International Journal of Computer Information Systems and Industrial Management Applications (IJCISIM)
ISSN: 2150-7988 Vol.1 (2009), pp.266-278
http://www.mirlabs.org/ijcisim

How can Ontologies Support Digital and Paper Archives? A Case Study

Ugo Barchetti, Alberto Bucciero, Anna Lisa Guido, Luca Mainetti, Roberto Paiano
University of Salento
Via Monteroni
73100, Lecce, Italy
+39 0832 297376
{ugo.barchetti,alberto.bucciero, annalisa.guido, luca.mainetti, roberto.paiano} @unisalento.it

Abstract

Up to some years ago, industries operating in the field of document filing systems had a great boost to their growth, driven, above all, by the need of Public Administrations and of large-scale industries to externalize their paper archives. Now document filing system companies are slowly trying to convert themselves into digital enterprises becoming, de facto, providers of services going far beyond the simple physical provision of space for rental in warehouses: services such as Substitutive Optical Scanning, document indexing, manual data input and OCR (Optical Character Recognition) are now essential. In this context these companies often encounter difficulties facing up to problems more typical of the software engineering field: small several documental production is ever more dependent on their clients' internal business processes and integrated with their information systems. This makes the clients increasingly subject to the work of external software house to perform every, even little, modification to the implemented systems. In this paper we present a case study: the YouFile service prototype. This system supports smart document (both paper and electronic) indexing. It is based on a WebOS interface in order to benefit from the desktop metaphor, and it uses an ontological approach that defines not only the document classes but also the semantic relationships between specific characteristics of each document class. The system allows the management of the process of document indexing and retrieval in an automatic way, generating the data entry forms at run-time.

1. Introduction

By their very nature, paper archives increase in quantity rapidly and the concurrent problems in accessing these documents for consultation grow exponentially. This is an issue faced by every working organisation, be it a government agency, a large enterprise or a public body.

One way to solve this problem is to convert paper documents into electronic format: this solution brings several advantages in terms of space optimization, flexibility, availability and ease of retrieval.

When indexing paper documents, very often the quality of the original document doesn't allow the use of OCR techniques so these documents are converted into simple raster images without any possibility for the electronic retrieval of their content.

Similarly, even when documents are directly generated as electronic (digital images, word processor documents, files produced by specific software applications) they contain such large quantities of information that easy aggregation of them into groups of similar documents as well as their retrieval after storage is not possible.

A solution to this problem is to refer to classification systems which support the user in the categorization of documents. These range from professional systems used for example in digital libraries to the more 'personal' ones such as for example the file tagging system of Windows Vista.

In this conception of classification the user has to extract the information from the text of the original document and it will be linked (as metadata) to the image file, qualifying it in the database. This solution increases the additional conceptual and operative effort for the final user because s/he needs to specify unique information to link to the image of the acquired documents (for paper documents) and/or to the electronic ones.

To build a system of this type it is important that a design phase is included and used to define the 'indexes' of different types of documents. This phase is very delicate because it has to be performed 'a priori' and it is the essential condition for enabling optimal research inside the electronic documental archive.

Typically the phase of keyword definition is the job of the company providing the recording service and it is an onerous and at the same time not very rewarding activity. The possibility of storing in the system not only electronic documents (obtained through the scanner or directly produced by the user) but also paper ones opens up the market to include of all those customers who may require the storage of their paper documents in a company warehouse but who are unable to index them in a

autonomous way, defining their own search keywords, without the help of the personnel of the company.

In this context two problems arise: first of all it is important to provide the user with extensive memory space in order to store all the electronic documents. The second problem is to allow the user access to the system in order to simplify both the storage and the following retrieval of their document (independently of its form).

The opening of the system to the end user has, as a consequence, an increase in the number of possible documents that the people inside the document filing company do not know 'a priori'.

Following these we decided to work on a centralized system (realized as a web application) able to provide its users not only with a recording space in a 'virtual hard drive' fashion, but also to support and facilitate the autonomous performance of the delicate preliminary phase of indexing, even by less technically skilled users.

Starting from this idea, we present here the YouFile service: a real case study developed under the eSCI project (eStore of Captured Information).

2. eSCI project

The eSCI project is a research project led by Memar Monteassegni, an Italian company which has been working for more than twenty years in the field of paper and digital storage and GSA-Lab (Graphics and Software Architectures Laboratory) of the University of Salento.

One of the main goals of this project is to develop a Web portal, called YouFile, that will allow users to directly interact with the document filing company in order to electronically store digital documents or to activate its services related to paper document recording. The portal will provide tools for the auto-configuration of the user's archives, OCR and Information Capture services etc. Through the YouFile portal, the user will select the desired service of either electronic or paper.

The main expected result is to free the document filing company of all those phases such as customer searches and acquisition as well as the indexing of customer documents. Also, such a system would allow the customer, without the need for help from operators at the document filing company, to request a paper document storage service, to request document digitalization, to store electronic documents and to organize them into folders in order to retrieve them in a fast and secure way.

The eSCI project is divided into three main research themes. The first is connected to the definition of a storage hardware and software architecture able to support many electronic repositories and a great amount of e-documents; the second concerns the automation of the paper storage process, booked or activated through the web application, and the third regards the development of a system to assist web users in the creation of their own electronic virtual folders and in following definitions of the indexes for each e-document class.

It is the last theme, i.e. the development of assistive systems to help web users in the creation and maintenance of their digital folders, that has been the main target of the project. In fact, as previously said, the main difficulty in these systems is to achieve the complete characterization of every document, with appropriate keywords, in order to perform targeted, effective and fast searches.

3. Related works

As related works, we must consider two different aspects not related in themselves but both feature in the international scientific panorama necessary to set this problem against the correct background. The first aspect is related to digital libraries in order to solve the problem of 'indexing' and the second aspect is 'WebOS' to allow the user to access, easily and everywhere, the copies of their own indexed digital documents as well as the paper ones, even if the latter are only available in terms of their reference numbers in the company warehouse.

The project's basic idea is not too far from the concept of digital library [1], which originated in the sphere of biblioeconomy and has since widely developed over time.

Example of digital libraries include the Vatican Library or the Digital Library at the University of Michigan. In the international scientific panorama there is a rightful distinction between digital libraries and digital archives. Differently from the digital libraries, digital archives were conceived to hold digital documents directly generated by whoever performs their upload. Moreover, in digital archives the documents are stored in groups and not as single items: the user performing the file storage, the organization of its contents following the user's own logic. Grouping, for example, in different folders documents that share the same content. Another problem is the privacy issue: whereas in digital libraries the contents could be accessible also to user groups (even if restricted), in digital archives, typically, every user can access only his own content.

Both digital libraries and digital archives must face the problem of document indexing: it has to be performed in order to facilitate as much as possible e-document retrieval.

In digital libraries the indexing problem is solved through the definition of a set of keywords more or less specific to well-known specific topics of interest because, in most cases, the documents are limited to a specific and well known sector. In the digital archive field it is impossible to think in terms of well established keywords because the types of documents are diverse and each user

could define different keywords for the same document type.

The first idea of document indexing practice (either for single documents or groups) has distant origins: the first was the ‘faceted‘ classification system, devised in 1930 by the Indian archivist Ranganathan and which subsequently evolved as the indexing standard [2].

Nowadays pattern recognition techniques are often used [3] and they allow the performance of indexing starting from a first phase of knowledge extraction. Such techniques assume the availability of scanned documents with a resolution high enough to perform some OCR which is not the case for us, at least for the treatment of the paper documents (there may even be several handwritten documents for example).

The problem of document classification has been faced also from the discovery point of view: in [4] authors present a tagging system based on the face images where face images are extracted from the documents and the documents are tagged to the face images. This idea, although interesting, is more oriented to face recognition rather than the real indexing problems.

In recent times, personal digital documents (mainly photographs and video) have become able to be stored in several portals, for example YouTube for videos (www.youtube.com) or Flickr (www.flickr.com) for photographs.

The proliferation of these portals, following the Web 2.0 trend of content sharing, is not at all suitable for the storage documents containing sensitive data such as private and personal documents which a user would like to store in a secret and secure place.

Moreover, those indexing systems are based only on simple user self-defined tags. Even if in Web 2.0 applications the tag definition is a useful and powerful feature, it does not seem very usable for a system where one of the main goals is to assist users in the correct indexing of their own documents by providing a set of keywords that should be able to entirely characterize the document. A study of this phenomenon can be found in [5] where it is asserted that the use of the tag is not enough to characterize a document because tags, without the use of a domain ontology only allow classification based on a string and do not allow any type of reasoning. It is very important to see the use of ontologies as a key aspect to solve indexing problems: the use of tags related to a defined ontology opens the research up to web 3.0; the next target of the web [6].

As stated before, another work area very useful for our study is related to Web Operative Systems. One of the first studies of WebOS was carried out at the University of Texas [7]. The study was done in order to support geographically distributed application. The main goal of the WebOS idea is to define a new metaphor for desktop application [8]: all the operations that a user may make using his own computer will be translated on the web. The desktop metaphor is, in our opinion, the best paradigm to allow a generic user to access and manage his own electronic documents in the simplest way. WebOS provides the basic operating systems services needed to build applications that are geographically distributed, highly available, incrementally scalable and that reconfigure dynamically. WebOS simplifies system development and improves resource utilization. One of the major WebOS components very useful for our study is the possibility it offers to manage a wide area system that supports replication and wide-scale sharing.

4. Open Issues and motivations

One of the main problems for companies who have digital and paper filing systems is the complexity of document retrieval from their paper warehouse or from their digital archives: an issue that usually falls into the area of indexing.

In order to be retrievable, every document has to be tagged with some relevant keywords. The choice of these search keys depends on the specific document class under consideration. For example in an invoice, keys could be the legal name of the biller, invoice number, invoice date and so on.

When a new class of document is to be indexed the data storage company carries out one or more interviews with the customer who will use the system. This phase of analysis is oriented towards the definition of a model for each specific document class. This model is used to define the most important unique data for each document class in order to allow the company to retrieve the right document when its owner asks for it. Once keys are chosen for a particular class of document, every other document belonging to the same class will be automatically treated in the same way. Traditionally, the phase of association of the search keys to each document is performed through the support of a software application which has a certain set of data entry forms, developed ‘ad hoc’, one for each document class.

The complexity of these recording and indexing management applications is mainly due to the large amount of data entry forms that must be developed from scratch in order to characterize every different type of document.

This is reflected in:

- an overhead in the from of the maintenance of the archive management application which must be continuously evolved in order to update the set of supported document models with new document types that, periodically, need to be added;
- a cost to the data storage company as it is permanently tied to a software house for such maintenance;

- a weak reactivity with respect to the consumers, who often have to wait to index their documents if they belong to a new and not yet supported class, while the indexing application is being updated;

Although we can hypothesize forms of customized indexing, where the customer themself takes care of the search key definition of their own digital archives, this would generally produce two negative effects. First of all, the resulting proliferation of heterogeneous document classification systems would discourage any standardization. The second negative aspect is that the customer is not assisted in the generation and maintenance of his own archives, devaluing, de facto, the experience in the document management field that the company has laboriously consolidated over the course of the years. This experience needs to represent an added value of the entire system.

In order to face these two negative aspects, it is important that the information used for the data entry form generation is expressed as concepts with strict semantic meanings. The use of a knowledge base, and thus of the ontologies, seems to be the best means for the archiving of this kind of concepts.

5. Proposed solution

In order to discharge the enterprise from tasks that aren't value added such as the *internal* (to the data storage company) manual document indexing and the *external* development and management of the software applications supporting the indexing process, a better solution would be to directly allow the customers to index their own documents without any intervention by the company.

This process should be automatic and simple enough to also enable users who not very technically skilled to be autonomous. As stated before, WebOS and thus the desktop metaphor seem to be the most useful technology for our goals. In fact, since the profile of the end users of this system may vary from those with a little or no IT ability to those with a very high one, the software system that supports the process of document storing, indexing and retrieval should be as simple and standard as possible with the most traditional working mechanism, obviously in the form of a web application.

Why WebOS? The history of personal computing has so far been dominated by one-box solutions: a physical container holding all the software and hardware you need to run your applications. A desktop operating system such as Microsoft Windows, Apple's OS X and Linux KDE, provides a series of interfaces between the hardware inside the box and the software that is run (the word processors, graphic design tools, Web browser, etc.)

The WebOS vision explodes the very concept of operating system itself, moving beyond the physical bounds of a box and across the Internet. File storage is no longer limited to local disks, application logic to local processing on a CPU or access to documents to just one person at a time.

This vision seems to be the best for several reasons:

- Users can access their resources (multiple archives of paper or electronic documents), in the same way as they do already on their desktop computers;
- Users have a virtually infinite storage space, are provided with high security policies and the archives are reachable from everywhere and at any time;
- Companies can provide their clients with a WebOS application in order to let them manage their own resources without any external intervention.

However, to have a simple and traditional interface to access resources solves only half the problem. In order to be autonomous in the difficult process of indexing users have to be, in some way, guided by the experience and the know-how acquired through years of work by the data storage company.

In traditional document archive systems the information about the documents to be indexed is acquired through data entry forms which are statically generated. This feature represents an important design constraint because it is not very scalable in architectures that make use of many document types. It would be indeed extremely complex and expensive to develop new interfaces whenever required to define new document classes. Another limit of a static form of generation system is surely represented by the issues raised in the updating of already defined models. In fact, the simple modification of a field would imply the reprogramming of the whole interface and the database structure.

We advocate that the solution, in order to make the indexing systems more flexible and effective, could be:

- the use of a WebOS as virtual operative system hosting the resource management application;
- the use of ontologies to codify the company know-how about the best indexing strategy for each document class;
- the development of a proper interface generator able to dynamically propose to users the search keys most suitable to describe each specific document.

To validate our thesis we have designed and developed the YouFile personal indexing and storing prototype service. We face the problems both of the knowledge base design and of the experimentation of it through our development of the YouFile service prototype.

6. Knowledge base

6.1. Design methodologies

There are several methodologies in the international scientific panorama useful to design a knowledge base(Uschold & King methodology [9]; Gruninger & Fox methodology [10]; Methontology [11]). In this paper we adopted the methodology defined by Stanford University [12] which was conceived in parallel with the tool Protégé, an open source editor for ontologies that allows the representation of a knowledge base in a formal way and which represents ontology visually and obtains its formal representation in OWL [13] language. The concept steps foreseen by the Stanford University methodology are general enough and therefore easily adaptable to any particular application domain. The methodology starts from the definition of the motivations that bring an ontology to be designed (through several questions that the designer asks himself), and allows for the consideration of the possibility that an already existing ontology could be used and to generate a list of terms useful to define the ontology. From that list it is possible to select those useful to the conceptualization of the classes and to the conceptualization of the relationships between classes. The methodology suggests that each class be characterized by its attributes and relationships. It also suggests that a restriction of the properties useful to define, if there are any, is carried out. For example the cardinality or other characteristics of each property. The last step is to define individuals for each class.

The choice of the Stanford University methodology comes from two main motivations:

- The description of the methodology is helpful: this is thanks to its definition of several questions, to its definition of a list of terms and to the separation it makes between concepts and relationships. The methodology may be used as a step by step tutorial for knowledge base definition.
- The methodology was conceived in parallel to the Protégé editor. The editor and the methodology have the support of a passionate community of developers and users: the inventor of the methodology is, therefore, always ready to provide very useful suggestions when the designer experiences a problem in the design of some specific situation.

6.2. Why use ontologies?

One of the main goals of the project was to provide the user with a system to store documents in a way that follows, as closely as possible, the user's own way of reasoning in order to facilitate the recovery of their documents. To do this, it is important to reproduce the same semantic connections as those within which the user operates when storing his documents. To reproduce the relationships between concepts in the real world is at the base of the semantic web, therefore the definition of an ontology that can store, in a formal way, the semantic connections among the representative concepts of the several documents that the final user would like to archive, seemed the solution most suitable for our aims.

Our basic idea was to totally define a type of document both as regarding its intrinsic properties (title of the document, date of creation, author of the document and so on) and also in terms of some other data useful to better define the document. To take the example of a document that stores the personal data of someone: the intrinsic property of the document will be the title of the document (e.g. 'curriculum vitae of…'), the date of the document (e.g. '01/01/2008'), the author (e.g. 'Paolo Rossi'), the format used (e.g. 'European format') and so on. The set of attributes that characterize the person inside the document (such as name, address, experience and so on) is not information that distinguishes the type of document from other documents. Starting from this idea, two different types of concepts can be proposed: primary concepts that characterize each type of document and secondary concepts that may be common to several types of documents. It is clear that it is possible to link primary concepts and also to link between them secondary concepts. It is important to link together concepts defined as 'primary' with concepts defined as 'secondary' in order to characterize each document (Figure 1).

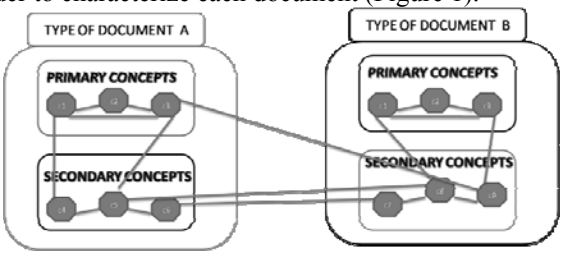

Figure 1 - Basic idea: primary concepts are specific to a document class while secondary concepts may be held in common by several document classes

A knowledge base defined in this way it is not only a simple 'data descriptor', but also a descriptor of possible semantic links between these data. The form generated starting from this knowledge base is very useful because it is a help for the user who, following the keyword defined in the knowledge base, may define his/her own document index structure both for the intrinsic characteristics and for the secondary ones that best define the document.

6.3. Definition of the knowledge base

It is important to define in detail how the knowledge base has been designed. In the design of the knowledge base for this work, we addressed two different problems:

- The knowledge base must represent all the concepts about a document class both primary concepts and secondary ones;
- The ontology is the base both for the form generation (where the keyword will be put) and for the database generation (where the keyword will be stored). Therefore all the information useful for the generation of the forms and/or of the underlying database, not just for characterizing documents, must be defined in it.

6.3.1. Ontological classes. The first idea in order to define the ontological classes was to devise all the possible typologies of document and, subsequently, to define as a data type property every attribute that characterizes the document. This solution was immediately discarded because it didn't solve the two problems previously defined. This solution does not define each attribute of the document in order to generate both the database and the forms. We observed that every attribute was characterized by different properties that define it such as, for example, the dimension and the data type: these attributes are very important for the on-the-fly data entry form generation. If we think, moreover, about the semantic relationships between specific characteristics of the documents, each attribute may be semantically related to other attributes and, if we define attributes as data type property, we cannot define these semantic relationships.

Starting from these considerations, we defined three types of ontological classes:

- *Documents*: subsequently divided into subclasses (primary concepts and secondary concepts). They define the possible typologies of documents that the system allows to be stored. The subclass 'Primary' defines several documents and the subclass 'Secondary' defines those concepts that do not characterize the document. Of course, well defined object properties will allow, where necessary, the semantically connection of Primary and Secondary concepts. For example, if the system allows to store 'Pictures', 'Fiscal Documents', 'Video', 'Curriculum vitae' and so on, these concepts will be subclasses of the subclass 'Primary'. Properties of these subclasses will characterize each type of document while properties of the 'Secondary' class will describe properties that do not characterize documents.
- *Attributes*: The class has several subclasses and defines all the possible attributes that can be individualized in the several typologies of documents examined. In turn every attribute is an ontological class that can contain other subclasses. An example of an attribute's subclass is 'date'. The class 'date' has its own properties (date format, allowed date, and so on) and it is possible to specialize this class with other subclasses in order to define, for example, special types of date (for example to separate Italian format from American format). An important characteristic that is possible to define when we define each attribute is the priority level. The documents inserted by the user will be stored in the WebOs in a specific folder and the user may retrieve their own documents just as they would retrieve a document in their hard disk. The documents will be stored in a specific folder hierarchy and the level of the hierarchy will be defined as a special characteristic of the attribute. This will be done by assigning a 'priority level' to the attribute. If an attribute has a priority level value of 1, the root of the folder where the document will be stored has the same name as this attribute.
- *Support*: this defines all those concepts that are not tightly bound to the document but which allow to better specify its characteristics. For example if we want to specify that the attribute 'date' will have only two possible dates from which user may select, these two available dates are instances of the subclass 'allowed_date' of the class 'support'.

6.3.2. Ontological properties. We defined three different types of properties:

- The object property that links together each type of document and each subclass of the class Attributes. The name of these properties is '*has_attributeName*' and they have as their range the class that represents the attribute previously defined. For example, if the document '*curriculum vitae*' has the field '*date*' the class 'curriculum vitae' will have, among its properties, '*has_date'* and the range will be the 'date' subclass of the class Attribute.
- The object property that links together primary and secondary concepts. Secondary concepts can group together several attributes or concepts in order to better define (and through properties that do not characterize the document) a document. For example if to describe a 'curriculum vitae' it is important to add information about the person and the 'person' is a secondary concept, in the class 'curriculum vitae' there will be the object property that links to the 'person' concept.

- The data type property that characterizes in a specific way each attribute of the document. For example the data type, the length etc. of each attribute will be property of the class that is a subclass of 'attributes'.

In the definition of the ontological properties we also used the '`owl:hasValues`' restriction. This restriction allows the definition of the value that a very specific property assumes in a well defined context. We use this restriction to define the properties that characterize each attribute (length, type of data and so on).

6.3.3 Individuals. The Individuals was used in this work in order to add a semantic layer to the concepts defined in terms of classes and properties. Individuals was added in the 'support' classes. For example, if for a specific property the user may define only two (or more) values, these values will be defined as individuals of the property: the data entry form automatically generated will provide to the user the possibility to select one of these values thereby avoiding error.

6.4. The persistence layer

The knowledge base is the starting point for the form generation but the problem remains of where to store information about documents inserted by the final user?

We analyzed three different hypotheses of work.

The first hypothesis was to make the ontological representation of all possible concepts that a user may define for a specific document. To do so, it is important to define, for all the document classes individuated, as many subclasses as possible documents that a customer would like to insert. For example if we consider the concept of '*WhiteWine'* it is possible to define it by adding the individual 'White' to the 'colour' property of the class '*Wine*'. The concept '*WhiteWine'* with the property Colour 'White' is a subclass of the class Wine. In the ontology definition, the focus, using this approach, is on the concept of '*WhiteWine'* that, in the knowledge base, is defined clearly. Using this strategy would mean to think and design in advance all the possible cases for every type of document and this would be very hard work.

The approach has been discarded for two main reasons:

- We plan to manage many documents so the number of subclasses to define would be excessively high.
- For each type of document several special requirements may exist for each user (for example to add or delete some field or to define a specific field better). To foresee in advance all the possible requirements of the users is impossible, so the situation would be that the user would often require the data storage company to update the ontology. This on the one hand increases the work load for the operator and on the other could easily lead customers to abandon the portal when their needs are not immediately satisfied.

Another hypothesis was to define in the knowledge base all the concepts and the semantic relations between them (related to well defined document classes). When the user adds their own documents to the portal, the specific information about the document will be an individual of ontological classes and/or property.

Using this hypothesis the focus moves from the concept to the metadata: the ontology will be used not to describe the specific typology of document that the user wants to store but the metadata that allows the user to add their own typology of document. In other words, the ontology defines a basic structure that guides the user in classifying and in retrieving their own documents.

If a user, for example, wants to store photos related to a specific landscape, the user will add an individual to the concept of photos and of landscape which are related to each other by a semantic relationship. If another user wants to store another photo that records another landscape he/she will use the same concepts but with another meaning.

Through this approach, each time that the user adds a new document, there will be a new individual in the knowledge base: the persistence will be in the knowledge base.

Another hypothesis was to define a scheme that allows the semantic characterization of the documents and to define a layer of persistence inside the database. When the user adds their own documents to the portal, the specific information about the document will be an individual of ontological classes and/or property. In an optimistic vision, if the YouFile portal were to become heavily accessed and used, this would bring about the creation of a large knowledge base but, to date, the technology is not ready to allow the management of very large ontologies and, at the same time, it is not possible to manage the security access to the knowledge base. These issues are however well known and supported in the database field. For these reasons, the solution chosen has been to define a scheme that allows the semantically characterization of the documents and the definition of a layer of persistence inside the database.

Thus the adopted solution has been to combine the undisputed advantages attached to the definition of an ontology with the advantages springing from the use of a database that will constitute, therefore, the layer of persistence. In this way both the syntactic and semantic expressiveness of the knowledge bases and the technologies already consolidated within the database field to manage massive structures of data can be fully exploited.

6.4.1. From the knowledge base to the database. To realize the persistence layer we define a simple algorithm

that allows the automatic generation of the database where the individuals that the final user adds to the system are stored.

- Each subclass of the primary concepts and each subclass of the secondary concepts is a table in the database: each record set in this table has its own primary key.
- Each ontological class that is a subclass of the ontological class 'Attribute' is a field in a table (either primary or secondary). Each field is defined following the property of the related ontological class (for the type of data, the length of the field and so on).
- The 'functional' object properties we define in a table of the database that represents the range of the object property the primary key of the table of the database that represents the domain of the object property (the idea is the same as that of the mapping between ER model and relational model for the 1:N relationship).
- For the other object properties (not functional) we define a new table that represents the object property and it takes, as key, the key of the domain and the key of the range of the relation (the idea is the same as that of the mapping between ER model and relational model for the N:M relationship).
- The concept subclass of the class 'Support' does not have a related concept in the database: these subclasses will be used only in the engine for the generation of the form in order to obtain specific information of interest.

The names of the tables and of the attributes are the same as those of the corresponding concepts and properties. This strategy foresees, obviously, the synchronization between knowledge base layer and the persistence layer: each change in the ontology must be reflected in a change in the database.

6.4.2. Knowledge base update: methodological guidelines. Using the designed and implemented portal it is possible that, in order to answer to new user requirements or to new company needs it is important to update the realized knowledge base. The update must regard several aspects and, for each of them, we provide some methodological guidelines to update the knowledge base.

- *Add/delete an attribute to an existing document*. To add an attribute to an existing document in the knowledge base predicates a first step of analysis of the knowledge base to ascertain that the same attribute has not been defined for some other document. In this case, it will be sufficient to analyze the attribute and, if it has the same characteristic of the already defined attribute, it is possible to reuse the same object property that links the already existing attribute to the document. If the attribute to add has different characteristics it is important to realize a subclass of the attribute adding the specific restrictions.
- *Add/delete a semantic relationship between a document and a new or already existing secondary concept*. It is possible that, to assist the user in the definition of his own document, is helpful to add information related to the document that does not characterize the document but that represents only some way of reasoning employed by the user. This information will be searched in the ontology and, if it is not already defined, this is inserted as subclass of the class 'secondary'. It is important to define the class that defines the concept and an Object Property will link the document with the secondary subclass just created.
- *Add a new typology of document*. To add a new typology of document it is important to follow different steps:
  - To create a subclass of the class 'Document' with the name of the type of document to add. The subclass may be or a subclass of the 'document' that is at the same level as the classes that have other documents as subclasses, or a subclass of a class that has other documents as it subclass.
  - To link the class defined in the previous step with already existing attributes (see add/delete an attribute to an existing document).
  - If the document has several attributes that are possible to group together, it is necessary to add a subclass of the ontological class 'SecondaryConcept' where these several attributes can be defined.
  - If the document has properties that do not characterize the document but that are useful to the user to provide the document with a semantic meaning, these properties must be defined as a subclass of the class 'SecondaryConcept'.

6.4.3. Database update. For each change to the knowledge base, there must be a corresponding change to the database. The change must on the one hand protect information already in the knowledge base and on the other hand must allow the insertion of the new information according to the new knowledge base. The update of the knowledge base must consider several possibilities:

- *Add an attribute to an already existing document*: the only thing to do is to add the attribute to the table related to the document;
- *Add a semantic relationship between a document and a new or already existing secondary concept*. When the secondary concept is new, it must create a new table to represent the concept. The table that

represents the document and the table that represents the secondary concept will be linked together through a 1:N or an N:M relationship. The cardinality of the relationship depends upon whether the object property is functional or not functional;

- *Add a new typology of document*. A table must be added in the database with all the attributes that characterize the document.

A specific software tool has been developed to detect the differences between two versions of the knowledge base and to automatically update the database in order to align the two information sources (knowledge base and database).

7. Experimental implementation of the YouFile service prototype system

7.1. System architecture

As previously stated, we selected WebOS as the architectural paradigm to inspire our system. Among all the reviewed implementations of either commercial or academic WebOS, the one that seemed most stable and complete was eXo Enterprise WebOS by Object Forge Web (http://www.exoplatform.com/portal/public/en/).
The reasons that lead us to choose eXO were:

- eXO is an Open Source project.
- eXO is developed under J2EE, so it guarantees the future scalability of the entire system.
- eXO is a Portlet Container, so every web application can be developed as a Portlet compliant with the JSR168 standard.
- eXO is not only a WebOS but also a Portal Server, so it can also be used to host the front end traditional web site of the company.

In the eXo platform all the business logic is encapsulated in services that are dependant but loosely coupled thanks to Inversion Of Control (IoC). Therefore, each product, as shown in Figure 2, is composed of a set of services, portlets that query them and one or several portal instances that are simple web applications (war) with dedicated configurations and web designs (each portal instance can define its own preconfigured organization model or security policy as well as many other configurations such as the predefined portal template pages to use when new users are created which can be useful for hosting environments). The service container layer is responsible for gluing the services. To customize the specific WebOS selected, a new layer called OMS (Ontology Management System) has been developed. Figure 2 represents the YouFile system architecture in the large; basically it is the infrastructure of the eXO platform to which have been added some modules (boxes with continuous dash lines).

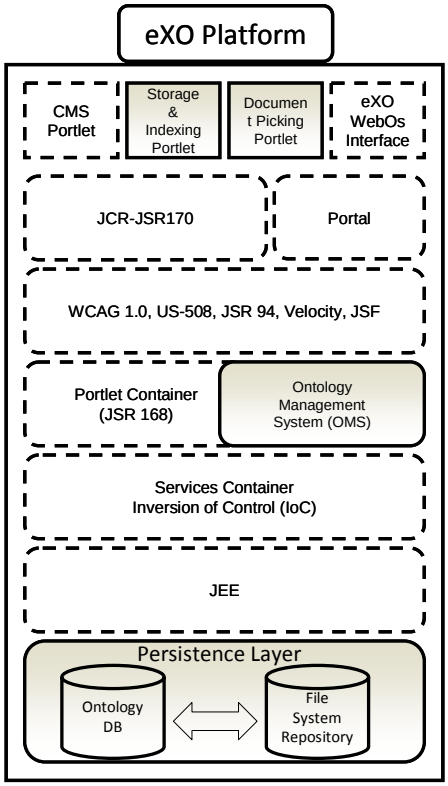

Figure 2 - YouFile software architecture

There are two portlets developed from scratch and deployed under eXO:

- *Storage and indexing portlet*: is the module that, based on the concepts stored in the document knowledge base, guides the user to choose the most efficient search keys to index their own document. The same search keys can be subsequently used to retrieve the correct stored documents.
- *Document picking portlet*: this is the module supporting the process of the paper documents' physical storage in the company warehouses. It is organized as a wizard.

Besides the ontological search another more immediate way to access indexed files is the proprietary File Explorer module of the eXO platform (Figure 3). Through this interface every user needing to view and/or download their files, has the opportunity to navigate through the virtual file system provided by the WebOS system, in the very same way they do at their own desktop computer. As shown previously (in Figure 2), the OMS module is partially overlapped by the Portlet Container. This means that a part of it, in particular the front end of the visual generation of the data entry forms, has been developed as a portlet. The remaining part, indeed, as the database and its business logic, file

repository (see at the bottom of Figure 2) are directly integrated inside the eXO platform.

Figure 3 - eXO portal file explorer

In Figure 4 the interaction between the OMS layer and the persistence layer is shown. Here we find two main subsystems:

- *Storage repository of document classes*, developed through the ontology language OWL (ontology web language): the document class descriptions are stored here in an XML file over the server file system.
- *Persistence layer*: Here are stored concepts (every keyword's value selected by the user to fully characterize his document) in the Ontology DB and electronic documents in the File System repository.

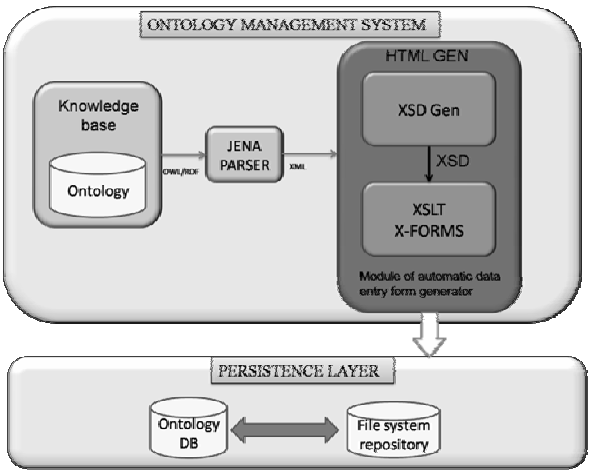

Figure 4 - OMS and persistence layer interaction

One of the main problems of this approach is the need to maintain the synchronization of the knowledge base of the document models and the ontology. This synchronization is important because every concept (expressed in the knowledge base layer) has to be translated in one or more tables inside the database.

7.2. Use case

In this paragraph we present a use case relevant for this system: the indexing of a document (see Figure 5). The core of the architecture, as said, is the knowledge base and it is stored as an OWL file on the file system. Once the user selects the document type he/she wants to upload and store (message 1:), the Jena Parser module will search the ontology for all the attributes/fields needed to describe this document class (message 3:) and then it will extract a subset of all the ontology concepts (message 4:), that fully represent the desired document type, writing it in a temporary XML file (message 5:). From this file the XSD Gen module will generate the corresponding XML schema file, containing all the attributes and their allowable value ranges (message 6:). Then, through an XSL-T transformation a XFORMS 1.1 document will be generated, and it will be loaded in the browser (message 8:). Finally, the user will input the desired values (the index keys) in the interface just loaded (message 10:) and they will be stored in the database being associated to the corresponding file (message 12:).

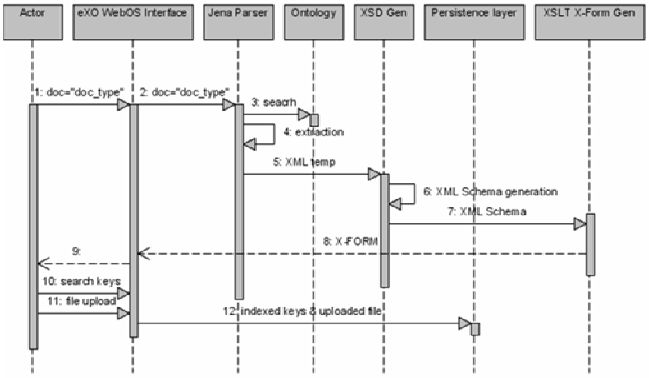

Figure 5 - Document indexing sequence diagram

It is important, now, to pay attention to a complete use case. We consider the storage and indexing portlet of the system and we show, using several screenshots of the developed system, two cases: the first is the storage (and retrieval) in the system of a document already defined in the knowledge base, the second is the storage of a custom document that is a new, user defined document.

In the first use case, we consider the type of document 'picture', and we define the overall process that starts with the selection of the type of document that the user wants to store and ends with the upload of the picture file. The characteristics of the picture are already present in the knowledge base so the user must select the typology of document and add his/her own data in the specific field. Clearly, the description of the data comes from the knowledge base. We take a look at the knowledge base, and we highlight the most important concepts that define a picture and how they were modelled.

We suppose that the picture is characterized by the year, the file name, the period of the day, the scenario, the recurrence, the location where the user took the photo, the orientation, the camera used, the colour of the picture (colour or black and white). It is clear that in the listed concept the file name, the year, the colour, the period of the day and the orientation characterize the picture, whereas the other concepts do not characterize the picture but they are very helpful in order to allow the user to remember their own keywords.

To define the part of the knowledge base for picture, first of all, we define as a subclass the primary concepts the ontological class ‘Picture' that has in its properties ‘hasFileName’, ‘hasYear’, ‘hasColour’, ‘hasOrientation’ as well as the properties ‘hasScenario’, ‘hasRecurrence’, ‘madeByCamera’, ‘hasPlace’. The concept of ‘Place’, for example, is a secondary concept and it is a subclass of the ‘SecondaryConcepts’ class: the object property ‘hasPlace’ has as a range the ‘Place’ concept. The ‘Place’ concept has four attributes: ‘Province’, ‘Nation’, ‘PlaceInTheCity’ and ‘City’ that refer to the relative subclasses of the class ‘Attribute’. The subclass of the attribute class ‘Year’, for example, has as a property ‘hasMaxLength’ and ‘hasTypeOfData’ in order to better describe the year name attribute. The ‘Year’ attribute has, also, a priority level defined with the restriction ‘hasValue Priority 1’. This means that the pictures will have the year as root of the folder.

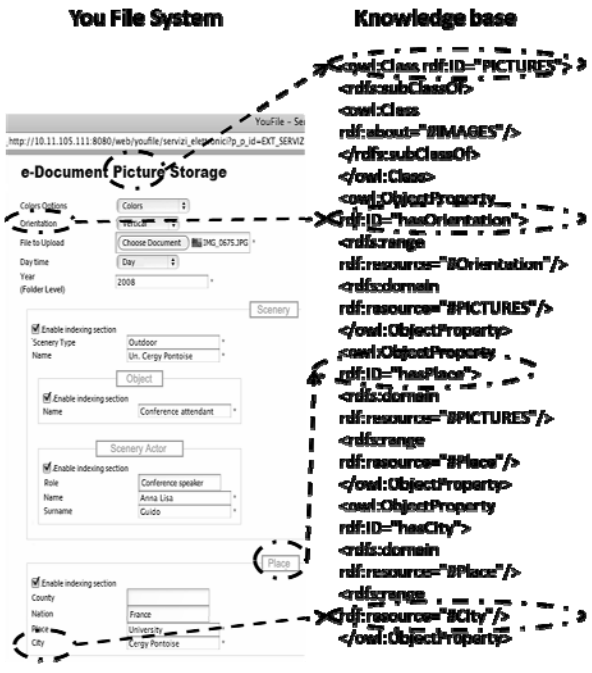

Figure 7 A screenshot of the YouFile system for the document ‘picture’

The secondary concepts here listed may be held in common with several typologies of document: we can conceive of a drawing or a screenshot or of another image that a user may want to archive. They are defined one time in the knowledge base and used again if this is necessary. In Figure 7 we present a screenshot of the form obtained starting from the knowledge base that describes the document ‘picture’ and the OWL code related to the concept Picture.

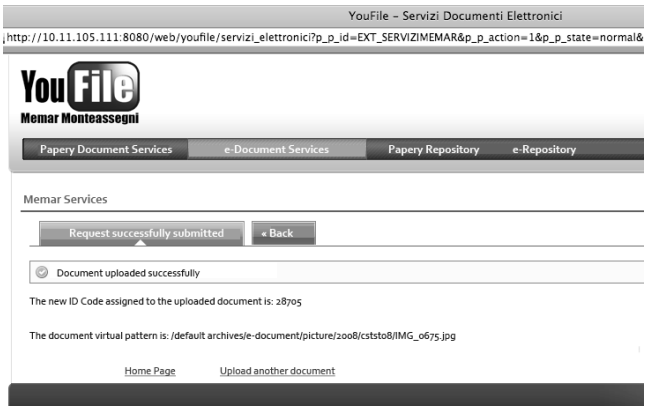

Figure 8 Confirmation

We highlight that the priority level 1 is defined for the field year. This means that the root of the folder will be, in this case, ‘2008’.

The user fills in the form and uploads the document by selecting the file from his own hard disk. The system confirms the upload (Figure 8). Starting from this moment the picture is in the virtual hard drive of the data storage company. The user may search for his/her document.

To search for the document, the user has two possibilities. The first is to search for a document in the virtual hard drive. The user will find the document in the folder following the priority level defined in the knowledge base. Another possibility is to search for the document by the keywords that the user inserted in the system when he/she uploaded the document. If the user wants to retrieve his/her pictures that have, as type of scenario, ‘Outdoor’, he/she must select the type of document to search and then he/she must type ‘Outdoor’ in the specific field (Figure 9). The system will find the document.

In this case, the system finds two pictures that have the type of scenario ‘Outdoor’ (Figure 10). If the user clicks on the link with the specific path, s/he can download his/her pictures.

We consider, now the possibility of the upload of a document that has not been defined in the knowledge base.

The user must select the option ‘Store a custom document ‘and the system will drive the user in the definition of his/her own document. The user will add the

document name (in Figure 11 ‘curriculum vitae’) and will define each name of the field that he/she wants to add. For each field, the user must define the name, the type and the folder layer.

Figure 9 Research

This information will be added in the knowledge base: the name of the document is a new primary concept subclass of the class ‘Document’. At present the system does not allow the adding of a secondary concept (this is a future work) but it does allow the definition of all the attributes that characterize the documents. All the fields that the user adds will be subclasses of the class ‘Attributes’ with the properties ‘typeOfField’ and ‘priorityLevel’ with the range defined by the user in the form.

Figure 10 Results

At this point, the user may select the new document that s/he has just defined and may store it in the system. The knowledge base of the user will be updated with the new document: this document will be shown only to the user that has added it; only if the company wants to share it to the public knowledge base will the new document be made visible to all the users.

The system, in an experimental phase, will be online soon.

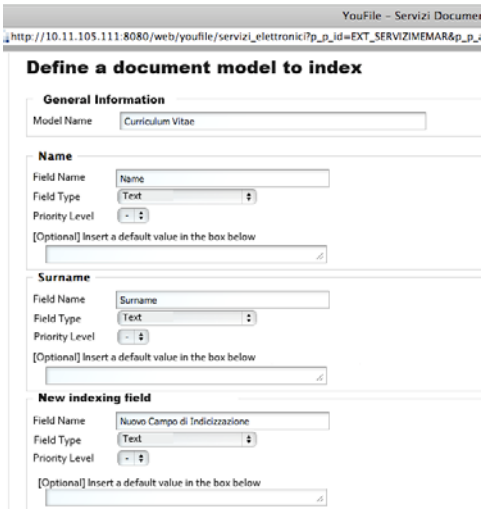

Figure 11 Definition of a new document

8. CONCLUSIONS

The large quantity of paper or electronic documents that every company manages in its daily business processes raises the problem of their storage (either physical or digital) in order to be able to retrieve them when necessary. A solution to this problem is to convert paper documents into electronic documents in order to simplify the process of indexing. This is a problem not only holds for companies but also for private citizens who may want to store both printed paper and electronic documents that can be of several types such as photos, video, word processed documents and so on.

The idea of the present research work is to provide a system able to store both paper documents and electronic documents and to guarantee easy research of what has been archived. This tool must be able, for the paper documents, to define the keywords strictly related to the document that the user wants to archive and, for the electronic documents, beside the possibility to define keywords, also to allow for the possibility to store the document in a ‘virtual hard drive’. This means the opening of the system not only to people who work in the companies that provide the service of filing but also to private users and thus increase the number of possible document types and consequently the number data entry forms needed to index them. The architecture proposed in this research work which makes up a part of the eSCI project considers two main aspects:

- the need to generate on the fly the data entry forms;
- the possibility to provide a ‘virtual hard drive’ to the user in order to access their documents just as they would do on their personal computer.

For the first aspect we defined a knowledge base and so, with the OMS layer of the architecture proposed, the system is able to generate on the fly the data entry form. For the second, the provision of a virtual hard drive, we use the WebOS technology. The use of the knowledge bases for the representation of the documents allows the know-how already acquired by the company in the field of the document indexing to be made explicit, avoiding, in this way, its fragmentation among the several divisions specialized in a particular, specific set of documents. The proposed architecture allows, moreover, the transformation of the know-how acquired by the company in a service that the company offers to its customers. Thanks to the formal and precise description of each document that the company allows to be stored, the data entry forms help the end user in the indexing of his/her own documents without any need for the intervention of the data storage company's personnel. The use of the WebOS technology allows the provision of a virtual unlimited memory space where a user may upload his/her files after the indexing phase. In the WebOS, through portlet technology, all the business processes of the company may be outsourced to the final user.

The project, currently, is under a test phase. The main problem that has been raised during the experimentation is that the tools to manage ontologies are not very efficient regarding performance and so, the on the fly data entry form generation was too slow. In the future work, it should be useful to pre-generate data entry forms, starting from, and maintaining synchronization with, the knowledge base in order to avoid their regeneration every time the same form is requested.

9. Acknowledgements

We would thank the Memar Monteassegni Company and Tommaso Mezzina for their tangible support, and WebScience SRL for their help in the experimentation and testing phase. Moreover, particular thanks to the Apulia Region that founded the eSCI project.

10. References

[1] Greenstein, D., and Thorin, S.E., 2002, "The Digital Library: A Biography," Digital Library Federation .

[2] Broughton, V., 2001, "Faceted classification as a basis for knowledge organization in a digital environment; the Bliss Bibliographic Classification as a model for vocabulary management and the creation of multi-dimensional knowledge structures," The New Review of Hypermedia and Multimedia, pp. 67–102.

[3] Berardi, M., Lapi, M., and Malerba, D., 2004, "An integrated approach for automatic semantic structure extraction in document images," In S. Marinai & A. Dengel (Eds.), Document Analysis Systems VI., 6th International Workshop, DAS 2004, Lecture Notes in Computer Science, pp. 179–190.

[4] Vikram T.N, Shalini R. Urs, and Chidananda Gowda , K., 2008, "Person Specific Document Retrieval Using Face Biometrics", Digital Libraries: Universal and Ubiquitous Access to Information Vol. 5362/2008, pp. 371–374

[5] Zhao N., Fang F., and Fan, L., 2008, "An Ontology-based Model for Tags Mapping and Management," In Computer Science and Software Engineering, 2008 Vol. 5, pp. 483–486.

[6] Hendler, J., 2009, "Web 3.0 Emerging," In Computer Vol. 42 (1), pp. 111–113.

[7] Vahadat, A., Eastham, P., Yoshokawa, C., Belani, E., David Culler, T., and Dahlin, M., 1997, "WebOS: Operating System Services for Wide Area Applications" Technical Report: CSD-97-938 of EECS.

[8] Weiss, A., 2005, "WebOS say goodbye to desktop application," Networker, Vol. 9 (4) pp. 18–26.

[9] Uschold, M., and King, M., 1995, "Towards a Methodology for Building Ontologies," In Workshop on Basic Ontological Issues in Knowledge Sharing, held in conjunction with IJCAI-95, Montreal, Canada.

[10] Grüninger, M., and Fox, M.S, 1995, "Methodology for the Design and Evaluation of Ontologies," In Proceedings of IJCAI'95, Workshop on Basic Ontological Issues in Knowledge Sharing, Montreal, Canada.

[11] Corcho, O., Mariano, F., Gómez-Pérez, A., and López-Cima, A., 2005, "Building Legal Ontologies with METHONTOLOGY and WebODE," In Benjamins, R.; Casanovas, P.; Breuker, J. and Gangemi, A. (ed.): Law and the Semantic Web, Springer-Verlag 3369, pp. 142–157

[12] Noy, N. F., and McGuinness, D., 2001, "Ontology Development 101: A Guide to Creating Your First Ontology," Technical Report Stanford Knowledge Systems Laboratory, Stanford Medical Informatics Technical Report.

[13] W3C OWL Web Ontology language Reference W3C, 2004.