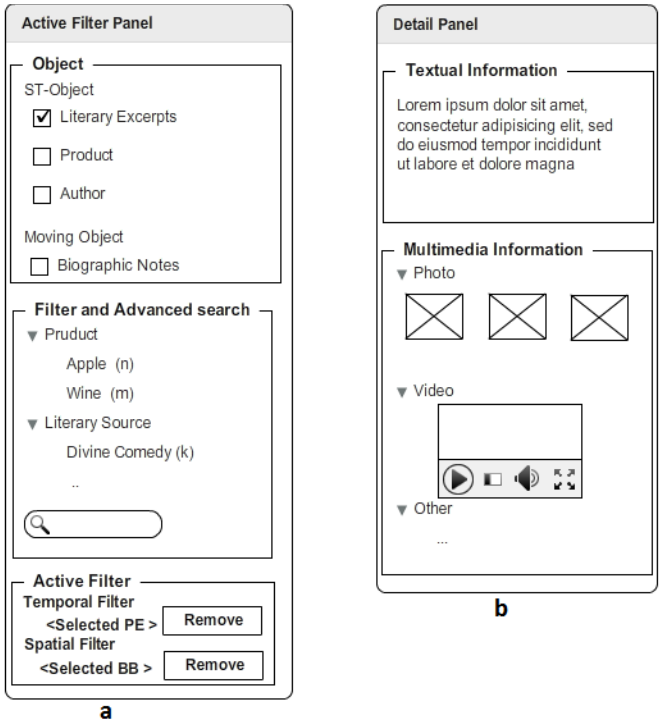

Fig. 60: Descriptive Component Mockup: a) Active layers, b) Detail Panel

Chapter 6 A Case of Study in Cultural Heritage field.

The topic of spatio-temporal visualization and exploration is assuming more and more relevance in the field of Cultural Heritage. The possibility of visualizing the spatial information of ancient artifacts on a map, or the ability of immediately recognizing the temporal relationship between events can improve the user knowledge discovery process.

The case study presented in this paper documents a joint work with Latin philologists of the department "Filologia Classica F. Arnaldi" and describes a deliverable of the project named TRACCIA supported by “FARO – Università degli Studi di Napoli Federico II – Polo delle Scienze Umane e Sociali”. The aim of the project was to find, document and give public access to the literary and historical evidences of typical agricultural *Products* in the area of *Campi Flegrei* in the neighborhoods of Naples a rich archeological site located nearby *Naples* (*Italy*) needing promotion in the spirit of increasing the international attention on Italian Cultural Heritage Patrimony. The documented literary evidences are a collection of Latin and Greek excerpts from classic authors (such a collection can be found in (Valenti, 2011)).

In particular, the objects carried out by this project are Latin and Greek *Literary Excerpts* related to some agricultural *Product* located in some *Places* in the *Campi Flegrei* area at a given time (*Period-Event*). *Literary Excerpts* belong to some *Literary Source* written by classic *Authors*. *Literary Sources* can be linked to some *Bibliographic References*. Changes in space and time during the life of an *Author* are managed through the entity *Biographic Notes*. In Fig. 61 a (abstract) class diagram of the case study under consideration is depicted.

The deliverable of TRACCIA Project was a web application, available at http://faro.dsf.unina.it based on our architecture and visualization framework presented in (Cutugno et al., 2012) and (Cerasuolo et al., 2012) and extended by the CMS presented in this thesis.

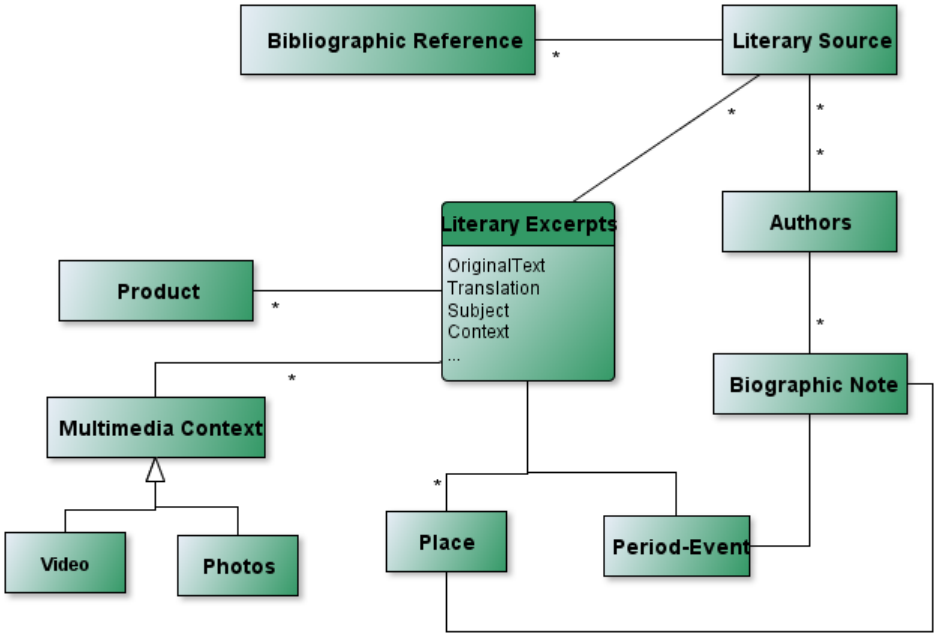

Fig. 61: Traccia Abstract Class Diagram

The proposed CMS allows the modelling and the representation of *Products*, *Literary Passage, Literary Source* and *Authors* in a common framework enhancing their temporal and spatial dimensions and allows the users to query data following the temporal and spatial dimensions. The application itself also manages the *Biographic Notes* on *Authors.*

The data generated by the framework case of study have been ported in the CMS in two steps:

1. The definition of the *Application Schema* using the *Meta-Model Generator*;
2. The porting of the relational DBMS on which TRACCIA data relies on in an RDF format compatible with the *Application Schema*. To afford this step we used RDF mapping tools well known in the Semantic Web world, like D2RQ (Bizer, 2004).

6.1 Spatio –Temporal Visualization

In particular in the considered case of study two *Features* involved in spatio-temporal phenomena are used: *Literary Passage*, representing portion of text extracted from some *Literary Source* citing agricultural products in a *Place* and in a defined *Period-Event*; *Authors*, having *biographic notes* describing their movements in space and time.

The feature *Literary Passage* includes the spatio-temporal property *citation* that is a subclass of *ST-Event property*. The *Visualization Personalization Component* automatically suggests for this kind of feature a set of properties compliant with an *ST-Event* phenomena. For this case study the metaphor *Cluster* was chosen: objects are grouped together in the same cluster (represented by a circular icon) depending on their spatial distance, the radio icon is directly proportional to the cluster size. Clicking on the icon a balloon will be showed containing a list of all the objects in the cluster. This list of objects is clickable: users can visualize details on the selected object in the *Detail Panel*. Admin-User can customize this metaphor by choosing how to represents the features in the balloon and in the *Detail Panel* and what are the applicable research filters using the feature property. The result of the application of the considered metaphor is shown in Fig. 62.

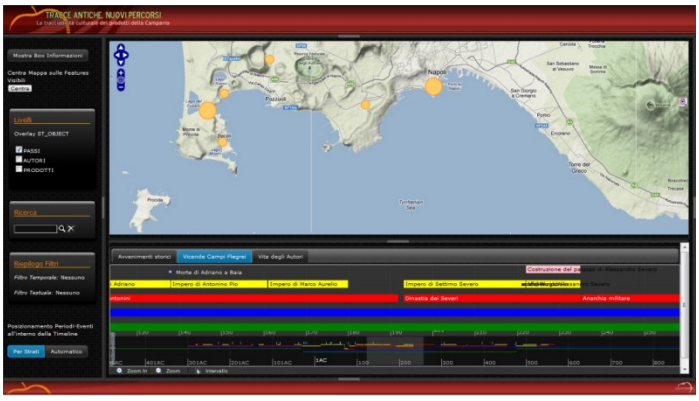

Fig. 62: Cluster Metaphor applied to Litterary Passage

The property *biographic note* in the feature *Author* is an example of a *ST-Change Postion* property. For this feature *Admin User* choose the *Path metaphor*,

that joins the *Place* on the map with a line ordered by time and allows user to detail the event by clicking on a placemark representing the event consequently, the temporal information is highlighted on the timeline component. The result of the application of the considered metaphor is shown in Fig. 63.

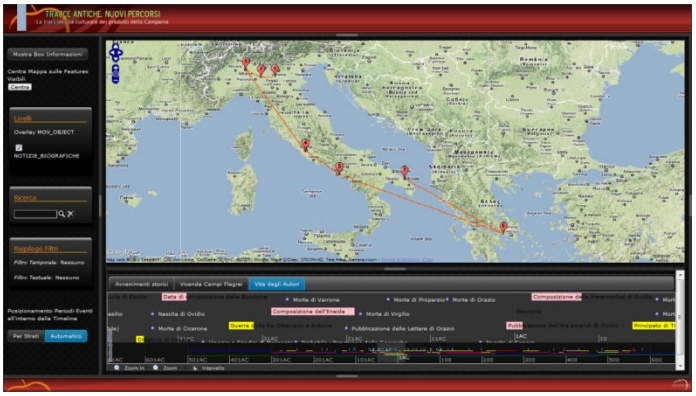

Fig. 63: Path metaphor applied to Author

6.2 Interaction Example

The CMS Front-End web interface consists of three synchronized areas (see Fig. 64): the *Timeline* that implements the visualization metaphor proposed in section 5.2.2.1, the *GeoBrowser* that allows user to make spatial queries and visualize the spatial referenced object and the *Active Layers Panel* that allows the selection of the information layer to be visualized on the map.

This interface allows users to navigate among the three dimensions: spatial, temporal and descriptive.

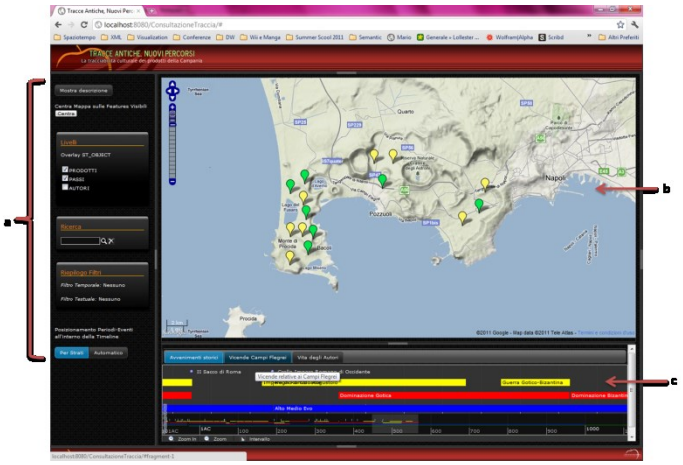

Fig. 64: Three panel web interface: a) Active Layers Panel b) Geobrowser c) Timeline

Clicking on a *Period-Event* or selecting an interval on the Timeline, users can filter information visualized on the map (Fig. 65). Timeline structure allows an easy understanding of temporal relationship (Allen, 1991) like before, after, etc., making also immediate to recognize the overlaps among *Period-Event* belonging to different layers.

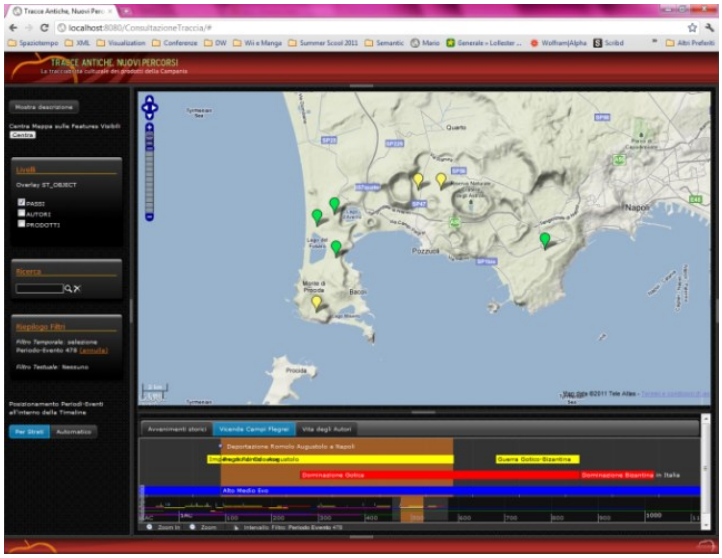

Fig. 65: Temporal Filter

Clicking on an object on the map, a balloon displaying some details shows up; clicking on a link in the balloon a further panel will appear, showing more detailed information (Fig. 66).

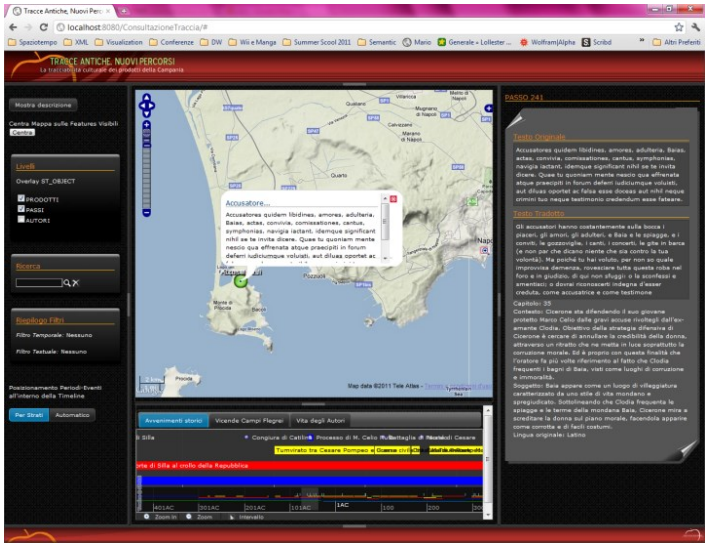

Fig. 66: Balloon and Details Panel

The *Active Layer panel* helps users to focus on the descriptive dimension. In this case this given by "*Literary Sources*", "*Authors*" and "*Product*". In this panel users can perform textual search and can filter any kind of data they want to be visualizes on the map. User can also switch among visual metaphors, for example choosing to visualize the path of an author over his lifetime (Fig. 67).

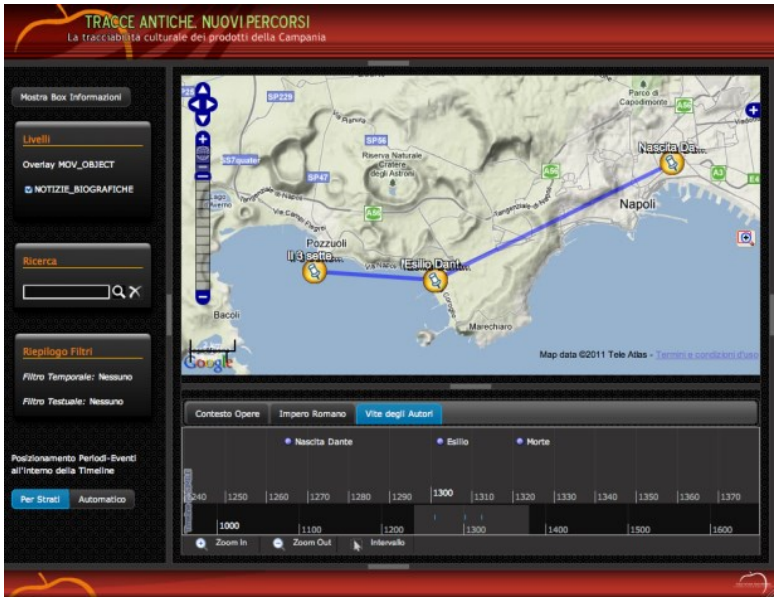

Fig. 67: Authors Life (moving object) metaphor

Conclusion and Future Work

In this thesis, a CMS for managing Spatio-Temporal data has been presented as result of our research on spatio-temporal data modeling and visualization.

The CMS is able to manage complex temporal domain structure including hierarchies, granularities and qualitative temporal references, typical of some fields like the Cultural Heritage one. We provide a temporal domain visualization metaphor that allows comparing and relating the temporal layers and relations within the domain elements.

One of the main original contributions of this thesis is the introduction of RDF technology to model hierarchical and stratified temporal and spatial domains and to manage spatio-temporal properties. The model manages spatial, temporal and spatio-temporal data independently from the application domain.

This work also provides a prototype allowing users to model their spatio-temporal data and visualize them into an integrated web-environment separating the design phases from the technical implementation of the lower layers of the system. This thesis provides a set of parameterized spatio-temporal metaphors fully customizable, allowing users to adapt visual representation to their application needs. The CMS front-end web interface allows users to navigate among the spatial, temporal and descriptive dimensions independently but in a synchronized way. The interface relies on a three-tier web architecture fully based on standard protocols and files that guarantees independence from storage and visualization tools.

In this thesis, we focus on data definition and management and for this we do not yet exploit a relevant feature of RDF techniques namely using an ontological reasoner. The focusing on this additional relevant dimension will be done in the next step of the future work. In the future work, we consider also the extension of spatio-temporal metaphor for visualization with particular attention to the browsing of the spatial hierarchy (as the one proposed for temporal domain). Moreover, we are planning extensive usability tests on specific domains of application.

Bibliography

Allen, J.F., 1983. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26, pp.832--843.

Allen, J.F., 1991. Time and Time Again: The Many Ways to Represent Time. *Journal of Intelligent Systems*, 6, pp.341--355.

Andrienko, N., 2003. Exploratory spatio-temporal visualization: an analytical review. *Journal of Visual Languages & Computing*, 14, pp.503--541.

Andrienko, N., Andrienko, G. & Gatalsky, P., 2003. Exploratory spatio-temporal visualization: an analytical review. *Journal of Visual Languages & Computing*, 14, pp.503--541.

Andrienko, G. et al., 2007. Geovisual analytics for spatial decision support: Setting the research agenda. *Int. J. Geogr. Inf. Sci.*, 21(8), pp.839--857.

Andrienko, G., Andrienko, N. & Wrobel, S., 2007. Visual analytics tools for analysis of movement data. *SIGKDD Explor. Newsl.*, 9(2), pp.38--46.

Asproth, V., Hakansson, A. & Revay, P., 1995. Dynamic Information in GIS Systems. *Computers Environment and Urban Systems*, pp.107-15.

Batsakis, S. & Petrakis, E.G., 2010. SOWL: spatio-temporal representation, reasoning and querying over the semantic web., 2010.

Becker, T., Köbben, B. & Block, C.A., 2009. TimeMapper: Visualizing Moving Object Data using WMS Time and SVG SMIL Interactive Animations. In *SVGOpen 2009: 7th international conference on scalable vector graphics*., 2009.

Berners-Lee, T., Hendler, J. & Lassila, O., 2001. The semantic web. *Scientific american*, 284, pp.28--37.

Bertino, E., Camossi, E. & Bertolotto, M., 2009. Multi-granular Spatio-temporal Object Models: Concepts and Research Directions., 2009.

Bertolotto, M., Martino, S.D., Ferrucci, F. & Kechadi, M.T., 2007. Towards a framework for mining and analysing spatio-temporal datasets. *International Journal of Geographical Information Science*, 21, pp.895-906.

Bettini, C., Wang, X.S. & Jajodia, S., 1996. A general framework and reasoning models for time granularity. In *Proceedings of Third International Workshop on*

Temporal Representation and Reasoning (TIME '96)., 1996.

Bizer, C., 2004. D2RQ - treating non-RDF databases as virtual RDF graphs., 2004.

Brickley, D. & Guha, R.V., 2004. RDF Vocabulary Description Language 1.0: RDF Schema. *W3C Recommendation*, 10.

Camossi, E., Bertolotto, M., Bertino, E. & Guerrini, G., 2003. A multigranular spatiotemporal data model. In *Proceedings of the 11th ACM international symposium on Advances in geographic information systems*. New Orleans, Louisiana, USA, 2003. ACM.

Cerasuolo, F., Cutugno, F. & Leano, V., 2012. Visualization with a New Visual Metaphor for Hierarchical and Stratified Temporal Domain. *Web and Wireless Geographical Information Systems*, pp.57--71.

Chaudhuri, S., 1988. Temporal Relationships in DataBase. In *Proceedings of the 14th VLDB Conference*., 1988.

Cormen, T.H., Leiserson, C.E., Rivest, R.L. & Stein, C., 2009. *Introduction to Algorithms (3. ed.)*. MIT Press.

Cox, S. et al., 2002. OpenGIS Geography Markup Language (GML) Implementation Specification, version.

Cutugno, F., Leano, V.A., Mangiacrapa, F. & Peron, A., 2012. A General Web-based Framework for Spatio-Temporal Exploration and Visualization applied to a Case Study on Cultural Heritage Data., 2012.

De Berg, M., Cheong, O., Van Kreveld, M. & Overmars, M., 2008. *Computational geometry: algorithms and applications*. Springer.

Doerr, M., Kritsotaki, A. & Stead, S., 2004. Which Period is it? A Methodology to Create Thesauri of Historical Periods. In *Computer Applications and Quantitative Methods in Archaeology Conference, CAA2004*. Prato, Italy, 2004.

Egenhofer, M.J., 1993. Whats special about spatial?: database requirements for vehicle navigation in geographic space. *SIGMOD Rec.*, 22(2), pp.398--402.

Franklin, C., 1992. An introduction to geographic information systems: linking maps to databases. *Database*, 15(2), pp.12--21.

Google, n.d. *Google Maps*. [Online] Available at: http://maps.google.it

[Accessed September 2011].

Gutierrez, C., Hurtado, C.A. & Vaisman, A., 2007. Introducing time into RDF. *IEEE Transactions on Knowledge and Data Engineering* , 19, pp.207--218.

Hassan-Montero, Y. & Herrero-Solana, V., 2006. Improving Tag-Clouds as Visual Information Retrieval Interfaces., 2006.

Hobbs, J.R. & Pan, F., 2006. *Time Ontology in OWL*. techreport.

InfoVis, 2013. *InfoVis Wiki*. [Online] Available at: http://www.infovis-wiki.net [Accessed 2013].

Innerebner, M., Böhlen, M. & Timko, I., 2007. A web-enabled extension of a spatio-temporal DBMS., 2007. ACM.

ISO, 2003. *ISO 19107:2003 Geographic information -- Spatial schema*. techreport.

Johnson, B. & Shneiderman, B., 1991. Tree-Maps: a space-filling approach to the visualization of hierarchical information structures., 1991. IEEE Computer Society Press.

Kisilevich, 2005. *Spatio-Temporal Clustering: a Survey*. ISTI - CNR.

Koch, W.G., 2001. Jacques Bertins theory of graphics and its development and influence on multimedia cartography. *Information Design Journal*, 10, pp.37--43.

Kolas, D., Hebeler, J. & Dean, M., 2005. Geospatial semantic web: Architecture of ontologies. *GeoSpatial Semantics*, pp.183--194.

Koubarakis, M., 1994. Database models for infinite and indefinite temporal information. *Information Systems*, 19, pp.141 - 173.

Koussoulakou, A. & Kraak, M.J., 1992. Spatio-temporal maps and cartographic communication. *Cartographic Journal, The*, 29, pp.101-08.

Lee, J., Chiu, H. & Koshak, H., 2005. Visualization System of Spatial-Temporal information for Historic sites based on GIS. In *Proceedings of Computers in Urban Planning and Urban Management (CUPUM '05) Conference*. London, UK, 2005.

Lieberman, J. et al., 2006. Geospatial semantic web interoperability experiment report. *OGC Inc. OGC*.

Lohfink, A., McPhee, D. & Ware, M., 2010. A UML-based Representation of Spatio-Temporal Evolution in Road Network Data. *Transactions in GIS*, 14, pp.853-

-872.

Lyell, M. et al., 2011. An ontology-based spatio-temporal data model and query language for use in GIS-type applications., 2011. ACM.

MacEachren, A.M., 1979. The evolution of thematic cartography/a research methodology and historical review. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 16, pp.17--33.

MacEachren, A.M. et al., 2004. Geovisualization for knowledge construction and decision support. *Computer Graphics and Applications, IEEE*, 24, pp.13 -17.

MacEachren, A.M. & Kraak, M., 2001. Research challenges in geovisualization. *Cartography and Geographic Information Science*, 28, pp.3-12.

McGuinness, D.L. & Van Harmelen, F., 2004. OWL web ontology language overview. *W3C recommendation*, 10, p.10.

Nadi, S. & Delavar, M.R., 2005. Toward a General Spatio-Temporal Database Strucuture. In *Proceedings of the International Symposium on Spatio-temporal Modeling, Spatial Reasoning, Analysis, Data Mining and Data Fusion*. Peking University, China, 2005.

Nadi, S. & Mahmoud, R.D., 2003. Spatio-Temporal Modeling of Dynamic Phenomena in GIS. In *ScanGIS'2003 - The 9th Scandinavian Research Conference on Geographical Information Science, 4-6 June 2003- Proceedings*. Espoo, Finland , 2003.

OGC, 2005. *OpenGIS web services architecture description*.

OGC, 2006. *OpenGIS Web Map Service (WMS) Implementation Specification (1.3.0)*.

OGC, 2010. *OGC Web Coverage Service 2.0 Interface Standard*.

OGC, 2010. *OpenGIS Web Feature Service 2.0 Interface Standard*. OpenGIS Implementation Standard.

OGC, 2012. *OGC GeoSPARQL - A Geographic Query Language for RDF Data*. OGC® Implementation Standard. Matthew Perry and John Herring.

Open Geospatial Consortium, 2003. *Abstract Specification Topic 1 - Feature Geometry (ISO 19107)*.

OpenLayers, 2011. *OpenLayers*. [Online] Available at: http://openlayers.org/

[Accessed September 2011].

Oracle, 2011. *Oracle Spatial and Oracle Locator*. [Online] Available at: http://www.oracle.com/it/products/database/options/spatial/index.html [Accessed September 2011].

OSGeo, 2008. MapServer. p.633.

OSGeo, 2011. *Geoserver*. [Online] Available at: http://geoserver.org [Accessed September 2011].

Parent, C., Spaccapietra, S. & Zimányi, E., 1999. Spatio-temporal conceptual models: data structures + space + time. In *Proceedings of the 7th ACM international Symposium on Advances in Geographic information Systems*., 1999.

Pelekis, N., Theodoulidis, B., I., K. & Theodoris, Y., 2005. Literature review of spatio-temporal database models. *The Knowledge Engineering Review*, pp.235--274.

Perry, M., Sheth, A., Hakimpour, F. & Jain, P., 2007. Supporting complex thematic, spatial and temporal queries over semantic web data. *GeoSpatial Semantics*, pp.228--246.

Peuquet, D.J., 1994. Its About Time - a Conceptual-Framework for the Representation of Temporal Dynamics in Geographic Information-Systems. *Annals of the Association of American Geographers*, 84(3), pp.441-61.

Pfoser, D. & Tryfona, N., 1998. Requirements, definitions, and notations for spatiotemporal application environments., 1998. ACM.

PostGis, 2011. *Postgis*. [Online] Available at: http://postgis.refractions.net/ [Accessed September 2011].

Postgres, n.d. *PostgreSQL*. [Online] [Accessed march 2011].

Prud'hommeaux, E. & Seaborne, A., 2008. *SPARQL Query Language for RDF*. techreport.

Robinson, A., Sale, R. & Morrisson, J., 1995. Elements of Cartography. *Editorial John Wiley & Sons Inc. USA*.

Sandvik, B., 2008. *Thematic Mapping Blog*. [Online] Available at: http://thematicmapping.org/ [Accessed 2013].

Shekhar, S. & Xiong, H., 2008. *Encyclopedia of GIS*. Springer Verlag.

Shneiderman, B., 1996. The eyes have it: a task by data type taxonomy for information visualizations. In *IEEE Symposium on Visual Languages Proceedings.*, 1996.

Slingsby, A., Dykes, J. & Wood, J., 2008. Using treemaps for variable selection in spatio-temporal visualisation. *Information Visualization*, 7(3), pp.210--224.

Slingsby, A., Dykes, J., Wood, J. & Clarke, K., 2007. Interactive Tag Maps and Tag Clouds for the Multiscale Exploration of Large Spatio-temporal Datasets., 2007. IEEE Computer Society.

Slocum, T.A., McMaster, R.B., Kessler, F.C. & Howard, H.H., 2009. *Thematic cartography and geovisualization*. Pearson Prentice Hall Upper Saddle River, NJ.

Spaccapietra, S., Cullot, N., Parent, C. & Vangenot, C., 2004. On spatial ontologies.

Stefanakis, E., 2008. A journey to the ancient greek myths - An enhanced educational framework to story-telling with geo-visualization capabilities. In *Proceedings of the First International Workshop on Story-Telling and Educational Games (STEG '08)*. Maastricht, The Netherlands, 2008.

Tryfona, N., Price, R. & Jensen, C.S., 2003. Conceptual Models for Spatio-temporal Applications., 2003.

Valenti, R., 2011. *Intorno ai Campi Flegrei - Memorie dell'acqua e della Terra*. Napoli: Grimaldi & C. Editori.

van Beek, P. & Manchak, D.W., 1996. The Design and Experimental Analysis of Algorithms for Temporal Reasoning. *Journal of Artificial Intelligence Research*, 4, pp.1--18.

Vilain, M., Kautz, H. & Van Beek, P., 1989. Constraint propagation algorithms for temporal reasoning: A revised report. *Readings in qualitative reasoning about physical systems*, pp.373--381.

Vretanos, P.A., 2010. OpenGIS Filter Encoding 2.0 Encoding Standard. 2010.

W3C, 2004. *Resource Description Framework (RDF): Concepts and Abstract Syntax*. [Online] Available at: http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/ [Accessed February 2013].

W3C, 2010. *XQuery 1.0: An XML Query Language (Second Edition)*. [Online]

Available at: http://www.w3.org/TR/xquery/ [Accessed 2011].

Wikipedia, 2013. *Thematic Mapping*. [Online] Available at: http://en.wikipedia.org/wiki/Thematic_map [Accessed 2013].

Wilson, T., 2008. OGC Keyhole Markup Language, 2.2. 0. *Open GIS Consortium*.

Worboys, M.F., 1994. A Unified Model for Spatial and Temporal Information. *Comput. J.*, 37, pp.36-34.

Yuan, M., 1996. Temporal GIS and Spatio-Temporal Modeling. In *Proceedings of Third International Conference on Integrating GIS and Environmental Modeling*. Santa Fe, New Mexico, Usa, 1996. NCGIA.