

# Ontology based framework for data integration

ALBERTO SALGUERO, FRANCISCO ARAQUE, CECILIA DELGADO

Department of Software Engineering :: ETSIIT

University of Granada (Andalucía)

C/ Periodista Daniel Saucedo Aranda s/n

SPAIN

agsh, faraque, cdelgado { @ugr.es }

*Abstract:* One of the most complex issues of the data integration is the case where there are multiple sources for the same data element. It is not easy to generate and maintain the integrated scheme. In this paper we describe a framework which encompasses the entire data integration process. The data source schemas as well as the integrated schema are expressed using an extension of an ontology definition language which allows the incorporation of metadata to support the integration process. The proposed model allows the user to concentrate in modeling the problem itself and not in the issues of dealing with the temporal and the spatial aspects concerning to many of the data sources usually used in the enterprises information systems.

*Keywords:* temporal model, data integration, ontology, OWL, metadata, enterprise information system.

## 1 Introduction

There are many problems, from the computational point of view, whose solution involves the use of temporal or spatial concepts. In the case of information systems, which are often used by companies in their strategic level, sometimes occurs that many of the data sources from which the information is obtained contain, in a way, this kind of spatial and temporal information. Moreover, because of the proper nature of information systems based on Data Warehouses (DW) (which we are interested in), data sources contain an intrinsic temporal characteristics that establish the proper manner to extract their data.

A DW is a database that stores a copy of operational data with an optimized structure for query and analysis. The data coming from the sources are integrated in the data warehouse to satisfy the necessities of collective access to the data, providing an historical vision of the data of the different operational systems.

Usually, whenever a company is faced with the task of including a new data source in its information system, they often face the process virtually from scratch. There are very few tools for assisting in the process of designing information systems based on DW [1] and the inclusion of new data sources may involve re-designing the scheme of DW.

This paper describes an ontology based model serving as the pivot which the different data source schemes are translated to (Canonical Data Model). From the data source schemes it is possible to generate an integrated DW straightforwardly. This

data model defines the spatial and temporal concepts used more frequently so they can be reused in each specific problem and the efforts are focused on the problem itself, and not in the definition of spatial and temporal information concepts in the source.

On the other hand, this data model, heavily oriented to work with information systems based on DW, also has the ability to annotate each of the data sources with information about their own temporal characteristics, making it possible to generate semi-automatically a set of rules that determine the most efficient way to extract information according to the level of detail required by the DW designer without having to query the data sources more than necessary.

The use of a data model based on ontologies is proposed as a common data model to deal with the data source schemes integration. This idea is based on [2], a project being developed by members of the University of Granada. Although it is not the first time the ontology model has been proposed for this purpose [3], [4], [5], to our knowledge, this is the first time the metadata storage capabilities of some ontology definition languages has been used in order to improve the DW data refreshment process design.

The remainder of the paper is organized as follows. In Section 2 some basic concepts are revised. Section 3 introduces our spatio-temporal model. Then, in section 4, we present all the involved components in the process. In Section 5 the usage of the spatio-temporal model is illustrated. Section 6 describes the data integration process and, finally, in section 7 the conclusions are presented.

## 2 Data Warehouses and ontologies

It is obvious that there is no organization running without data. The data can be viewed as tangible assets of an organization just as any physical asset. So, they need to be stored and made available to those who need them in order to be used at any moment. Since the data by themselves are useless, they must be put together to produce useful information. In turn, information becomes the basis for relational decision making. To facilitate the decision-making process, a new piece of technology more sophisticated than a database system was developed and called Data Warehouse (DW). The DW can be generally described as a decision-support tool that collects its data from operational databases and various external sources, transforms them into information and makes that information available to decision-makers (top managers) in a consolidated and consistent manner [4], [7], [8]. The persistence of huge amounts of data (possibly distributed and heterogeneous) opens a new perspective for various statistical analysis methods which are essential for strategic decisions in tourism.

Inmon [4] defined a Data Warehouse as “a subject-oriented, integrated, time-variant, non-volatile collection of data in support of management’s decision-making process”. A DW is a database that stores a copy of operational data with an optimized structure for query and analysis [8]. The scope is one of the issues which define the DW: it is the entire enterprise. In terms of a more limited scope, a new concept is defined: a Data Mart (DM) is a highly focused DW covering a single department or subject area.

The generic architecture of a DW is illustrated in Figure 1. Data sources include existing operational databases and flat files (i.e., spreadsheets or text files) in combination with external databases. The data are extracted from the sources and then loaded into the DW using various data loaders and ETL tools. ETL stands for extract, transform and load, the processes that enable companies to move data from multiple sources, reformat and cleanse it, and load it into another database or on another operational system to support a business process. The warehouse is then used to populate the various subject (or process) oriented data marts and OLAP servers. Data marts are subsets of a DW categorized according to functional areas depending on the domain (problem area being addressed) and OLAP servers are software tools that help a user to prepare data for analysis, query processing, reporting and data mining. Thus, a DW coupled with OLAP enables managers to creatively approach, analyze and understand the problems. The

OLAP analyzes data using special DW schemas and enables users to view data using any combination of variables.

A federated database system (FDBS) is formed by different component database systems; it provides integrated access to them: they co-operate (inter-operate) with each other to produce consolidated answers to the queries defined over the FDBS. Generally, the FDBS has no data of its own as the DW has. Queries are answered in the FDBS by accessing the component database systems.

We have extended the Sheth & Larson five-level FDBS architecture [9], which is very general and encompasses most of the previously existing architectures. In this architecture three types of data models are used: first, each component database can have its own native model; second, a *canonical data model* (CDM) which is adopted in the FDBS; and third, external schema can be defined in different *user models*.

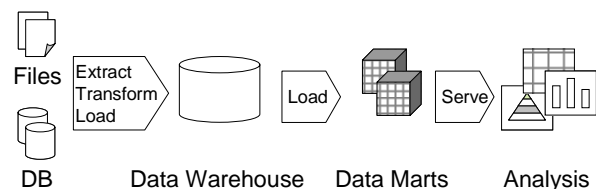


Fig.1. A generic Data Warehouse architecture.

One of the fundamental characteristics of a DW is its temporal dimension, so the scheme of the warehouse has to be able to reflect the temporal properties of the data. The extracting mechanisms of this kind of data from operational system will be also important. In order to carry out the integration process, it will be necessary to transfer the data of the data sources, probably specified in different data models, to a common data model, that will be the used as the model to design the scheme of the warehouse. In our case, we have decided to use an ontological model as canonical data model.

An ontology is the specification of a conceptualisation of a knowledge domain. It is a controlled vocabulary that describes objects and the relations between them in a formal way, and has a grammar for using the vocabulary terms to express something meaningful within a specified domain of interest. They allow the use of automatic reasoning methods. OWL is a language for defining ontologies. OWL ontologies may be categorised into three species or sub-languages: OWL-Lite, OWL-DL and OWL-Full.

OWL-Lite is the syntactically simplest sub-language. It is intended to be used in situations where only a simple class hierarchy and simple constraints

are needed. OWL-Full is the most expressive OWL sub-language but is not able to guarantee the decidability or computational completeness of the language, so it is not always possible to perform automated reasoning on it. OWL-DL is much more expressive than OWL-Lite and allows the use of automated reasoning restricting some OWL-Full constructions.

We have extended OWL with temporal and spatial elements. We call this OWL extension STOWL. This is also our proposal: use an ontological-oriented model as CDM, enhanced with temporal and spatial features, for the definition of the data warehouse and data mart schema to define the refreshment of the data warehouse and data marts.

### 3 STOWL: The Spatio-Temporal OWL

According to the integration architecture proposed in [10], [11], [13] it is necessary the selection of a CDM which serves as pivot which all the schemes that specify the structure of the different information sources and the DW will be translated to. The obtained DW scheme will generate, in turn, the schemes which will be used by the query and the data analysis tools. The requirements to be performed by this model are:

- *Correctly treatment of temporal information.* Being an architecture geared towards building data warehouses it must be able to provide high-quality support for this type of information because it is the basis of this kind of architecture (a data warehouse is basically a set of historical data). This is achieved by defining temporal primitives.
- *Support for the adequate representation of spatial information: geographical, topological...* The model must define spatial primitives allowing the representation and the proper treatment of information from the different data sources.
- *Semantic data annotation to support their integration.* It should includes all the available information in the data sources to make easy the design of the extracting, transforming, loading and refreshing data tasks from the DW.

The data models based on ontologies are a good option as a CDM but too general. All studied language proposals for defining ontologies used ultimately a very limited set of basic data types to define underlying concepts. This set of basic types are usually very general and does not support the spatial information types. Extending any of the languages for defining ontologies to cover these

requirements seem the most suitable option. Among all of them, OWL language is selected as the base for defining the CDM.

Starting from the OWL basic data type set we can express in form of ontology the common spatio-temporal concepts required for the schema definition of data sources to perform the integration phase. Thus, when describing the scheme of the data source, the design process will focus on the data source characteristics and not in the process of specifying spatio-temporal concepts.

Firstly an ontology in the OWL language will be built to define the generic temporal primitives found of interest. Then the spatial primitives will also be incorporated. Finally, the information that describes the characteristics of data to integrate, i.e. the metadata which will be useful to design the data extracting, loading and refreshing processes, will be incorporated to the data source scheme using the annotation properties of OWL. Annotation properties are a special kind of OWL properties which can be used to add information to classes, individuals and object/datatype properties. In fact, they act as the data source metadata, which has been successfully used for data integration [11].

The result will be an ontology, expressed in OWL, defining the spatial and temporal primitives which will be used to build the schemes of the data sources and a set of properties which will allow the addition of information about the data sources characteristics. We call STOWL (Spatio-Temporal OWL) to this base ontology.

Basically, STOWL is an application of OWL. The primitives on which STOWL rests come basically from GML (spatial information) and ODMG-T (temporal information) data models.

OWL allows classes, properties, individuals and the ontology itself (technically speaking the ontology header) to be annotated with various pieces of information/metadata. These pieces of information may take the form of auditing or editorial information. For example, comments, creation date, author, or, references to resources such as web pages etc.

OWL Full does not impose any restriction in the use of the annotation properties. The problem of using OWL Full is that the ontologies expressed in this language does not always make possible the use of automatic reasoners on them, which results in the loss of one of the best characteristics that made us decide on using this kind data model. To avoid this issue OWL Full will not be used until necessary. The advantage of using the annotation properties of OWL is that, under certain restrictions, the resulting ontology still satisfies the OWL DL requirements:

- The filler for annotation properties must either be a data literal, a URI reference or an individual.
- Annotation properties cannot be used in property axioms — for example they may not be used in the property hierarchy, so they cannot have sub properties, or be the sub property of another property. They also must not have a domain and a range set for them.

The ontology itself can be seen as a resource that describes the scheme of a data source. Therefore, this scheme can be annotated with information about the general characteristics of the source. This information will be used to design the extracting, transforming and loading process of the data in the DW.

### 3.1 Temporal annotation of data

Due to the nature of the DW the annotation properties will usually refer to the temporal characteristics of data, so a set of annotation properties is defined to describe the sources according to some of the temporal concepts studied in [11]. All these properties are associated directly to the ontology viewed as a resource and not individually to each resource defined in the ontology. This means that all the resources of the ontology share the values assigned to these properties (expected if extracted from the same data source).

```
<owl:DatatypeProperty rdf:ID="hasExtractionTime">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#AnnotationProperty"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
</owl:DatatypeProperty>
```

Figure 2: Extraction Time definition in STOWL.

STOWL defines, for instance, the Extraction Time of a change in a data source as shown in figure 2. The Extraction Time parameter can be defined as the time expended in extracting a data change from the source. Some examples of the temporal parameters [14] that we consider of interest for the integration process are:

- Availability Window (AW). Period of time in which the data source can be accessed by the monitoring programs responsible for data source extraction.
- Extraction Time (ET). Period of time taken by the monitoring program to extract significant data from the source.
- Transaction time (TT). time instant when the data element is recorded in the data source computer system. This would be the data source transaction time.

- Storage time (ST). Maximum time interval for the delta file, log file, or a source image to be stored.
- Temporal Reference System (TRS): a temporal reference system provides standard units for measuring time and describing temporal length or duration. Gregorian calendar with UTC is often used as the default temporal reference system.

All the temporal concepts explained previously can be applied to a data source as a whole because all data from the same source share these temporal characteristics. On the other hand, there are certain characteristics that should be defined in more detail. Within the scheme of the data source, each of the data fields that can be extracted can be annotated semantically with temporal information to further facilitate their integration into the data warehouse.

As with the case of generic data source features, the annotation properties to describe the characteristics of different data fields will also be used. For the temporal aspects of data STOWL defines the following property.

- Temporal Granularity (TGr): It is the extent to which a system contains discrete components of ever-smaller size. Generally speaking, information granules are collection of entities, usually originating at the numeric level, that are arranged together due to their similarity, functional adjacency, indistinguishability, coherency or the like. In our case, because we are dealing with time, it is common to work with granules like minute, day, month...

### 3.2 Spatial annotation of data

Once incorporated the temporal primitives to the CDM it is necessary to define the spatial primitives. After reviewing various spatial data models we have chosen to follow the model proposed by the Open GIS Consortium. The main reason is that it is a standard widely accepted by the majors companies working on GIS systems. This standard, called Geography Markup Language (GML), is an XML grammar written in XML Schema for the modelling, transportation and storage of geographic information. GML is often used as a communication protocol between a large set of GIS applications, both commercial and open source. This spatial data model has been tested and polished over years and it is accepted by most of GIS related companies and associations. It has been necessary the translation of GML, written in XML Schema, to OWL.

GML defines several kinds of entities (such as features, geometrical entities, topological entities...) in form of an object hierarchy. As well as the temporal properties they have been incorporated to STOWL in order to support the inclusion of spatial metadata in the data sources schemes.

There are also spatial characteristics which can be associated to the data source as well as other kind of spatial characteristics which must be associated to each data field of the data source. In the former set we have, for instance, the *Coordinate Reference System* (CRS) concept. A coordinate reference system consists of a set of coordinate system axes that is related to the earth through a datum that defines the size and shape of the earth. A CRS provides a method for assigning values to a location. All the data in a data source must share the same CRS.

In the latter set of spatial characteristics we consider, for instance, the *Spatial Granularity* (SGR).

In this case, because we are dealing with spatiality, it is common to work with granules like meter, kilometer, country...

As with the temporal data source features, the annotation properties are used to describe the source spatial metadata.

### 4 DW Architecture

Taking paper [14] as point of departure, we propose the reference architecture in figure 3. In this figure, the data flow as well as the metadata flow are illustrated. Metadata flow represents how all the data that refers to the data, i.e. the schemes of the data sources, the rules for integrating the data..., are populated through the processing units of the architecture. These processing units have been designed to be independent among them.

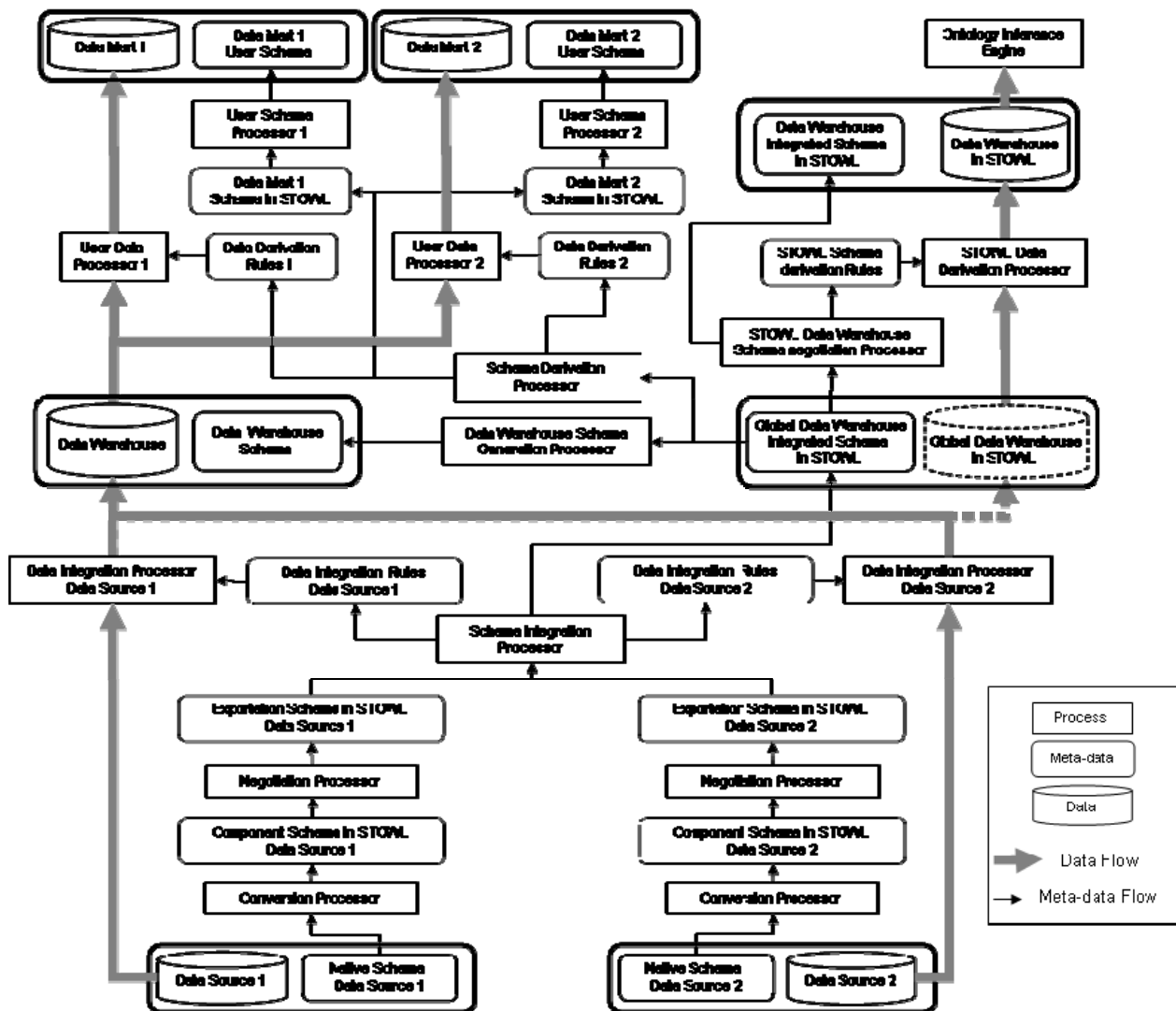


Figure 3: Functional architecture

The general external architecture of a metadata processor is depicted in figure 4. A metadata processor is responsible of dealing with multiple instances of metadata and generate, at least, another metadata instance as a result.

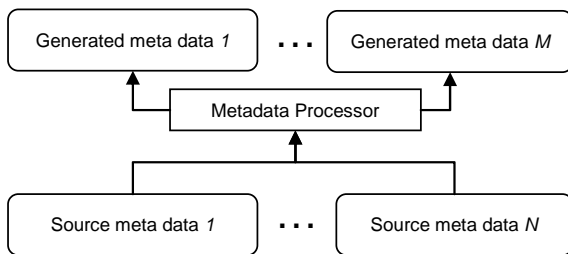


Figure 4: Generic metadata processor external architecture

On the other hand, the data processors are responsible of dealing with data. According to the metadata generated by the metadata processors (in form of rules) the data processors generate a data set as a result starting from another data set (fig 5).

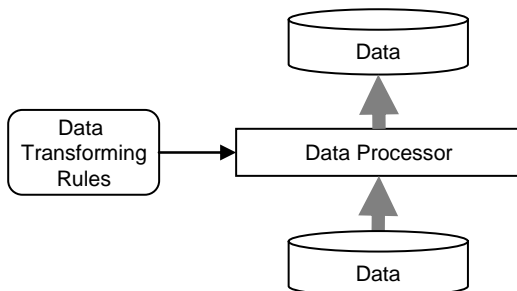


Figure 5: Generic data processor external architecture

The architecture in figure 3 is designed following these patterns. As it has been explained in section 2, it extends the Sheth & Larson FDBS base architecture [9]. Following are explained the involved component.

**Native Schema.** Initially we have the different data source schemes expressed in its native schemes. Each data source will have, a scheme, the data inherent to the source and the metadata of its scheme. In the metadata we will have huge temporal information about the source: temporal and spatial data on the scheme, metadata on availability of the source (availability of the log file or delta if it had them, etc).

**Preintegration.** In the *Preintegration* phase, the semantic enrichment of the data source native schemes is made by the conversion processor. In addition, the data source temporal and spatial metadata are used to enrich the data source scheme with temporal and spatial properties. We obtain the component scheme (CS) expressed in the CDM, in

our case using STOWL (OWL enriched with temporal and spatial elements).

From the CS, expressed in STOWL, the negotiation processor generates the export schemes (ES) also expressed in STOWL. The ES represents the part of a component scheme which is available for the DW designer. It is expressed in the same CDM as the Component Scheme. These ES are the part of the CS that is considered necessary for its integration in the DW. For security or privacy reasons part of the CS can be hidden. The ES can be seen as external schemes of the CS.

External schema generation involves obtaining all the relationships existing between the classes selected to make up an external schema. We propose the use of the method described in [15], [16] for defining external schemas in object oriented models (ODMG actually), adapted to the ontology model. External schemas generated with this process avoids the generation of unnecessary intermediate classes.

**Integration.** The DW scheme corresponds to the integration of multiple ES according to the DW designer needs. It is expressed in an enriched CDM (STOWL in our case) so that temporal and spatial concepts could be expressed straightforwardly. This process is made by the Schema Integration Processor which suggests how to integrate the Export Schemes, helping to solve semantic heterogeneities (out of the scope of this paper), and defining the Extracting, Transforming and Loading processes (ETL). In the definition of the DW scheme, the DW Processor participates in order to contemplate the characteristics of structuring and storage of the data in the DW.

The integration processor consist of two modules which have been added to the reference architecture in order to carry out the integration of the temporal and spatial properties of data, considering the data source extraction method used: the *Temporal and Spatial Integration Processor* and the *Metadata Refreshment Generator*.

The *Temporal and Spatial Integration Processor* uses the set of semantic relations and the conformed schemes obtained during the detection phase of similarities [17]. This phase is part of the integration methodology of data schemes. As a result, we obtain data in form of rules about the integration possibilities existing between the originating data from the data sources (minimum resultant granularity...). This information is kept in the Warehouse Scheme using STOWL as well as the data sources were annotated with temporal and spatial metadata.

In addition, as a result of the integration process, a set of mapping functions is obtained. This set of functions identifies the attributes of the schemes of

the data sources that are self-integrated to obtain an attribute of the DW scheme.

The *Metadata Refreshment Generator* determines the most suitable parameters to carry out the refreshment of data in the DW scheme [18]. The DW scheme is generated in the resolution phase of the methodology of integration of schemes of data. It is in this second phase where, from the minimum requirements generated by the temporal integration and stored in the Temporal Metadata warehouse, the DW designer fixes the refreshment parameters. As result, the DW scheme is obtained along with the Refreshment Metadata necessary to update the former according to the data extraction method and other temporal and spatial properties of a concrete data source.

Obtaining of the DW scheme is not a linear process. We need the Integration and Negotiation Processors collaborate in an iterative process where the participation of the local and DW administrators is necessary [17]. Taking both the minimum requirements to fulfil the needs of carrying out the integration between two data of different data sources (obtained by means of the Temporal Integration module) and the integrated scheme (obtained by the resolution module) the refreshment parameters of the data stored in the DW are established.

**Data Warehouse Refreshment.** After the schema integration and once the DW scheme is obtained, its maintenance and update will be necessary. This function is carried out by the Data Integration Processor. This processor is detailed in section 6.

## 5 Defining new data source schemes

We have developed a plug-in for *Protégé* which allows an easy definition of new data sources schemes using the STOWL language. *Protégé* is a free, open source ontology editor and a knowledge acquisition system. It is being developed at Stanford University in collaboration with the University of Manchester.

The operation of the plug-in is pretty simple. From all of the properties defined in STOWL, those considered to be of interest for defining the schemes of data sources by users are modified to be also a subproperty of a property called “externalProperty”. All these properties will be available to the users when designing the schemes of the data sources. This approach allows an easy and transparent modification of the classes and basic properties of STOWL for the end user.

Let suppose a company which needs to incorporate tourist information from a database of hotels to its information system. One of the typical properties that

share the hotels is the definition of different holiday periods depending on the demand (high, medium and low seasons). In this case, the user should follow the following sequence of steps to define this property using the plug-in we have developed:

1. *Create a new class named “Hotel”*. The plug-in creates a new class with the name chosen by the user and makes it inherits directly from the class “OWL: Thing”.
2. *Create a new attribute for the class “Hotel” called “hasOccupation”*. Internally, the plug-in creates a new property called “hasOccupation” and assign the class “Hotel” as its domain.
3. *Select STOWL property that best matches*. Using a drop-down menu, containing the complete list of properties defined as external in STOWL, the user has to select the property which best mach with the new created property. Following with the tourist example, the most appropriate property in this case is the “hasInstant” property whose range is instances of the class “TimeInstant”. The new property “hasOccupation” will be created by the plug-in in such a way that it inherits directly from the property “hasInstant” defined by STWOL, inheriting all its features.

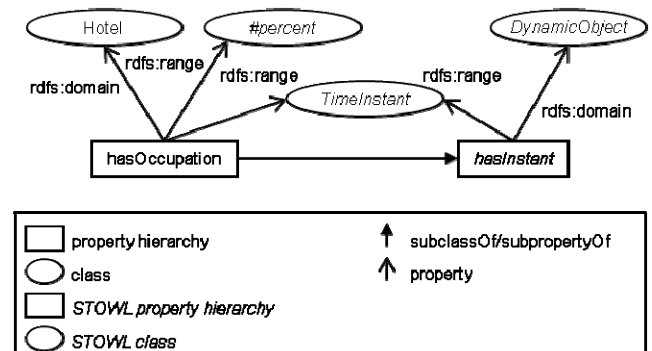


Figure 6. hasOccupation property definition as hasInstant subproperty.

In figure 6 it is shown how the new “hasOccupation” property is defined and how it is linked with the rest of the properties. It is important to note that, because of the inherent reasoning capability of ontology based model, the class “Hotel” would also be automatically assigned to the class of dynamic objects (the domain of the property “hasOccupation” is the class “DynamicObject”) without the user having to take part. Using this ontology feature is possible to make queries involving some kind of reasoning. This feature can also be used in the integration process of the different data sources schemes.



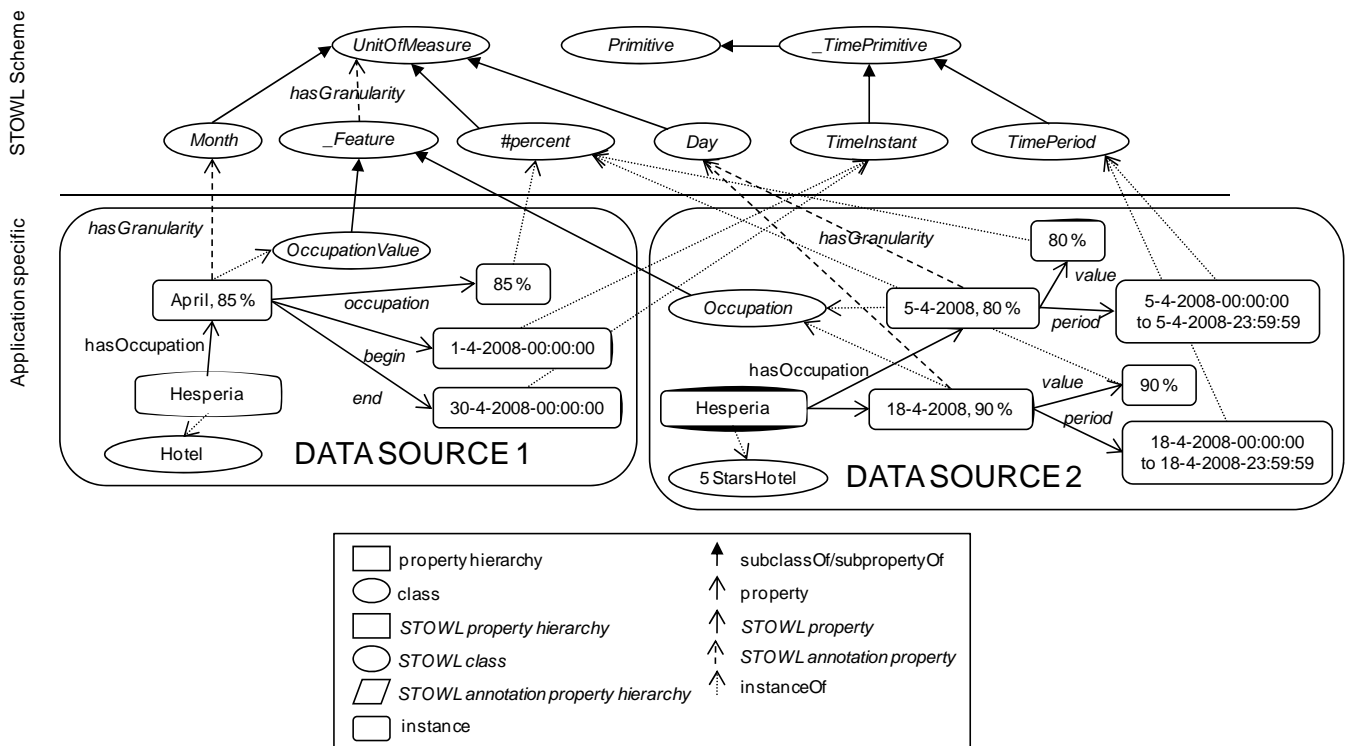


Figure 7. hasHighSeason property usage.

Once defined the “hasOccupation” property we can use it to describe the data in the data sources, i.e. the ontology instances. In figure 7 it is illustrated how the new “hasOccupation” would be used to indicate that that Hesperia hotel has an occupation of 85 % at April (data source 1). The objects below the line are the only classes or properties the developer of the data source scheme should create. The classes and properties above the line are objects defined by STOWL and reusable in different applications and data sources scheme.

It is also important to notice that the occupation value instance “April, 85 %” has a granularity of month. To indicate this fact the designer of the data source scheme has used the annotation property “hasGranularity” of the class “\_Feature”. Knowing this information it is possible to assure that it is not possible to obtain an integrated scheme where the level of detail of the season time periods will be smaller than a month. The DW refreshing process can take advantage of this knowledge and query the data sources consequently.

You may also notice that in figure 6, although both data sources are designed to store the same kind of information, their schemes are slightly the different. Not only the names of the concepts are different. The former data source uses a triple to store the occupation values whereas the latter uses a duple,

replacing the pair of instants by a time period. We can take advantage of the reasoning capabilities of the ontologies to discover when different concepts in different schemes represent the same concept and, consequently, integrate them in the DW scheme.

## 6 Data integration

Once annotated the data sources and DW schemes the next step is the integration of the data. The Data Integration Processor is responsible of doing this task.

The set of Data Integration Processors can actually be seen as a unique data warehouse data refreshment processor (see figure 8). For the sake of simplicity it has been sketched independently in figure 3 but, in fact, they cooperate in the refreshment process to produce the integrated data. Each Data Integration Processor is responsible of doing the incremental capture of its corresponding data source and transforming them to solve the semantic heterogeneities according to the integration rules obtained in the integration phase. Each Data Integration Processor accesses to its corresponding data source according to the temporal and spatial requirements obtained in the integration stage.



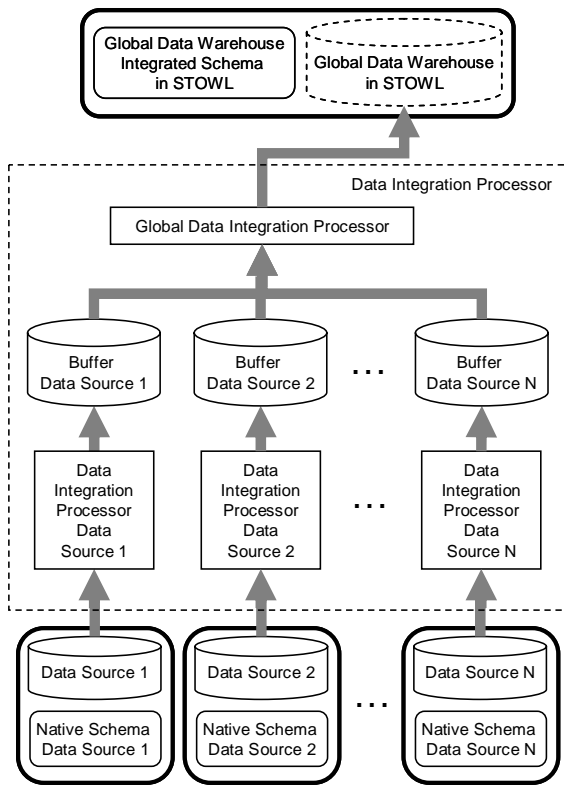


Figure 8: Data Integration Processor

If each Data Integration Processor acts as a data producer there must exist another processor which acts as a consumer. This is the case of the Global Data Integrator Processor which uses a parallel, fuzzy data integration algorithm to integrate the data independently generated by each Data Integration Processor [13].

The *Global Data Integrator Processor*, which uses a fuzzy data integration algorithm, integrates the data independently generated by each Data Integration Processor. The procedure is directed by the data source with minimum granularity level (the one with the largest granule). The processor retrieves those values by accessing to the STWOL data sources schemas, which has been extended to support this kind of metadata [10]. All the changes detected in this data source are aggregated into one unique value. The aggregation operator has to be defined by the DWA (maximum, minimum, average...). The date attribute of the changes has to be also integrated according to the same standard aggregation operator (date values are treated always as integer values). The changes in the rest data sources will be aggregated according to the similarity degree between their dates and this aggregated date value. This process is performed every DW granule length for the integrated value, i.e. the target granularity.

Following with the example of the touristic company, let suppose we have to integrate two

different data sources about occupation of the hotels. One of the data sources stores the information about the occupation with a level of detail of months (see figure 7), i.e. the occupation of some hotel in April, for instance, has been the 80%. The other data source also stores this information but, in this case, it is stored with a level of detail of days, i.e. the occupation of a hotel is specified considering the days of the month (the 3<sup>rd</sup> of April the hotel has an occupation of 80% and the 18<sup>th</sup> of April has an occupation of 90%, for instance). This metainformation about the data source is annotated in its scheme by mean of the annotation properties of OWL, as shown in point 3.5. In this case, the property that stored this kind of information is “hasGranularity” and it is defined in the same way the property “hasExtractionTime” was introduced in figure 2.

According to the algorithms in [14], the former data source would be queried only once a month at most whereas de latter would be queried once a day (if we do not want to lose any change), producing zero or many values for each month.

But, what is exactly the value we must store in the DW for a specific month? These values along with the value of the former data source should be integrated in one unique value in the DW. For this, we use a fuzzy algorithm [13] which takes into account the temporal characteristics and the data extracting method of the source. Basically, the algorithm uses the average of the values, weighted by their similarity with respect to the measured feature (fig. 9). The resultant value is supposed to be more accurate than doing the average or discarding some of the values.

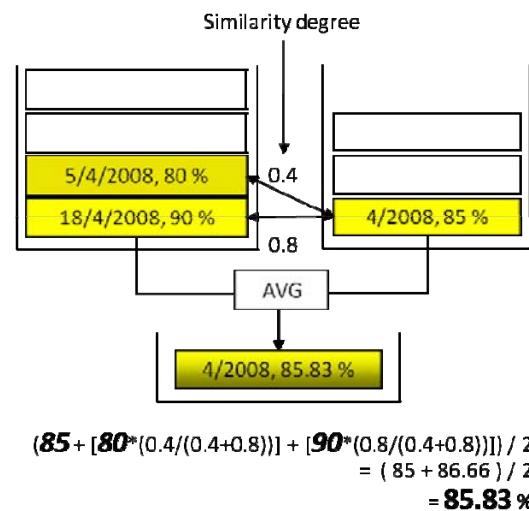


Figure 9. Data integration example.

## 7 Conclusions


A data warehouse is a database that stores a copy of operational data with an optimized structure. The data coming from the sources are integrated in the data warehouse to satisfy the necessities of collective access to the data.

To carry out the data integration process, it is necessary to transfer the data from the data sources, probably specified in different data models, to the data warehouse. To overcome these differences and for design the data warehouse schema it is used a common data model.

As common data model, we have suggested an ontology-based model; in particular, we have defined an spatio-temporal extension of the OWL language, and to store metadata about the temporal data sources characteristics we propose the use of the annotation properties of OWL.

Making use of the reasoning capabilities of OWL it is possible to derive rules to integrate the schemes of the different data sources and to design efficient DW refreshing process.

## Acknowledgements

This work has been supported by the  Research Program under project GR2007/07-2 and by the Spanish Research Program under projects EA-2007-0228 and TIN2005-09098-C05-03.

## References

- [1] Thalhammer, T., Schrefl, M., Mohania, M., 2001. Active data warehouses: complementing OLAP with analysis rules, *Data & Knowledge Engineering*, v.39 n.3. 241-269.
- [2] Samos, J., Araque, F., Carrasco, Corral, A., R., Delgado, C., Garvı, E., Ruız, E., Salguero, A. and Torres, M., 2006-2008. Ontology-based spatio-temporal DW models research project. TIN2005-09098-C05-03. Spain.
- [3] Skotas, D., Simitsis, A., 2006. Ontology-Based Conceptual Design of ETL Processes for Both Structured and Semi-Structured Data. *International Journal on Semantic Web and Information Systems*, Vol. 3, Issue 4. pp. 1-24.
- [4] Goudos, S. K., Peristeras, V., Tarabanis, K. A., 2007. Semantic Web Application for Public Administration using OWL for Public Domain Data Knowledge Representation. *WSEAS Transactions on Information Science & Applications*. Issue 4, Volume 4. Pp. 725-730.
- [5] S. Y. Yang, 2006. How Does Ontology help Web Information Management Processing. *WSEAS Transactions on Computers*. Issue 9, Volume 5. Pp. 1843-1850.
- [6] Inmon W.H, 2002. Building the Data Warehouse. *John Wiley*.
- [7] Kimball, R., & Ross, M. The Data Warehouse Toolkit: The Complete Guide To Dimensional Modeling, 2nd Edition. *John Wiley*. 2002.
- [8] Harinarayan, V., Rajaraman, A., Ullman, J., 1996. Implementing Data Cubes Efficiently. *Proc. of ACM SIGMOD Conference*. Montreal.
- [9] Sheth, A., Larson, J., 1990. Federated Database Systems for Managing Distributed, Heterogeneous and Autonomous Databases." *ACM Computing Surveys*, Vol. 22, No. 3.
- [10] Salguero, A., Araque, F. and Delgado, C., 2008. Using ontology meta data for data warehousing. *10th Int. Conf. on Enterprise Information Systems (ICEIS)*. Barcelona, Spain.
- [11] Huang, S. H., Ke, H. R., Yang, W. P., 2006. Using Metadata to Integrate Digital Libraries by Three-Layer Architecture. *WSEAS Transactions on Computers*. Issue 10, Volume 5. Pp. 2301-2308.
- [12] Araque, F., Salguero, A., Delgado, C., 2007. Monitoring web data sources using temporal properties as an external resources of a data warehouse. *ICEIS*. 28-35.
- [13] Araque, F., Carrasco, R. A., Salguero, A., Delgado, C., Vila, M. A., 2007b. Fuzzy Integration of a Web data sources for Data Warehousing. *Lecture Notes in Computer Science (Vol 4739)*. Springer-Verlag.
- [14] Araque, F., Salguero, A., Delgado, C., Samos, J., 2006. Algorithms for integrating temporal properties of data in DW. *8th Int. Conf. on Enterprise Information Systems (ICEIS)*. Paphos, Cyprus. May.
- [15] Torres, M., Samos, J., 2001. Generation of External Schemas in ODMG Databases. In proceedings of International Database Engineering and Applications Symposium (IDEAS'2001), IEEE Computer Society Press, pp. 89-98, Grenoble.
- [16] Araque, F, Samos, J., 1999. External Schemas in Real-Time Object-Oriented Databases. *20th IEEE Real-Time Systems Symposium*, WIP Proceedings, pp. 105-109. Phoenix.
- [17] Oliva, M., Saltor, F., 1996. A Negotiation Process Approach for Building Federated Databases. In Proceedings of 10th ERCIM Database Research Group Workshop on Heterogeneous Information Management, Prague. 43-49.
- [18] Araque, F., Samos, J., 2003. Data warehouse refreshment maintaining temporal consistency. *5th Intern. Conference on Enterprise Information Systems, ICEIS '03*. Angers. France.