A Detailed Description of the EC^{2}M Project: Exploiting Ontologies for the Automatic and Manual Documents Classification in Industrial Enterprise Content Management Systems

Daniela Briola
DIBRIS, Genoa University
Via Dodecaneso 35, 16146 Genoa, Italy
daniela.briola@unige.it

Alessandro Amicone
GFT Italia S.r.l.
Via Cesarea, 2/45, 16121 Genoa, Italy
alessandro.amicone@gft.com

***Abstract*—Enterprise Content Management (ECM) systems represent a crucial aspect in the efficient and effective management of large-scale enterprises, in particular for those made up of several sites distributed all over the world. The increasing number of documents to be managed, the problems related to the sharing of private information between commercial partners, the need for semantically describing the contents of shared documents have pushed researchers to find new techniques and solutions to deal with these challenges. We already presented the high level description of a joint project of the Department of Informatics, Bioengineering, Robotics and System Engineering of the University of Genoa, Italy, and two companies, Nacon (member of Sempla Group, now part of the GFT Group) and Nis, to create an improved ECM system (named EC^{2}M) exploiting ontologies to better classify, retrieve and share documentation among different sites of the involved companies: in this paper, we give a more detailed description of the project, with respect to its modules and to the underlying ontology used to classify documents. We present the automatic documents classification algorithm too, with an example of its execution. The developed system, which was born from a real industrial need, is currently used by GFT Italy to manage and share its documents among more than 600 users distributed in many different geographical locations and, thanks to the ontology, the semantic tagging process and the automatic documents forwarding have been successfully achieved. This joint project proves how a more formal representation of the documents domain can effectively improve the standard way of classifying and retrieving documents in real industrial scenarios, representing a winning collaboration between university and industry.**

Keywords– Ontologies, Semantic Classification, Knowledge Representation, Industrial Application, Automatic Documents Classification.

I. INTRODUCTION

This paper presents in detail the "Enterprise Cloud Content Management (EC^{2}M)" system, that was previously described in [1]: it is an "Enterprise Content Management" system deployed over a cloud platform, improved with the capability of semantically tagging the documents using an ontology and exploiting context information from mobile devices. This paper is an extended version of the previous one, and presents new information about the architecture and the implementation of the system modules, about the underlying ontology and about the classification algorithm used to automatically tag documents. Figures and results have been updated too, to reflect the actual running system.

The international Association for Information and Image Management (AIIM), the worldwide association for enterprise content management, defined the term "Enterprise Content Management" in 2000, but it has been updated many times to adhere to the continuous new market needs. The more recent definition is: *Enterprise Content Management (ECM) is the strategies, methods and tools used to capture, manage, store, preserve, and deliver content and documents related to organizational processes. ECM covers the management of information within the entire scope of an enterprise whether that information is in the form of a paper document, an electronic file, a database print stream, or even an email* [2]. ECM is an "umbrella term" covering document management, web content management, information search, collaboration, records management and many other tasks, but it is primarily aimed at managing the life-cycle of information, from initial publication or creation to its disposal, to preserve a company's internal (often unstructured) information, in all of its forms.

Therefore, most ECM solutions focus on Business-to-Employee (B2E) systems, but nowadays, thanks to the improvement of the IT capabilities and because of the increasing users' need to classify documents according to their meaning, these systems have grown in complexity and often integrate modules to exploit more structured information, taxonomies, dictionaries and so on.

This trend is identified both in the industrial area ([3], [4]), where the focus is usually on improving already existing products and on increasing their usability, efficiency and functionalities, both in the academic field, where the focus is more on studying new knowledge representation formats and their exploitation in automatic data analysis, classification and storage for automatic reasoning or user centric services (see next Section).

Many vendors are offering products in this area, starting from the commercial ones (Microsoft, IBM and Oracle) moving to many powerful open source solutions (for example

Alfresco [5], Plone [6] and SenseNet [7]). The new trend in this field is to create ECM systems that can automatically extract information from documents to classify them or add a semantic layer to tag documents in a more structured and interesting way: this is the area where the EC^{2}M project is located.

The problem of classifying, retrieving and sharing documents among users and companies pertains to the research field of knowledge sharing, whereas the problem of semantically tagging documents pertains to that of the knowledge representation. Both fields are relevant both from an academic and an industrial viewpoint, and this motivates the joint academic-industry EC^{2}M project. In EC^{2}M, we used ontologies as a way to structure information describing documents and their content.

Many similar studies and projects have been conducted in this area: an example of a commercial ECM system semantically enriched is SmartLogic [8], which offers an automatic classification, based on an automatically-extracted taxonomy of documents. Many open source systems have been developed to integrate semantic services in the document/information management, among which H-DOSE [9], OPEN-CALAIS [10] and APACHE STANBOL [11]. Even if they are not ECM systems according to the standard definition, they deal with very similar problems.

We decided to exploit for our system some of these open-source systems (as described later), to be able to freely combine them to get an improved ECM system. None of the available systems offered a complete solution to our problem, so we adopted a mesh-up approach, based on open source softwares, and we then integrated in it an ad hoc ontology, shared among the different nodes of the network, to model the documents types and their content.

The system offers a publish/subscribe service and is based on a cloud platform. It also takes exploits contextual information on the location and device used by the user to implement context-awareness. In this way, the resulting system takes advantage of well known and high quality open source softwares and of a powerful cloud platform, and enriches the available solutions with new techniques not used in standard ECM systems.

EC^{2}M is thus a concrete industrial-academic example of how these technologies can be composed and improved to create a new powerful system, which can be actually used by enterprises.

The rest of the paper is organized in six sections: Section II presents the state of the art regarding similar academic systems, Section III describes the EC^{2}M system, Section IV presents the ontology, Section V shows the actual implemented system, Section VI describes the automatic classification algorithm used in the system and, finally, Section VII concludes the paper.

II. State of the art

If we look at academic research, many studies underline how a semantic and structured representation of the domain (with ontologies or similar techniques) can improve systems like CMSs, where documents and data must be stored and classified, manually or automatically, so that users can easily find what they are looking for.

For example we can cite [12], describing the Rhizomer CMS, which tags its items using semantic metadata semi-automatically extracted from multimedia sources, or [13], which proposes a framework to manage and share written information contents using an ad-hoc knowledge model for an industrial research center, or [14], which presents an open architecture framework based on the open-source CMS OpenCMS and a Java-based web management system for learning objects, which were derived from the instructional materials used in several postgraduate courses.

More recently, many other analyses and examples have been realized.

In [15], a set of tools have been developed to semi-automatically explicate the semantics of a content repository into a knowledge-base and to establish semantic bridges between this knowledge-base and the content repository (the tools set is complemented with a search engine that makes use of the explicated semantics). In [16], a semantic-based content abstraction and annotation approach is proposed: based on this approach, a semantic-driven content management environment has been developed to deliver the right content to the right user at the right time. According to the authors of [17], dealing with problems and possible architectural solutions in managing heterogeneous oceanographic data are reported, a careful employment of ECM systems may be beneficial in that setting, with no need to adopt complex ad hoc solutions that are difficult to maintain by personnel not specifically skilled in data-handling techniques. A data model to support the storage of refined data in structured repositories is developed and presented in that paper as well. In [18], the described approach is to model context (the public documents of a Public Administration) in an ontology and to use that ontology to infer content-related metadata to be associated to the documents, avoiding to do this operation manually.

Our project is mainly focused on the knowledge representation and sharing research areas, but it takes into account software design problematics too, like those found in user-centric and context aware systems development: many academic research works dealing with these topics exist, proposing different solutions and approaches.

For example, in [19] the Multiagent paradigm is used as underlying architecture to develop distributed intelligent ubiquitous systems where applications and services can communicate in a distributed way with intelligent agents, even from mobile devices, while in [20] the authors present a framework to develop context-aware dialogue systems that dynamically incorporate user specific requirements and preferences as well as characteristics about the interaction environment, in order to improve and personalize web information and services. In [21], a distributed architecture called inContexto, which uses mobile phones, is used to infer physical actions performed by users starting from user context information. Starting from the assumption that the human context within which a software system will operate is fundamental for its requirements, in [22] a framework based on the socio-psychological Activity Theory and its analysis of human contexts is presented.

We also found systems that exploit ontologies to improve knowledge sharing, but dealing with domains that are completely different from ours (for example [23], which is a semantic television content management system based on ontologies) or relying on different architectures (for example [24], where an Ontology Server (OS) component is created to be used in a distributed content management grid system): even if the domains or the proposed solutions are different, the underling problem still remains the same, proving that it is still open and studied.

III. THE EC^{2}M SYSTEM ARCHITECTURE

The Enterprise Cloud Content Management (EC^{2}M) system was born from the collaboration among the Department of Informatics, Bioengineering, Robotics and System Engineering of the University of Genoa and two outstanding IT Italian enterprises, namely Nacon (member of the Sempla Group, now GFT Italy, specialized in the design and implementation of complex systems, ECMs, and process management) and Network Integration & Solutions (Nis), specialized in the design and development of network products and services for businesses, public administration and end-users.

The developed system is a Content Management System that aims to automatically classify documents with respect to an ontology defining the possible predefined tags (and, at the same time, to help users in semantically tagging documents that they are manually inserting in the system): these documents will be then shared among different partners (called "Nodes of the EC^{2}M network", which are companies or companies' sites, which need to collaborate and agree on using the common ontology to tag documents), located in various physical locations, in an automatic way.

The types of documents and their possible contents are modeled using an ontology that formally describes them; the ontology instances are used to tag the documents with semantic information. Every user in the system is able to subscribe to a set of "interests" (chosen from the instances in the ontology) so that when a new document is inserted in the EC^{2}M network and tagged (manually or automatically) with terms from the ontology, those users that are interested in those terms are proactively informed that a new document is available.

The routing process, which in our case is the process of informing the Nodes in the EC^{2}M network about the existence of new documents and of consequently sharing them, is demanded to a Central Router Node.

A software module manages the context (location and device) where the user is acting, to give the user a subset of the information he needs considering the device he is working on.

The system is deployed over the Cloud Amazon Web Services (AWS) platform (using it as an "Infrastructure as a Service"), which is a good compromise between cost and performances. This solution allows to simply scale the number of nodes in the EC^{2}M network or to scale the physical resources used to manage the network, to get better performances, and at the same time grants a reliable Central Router implementation, avoiding the standard "single point of failure" problem related to the centralized routing architecture.

Anyway, the EC^{2}M system has been designed to run on a private physical network too.

The EC^{2}M system offers "internal services" to individual nodes (corresponding to an enterprise site) and "external services" to let nodes interact. Looking at individual nodes, the so called "semantic publish/subscribe" service allows every user to declare the arguments he is interested in, chosen from those described in the ontology, and then makes available (globally on the network and locally to the node) the documents matching the subscribed interests.

Looking at the complete network, that is, at the services connecting different nodes, the aim of the system is to allow users from different nodes to be informed of documents, on other nodes, which are interesting for them. This is where the ontology comes into play: sharing interesting documents across nodes is in fact possible because the nodes share a common ontology (or a subpart of it).

Every node in the network may have privacy policies, because not every document of a node should be read by all the other nodes: maybe only some information as title, abstracts, etc. can be shared. These policies are managed by the Nodes. Issues related to policy management are out of scope of this paper and are not described here.

Every document is characterized by a set of standard attributes (or tags), like its Name, Creation date, Abstract and so on, whereas the document type is chosen from the ontology: then the user can add other tags in a manual or semi-automatic way (see more details in Section V), selecting the values from the instances of the ontology and driven by their relationships.

The EC^{2}M system can be "instantiated" many times, to be useful for different groups of enterprises (that is, for new enterprises' networks): a new ontology, describing types of documents and their possible contents must be created, but the overall structure and behavior remain the same. In this sense, EC^{2}M is "parametric" in the used ontology.

The core portion of the functional requirements (services) specification of EC^{2}M system has been created using the method proposed in [25], and consisted of: (1) an UML Use Case Diagram, (2) the Use Case descriptions, (3) a glossary, and (4) the screen mockups (sketches of the future GUIs). During the project meetings, the industrial partners found the screen mockups (and the glossary) very effective in improving the comprehensibility of the use cases and useful to identify early ambiguities in the requirements specification.

We do not list here the Use Cases, but the services of the system are presented in the next Sections, with information about their implementation and coordination.

The main services are those related to the documents management, and are associated to the GUIs described later. The algorithm for automatically tagging a new document is explained in Section VI. The manual insertion of a new document is described with a concrete example in Section V. Lastly, the system offers services reserved to the administrator of the Node, for example those regarding the approval of the automatically tagged documents, the management of the ontology, users and of the rules for distributing documents over the network: those are cited while describing the Modules that use them.

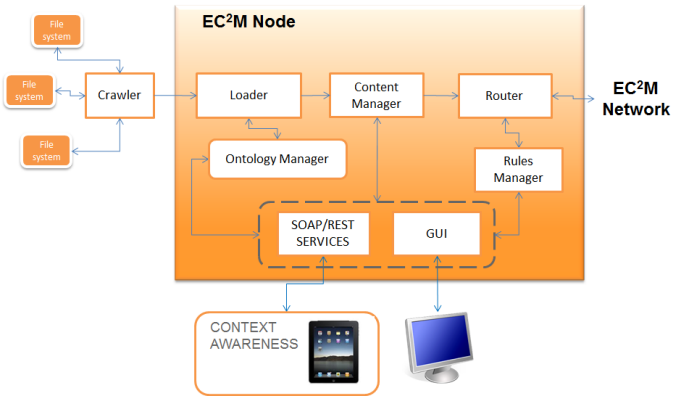

Figure 1. The high level architecture of the EC²M system.

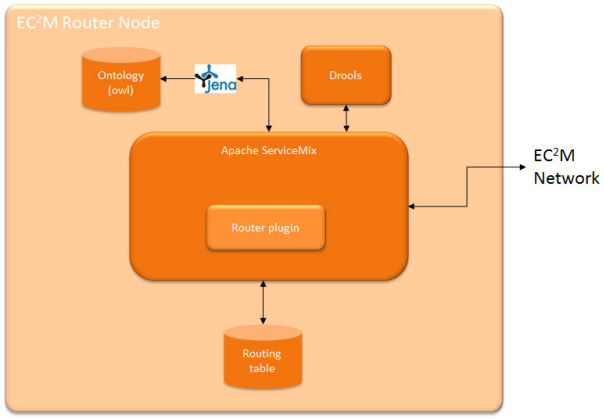

Figure 2. The software architecture of the Central Router.

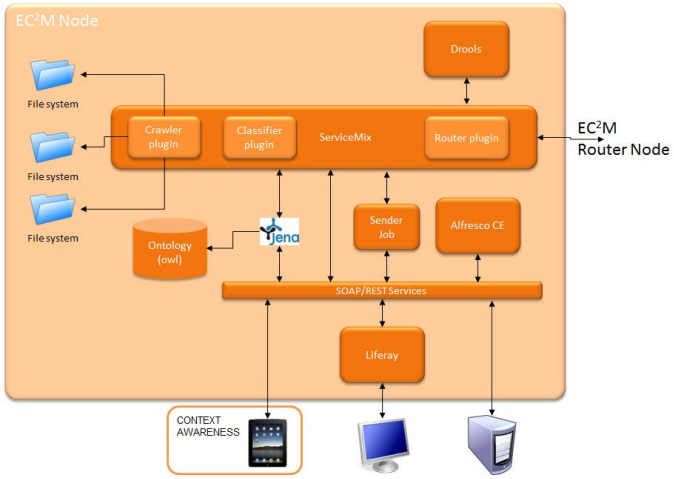

Figure 3. The software architecture of the EC²M system.

Looking at the system architecture from a high level point of view, the system is divided into different modules (see Figure 1):

- Graphical User Interface (GUI): front end of the system, where users can log in from a local terminal and can insert/retrieve/manage documents (based on the ontology);
- Soap and Rest Services: used by the GUIs and by the mobile application which, exploiting context information, lets the user access the system from a remote device;
- Crawler: software that explores predefined hard disk sectors to find documents that are sent to the Loader;
- Loader (or Classifier): module that automatically extracts from the documents tags corresponding to those in the ontology and that enriches the documents with tags related to those already manually associated with the document, using the instances in the ontology and their relationships;
- Ontology Manager: module that queries and manages the ontology;
- Content Manager: module that stores the documents and manages their sharing and retrieval;
- (Local) Router: module that manages the sharing of documents between nodes;
- Rules manager: module that the Router uses in order to define, and dynamically create, routing rules.

On the right side of Figure 1, the arrow points to the EC²M Central Router, reported in Figure 2. In Figure 3, the reader can see which (existing or new) software modules have been used to implement the EC²M system: in the next subsections we give more details on the functionalities and implementation of the different modules.

On each Node Apache ServiceMix is installed: the Crawler, the Classifier and the Router Modules exploit many of its services to implement their functionalities, as described later.

A. Graphical User Interface (GUI)

At design time, we decided to create GUIs simply changeable with respect to the user desires, so that any user can adapt its interfaces. To achieve this goal, we adopted Liferay [26], which is a well known Portal Server. With this platform, the GUIs are portlets that can be plugged into any pre-existing portal. To develop these portlets we used AJAX Vaadin, which is already integrated in the version 6 of Liferay.

Starting from the Use Cases describing the services that must be provided by the system, many GUIs (portlets) have been implemented. The main ones are:

- UploadPortlet: this GUI lets the user insert a new document. As shown later in Section V, from this GUI the user can manually associate tags, selected from the ontology, to the document. Furthermore, as last step before confirming the insertion, the system will suggest the user some more tags to be related to the document;
- SearchPortlet: this GUI lets the user search for documents. This GUI helps the user specifying the search criteria, which will be based on the ontology and on standard document properties (author, creation date and so on);

- ApprovePortlet: users can also put documents in a shared folder and let the Classifier automatically classify them (that is, adding tags from the ontology). These documents must be analyzed by a super user before they are really inserted into the system and shared among users. From this GUI the super user can analyze the automatically associated tags (with the algorithm shown in Section VI), can modify them and lastly can move on with the insertion of the documents in the system;
- SubscribePortlet: from this GUI the user can subscribe to a set of arguments that he is interested in. These arguments are chosen from those in the shared ontology, so that when a new document tagged with one of these arguments is inserted in the network, the user can be automatically informed by the Router Module.

Other GUIs exist to manage the administrative data, such as the users list, the current used ontology, the users permissions and so on.

GUIs exploit the SOAP/REST interfaces, described later, to invoke the services offered by the system.

B. The Crawler and the Classifier Modules

Apache ServiceMix offers many services and components: some of them help polling the file system, also using the FTP protocol. The Crawler Module is a plugin of the local instance of ServiceMix and exploits these services to cyclically search for new documents in predefined shared folders, where users can put documents.

These documents are then sent to the Classifier, which is again a plugin of ServiceMix (but is installed on the cloud): for each document received from the Crawler, it automatically selects related tags (as shown in Section VI) from the ontology and associates them to the document. Then these documents can be analyzed by the superuser to be definitively inserted into the Content Manager.

C. Ontology Manager

To create and manage the ontology we adopted the Web Ontology Language (OWL [27]) and Protégé [28]. The ontology is not subject to frequent changes: if it needs to be modified, this is done using Protégé and then the new version is again made available to the framework for queries.

D. SOAP/REST service

To let different clients interact with the system (from outside of the local Node or from the Node itself), we created a SOAP and a REST interfaces for the main services:

- uploadDocument: it is invoked to add a new document, with its tags, to the content manager;
- search: it lets the invoker searching for a document, specifying as input a list of tags;
- getDocument: it returns the document with the identification number given in input;
- getOntology: it returns the ontology used in the system;
- updatePosition: it is used to send the current position of a mobile device to the Node;
- getTicket: it takes as inputs a username and a password and returns an "authorization token" to let the user be identified in the system;
- subscribeUserFeed: it stores the user interests, given as input;
- retrieveUserFeed: it returns the list of user interests.

E. Content Manager

To physically manage the documents, we adopted Alfresco, a well known open source Java Content Management System: this system allowed us to exploit all the facilities of a high performances business platform with the good property of being an open source software. Furthermore, Alfresco takes advantage of many other well known open source systems like Spring, Hibernate, Lucene and MyFaces, helping developers in creating high quality software in a cost and time limited way.

F. Router

The Router Module is a plugin of ServiceMix, and manages the distribution of the documents in and out of the Node. There is a Router Module on each Node: this module is in charge of informing the node' users of the presence of new documents (local documents or on other nodes) and then of informing the other nodes (thank to the Central Router) of the existence of new documents. The local Router knows the users interested in the new documents (considering the subscribed interests of each user) and, thanks to the Rules Manager Module (described later), can inform them of their existence. Furthermore, it knows the documents exchange rules so it will also distribute the document (or only some parts, for example the title and abstract) to the Central Router (that will spread it to the other interested nodes). The local Router receives the documents from the Central Router too: then it sends these new documents to its Classifier so that they can be inserted into the local CMS.

G. Rules manager

Drools [29] is a well known open source system to create and manage business policies. Being this tool really stable, powerful and already integrated into ServiceMix, we decided to exploit it instead of developing a new module from scratch. The Drools rules are created thanks to an Eclipse plugin and then are read by the Router Module.

For example these are some of the rules regulating the documents notification to users:

- A Notification regarding a new document is sent only to users that subscribed to a topic that appears among the document tags.
- If a requested document is bigger than 1MB, only a link to the document is sent to the Mobile Users (not physically present in the Node), while the complete document is sent if it is smaller than 1MB.

- When a new document is inserted on a Node, only a link to it is to be sent to all the interested local standard (not-mobile) users (because they can directly download it from the local repository).

H. The central Router

The central Route, described in Figure 2, is developed exploiting again Apache ServiceMix as platform, whit the same plugin used for the Router Module, since their functionality is quite the same. As for the other modules, Drools is used to manage the routing rules and Jena is used to interface with the ontology. This Node manages the interaction among the Nodes in the network, distributing the new documents with respects to the subscribed interests of the Nodes. It owns the list of the Nodes in the network, and knows for each Node which are the topics, chosen from the common ontology, they are interested in (that are those the users of that Node are interested in): in this way the Central Router is able to correctly forward new documents among Nodes, avoiding flooding them with useless documents.

Its availability is assured by the cloud platform where it is deployed, reducing the risk of a single point of failure and of low performances.

I. Context Awareness

A mobile application for smartphones was developed for the system: it is able to identify the location of a user currently away from its company node and to act consequently, to let the user be informed of new documents or to let him search for documents in the system from a mobile device.

A "Fingerprint" (a set of information characterizing the Node) was calculated once for each Node and saved in the mobile application, so that it is able to identify when the user is near to its company Node and to move on with the automatic check-in (precondition to use the system). This context awareness algorithm for automatic check-in is called LRACI and is described in details in [30]: its performances are device independent, it is based on GPS/HPS information and is able to exploit Wi-Fi access points in an opportunistic way.

Thanks to this module, that interacts with the system using to the REST/SOAP interfaces developed for it, users can exploit the EC^{2}M system services from mobile devices in a transparent and automatic way.

IV. The Ontology

The EC^{2}M system was designed to work with any ontology describing the documents to be shared and their contents. To start with a concrete example, we decided to design, implement and use an ontology modeling the Sempla's business proposal. With respect to [1], the ontology structure has been changed a little, while some more instances have been created. Furthermore, we added many labels that are used in the automatic classification algorithm and in the GUIs, as described in the next Sections.

Sempla, as a brand, was founded in 2009 and operates in System Integration and Information Technology consulting. It has nearly three decades of experience with the most important Italian groups from the Financial, Production and Public sectors. Now Sempla has been transformed into GFT Italy, member of the GFT Group: we will anyway refer in the paper to Sempla, to keep consistence with [1].

This domain was chosen because Sempla is a very large enterprise, covering different business areas, so its documentation presents many types of documents and a large set of terms that are of interest for different users. These terms and types of documents are quite common in this business area, so modeling the Sempla domain is the best choice because the emerging ontology is correct also for Nacon (that now is member of the Sempla Group) and for Nis (that often collaborates with Nacon so can easily adhere to the ontology), which are the other Nodes in the system.

A. The Ontology Design

To model the domain with an ontology (as defined in [31]), we adopted the Noy and McGuinness methodology [32], which being an agile method is very suitable for collaborating with industrial partners. This methodology foresees these stages:

1) determine the domain and scope of the ontology;
2) consider reusing existing ontologies;
3) enumerate important terms in the ontology;
4) define the classes and the class hierarchy;
5) define the properties of classes-slots;
6) define the facets of the slots;
7) create instances.

The first step was quite simple to follow: the domain of the ontology was the Sempla's business proposal. It is translated into a complex organization of the logic concepts that describe what Sempla offers to its costumers, in terms of products and high technical and management consultancy.

Documents must be tagged to describe, with instances found in the ontology, their structure (some type of documents can have many attachments) and above all their technical and business items: for every business market Sempla has a "portofolio" of products and consultancy services that is well organized and defined.

We searched for similar ontologies, but we were not able to find one that was useful for modeling our domain. Maybe other companies own a similar ontology, but they are not public. We also considered existing ontologies, for example Bibo ontology [33], but we did not use them because they share only very few terms with those used in our domain. We could use the already exiting Sempla's documentation, which offered us an already well-defined set of terms describing the domain, although in natural language and not structured in any standard format.

The third step was conducted with the collaboration of the domain experts, which were the scientists from Sempla, Nacon and Nis. The majority of the terms was collected analyzing the brochures describing Sempla's business proposal (one is summarized in Table I) as well as a large set of documents selected as example from the real ones and a list of terms created by the "users-to-be" of the system, which listed the

TABLE I. THE SEMPLA BUSINESS PROPOSAL (BUSINESS ITEMS) FOR THE "FINANCIAL SERVICES" MARKET, GROUPED BY BUSINESS AREAS.

Business IT Consulting	Digital Marketing & Design	Business Solution	IT Solutions	IT Services	BPO
BPR; Studi di fattibilitá; Enterprise architecture planning; Program management consulting; Organizzazione processi IT; Project portfolio management.	Web & Content Design; User Experience; Community management; Digital Advertising; Augmented Experience.	Credit & Risk Management; Credito Lab; Credito al consumo; Filiale a CRM; Contact center; Pagamenti, Monetica, ATM/POS; Finance & Wealth Planning; Controlli e compliance; Sicurezza e Antifrode; Tesoreria; Human Resources; Reporting & Business Intelligence; Leasing & Factoring; Banca Virtuale; Project Portfolio; General ledger.	System Integration Framework; Application Frameworks; Metodologie di Delivery; Multicanalitá; Enterprise Content Management; DB Administrator.	Application Management; Application Modernization; IT Infrastructure Management; ITIL Implementation.	Contact Centre; Back Office; Fiscalitá Locale; Postalizzazione; Business travel management; Formazione, RollOut, Help Desk.

terms (corresponding to logic concepts used in their work) that they would like to use to classify a document.

To define the classes and their structure, we asked the domain experts to describe in details the types of documents they use, the most relevant information characterizing them and how they model the different business markets. We also took inspiration from the file system where documents were stored: the directories were partially organized as the business areas, and this organization reflected the way Sempla divides its business proposal.

The definition of the properties was done following a similar process.

As a last step, we manually inserted the instances in the ontology: the instances are named using some of the terms listed before, and the properties join them to completely describe the domain. It should be noted that in this process, stating what had to become a class and what an instance was a complex task, because some logical concepts of the domain may be mapped into a class (if we foresee a possible future extension) or into an instance as well (because they are something already stable and with different properties values), for example for *Market* instances. In these cases, we must consider that only instances will be used to tag documents, so it was an obliged decision to model those terms as instances.

Since the ontology is aimed at being used by Italian users, it was created in Italian but some terms (in particular the instances' names) are in English, to maintain a link with the existing documentation and Sempla's internal standards: we are aware that having an ontology with both Italian and English terms was not a perfect solution, but since Classes and Instances should be used as tags for many already existing documents, and considering that their names should be searched inside the text of the documents, we adopted anyway this solution.

B. The ontology details

The Sempla business proposal is organized considering different business markets, to propose ad hoc solutions for each area. Each high level business market is called *Mercato* (Market), and each *Mercato* is characterized by some distinct *Settore* (Sectors) (see Figure 4). To give and explanation of what we assume to be a market, consider that its instances are "Product", "Financial Services" and "Insurance", describing the main activities of the costumers operating in each market. Starting from this macro division of areas, all the other classes are related and organized considering these three sectors. For example, each costumer (class *Cliente_o_Prospect*) refers to one sector, so that his market is uniquely identified.

The Sempla business proposal is created combining different items, identified with the class *Business_Item*, which are divided into different types (subclasses) and that refer to six different Business Areas (as shown in Table I), modeled with the class *Area_Business*, with instances: "Business_IT_consulting", "Business_project_outsourcing (BPO)", "Digital_Marketing_And_Design", "Business_Solution", "IT_services", "IT_solutions".

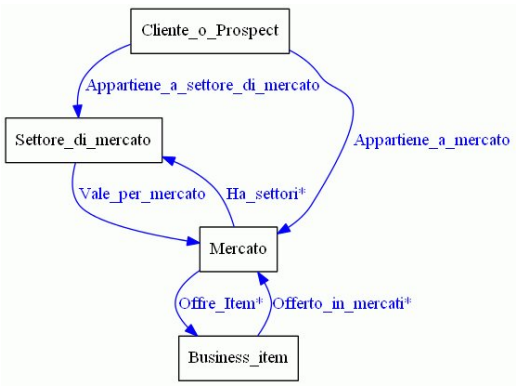

Figure 4. Class *Mercato (Market)* and its main relationships.

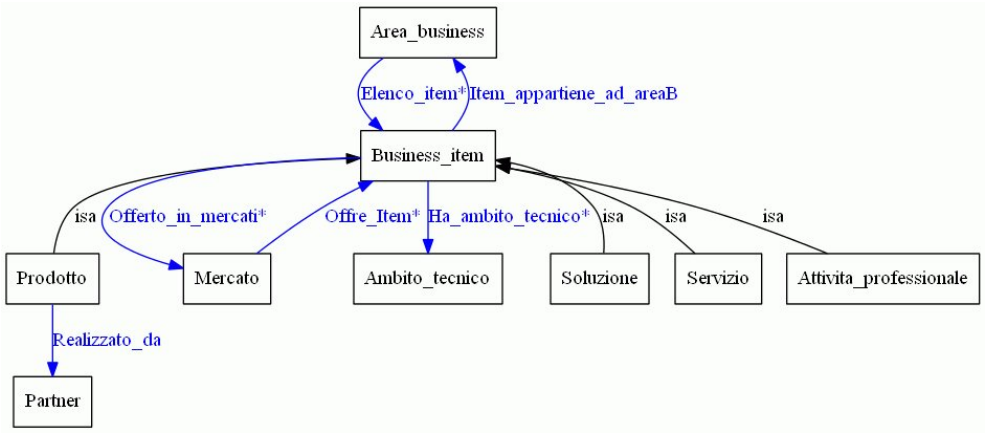

Figure 5. Class *Business_Item* and its main relationships.

Every *Business_Item* can be associated with only one *Area_business* but can be offered to different Markets. A business item that is offered in many markets is called "cross market". The class that groups these items is defined with a necessary and sufficient condition, and is called *BI_Cross_Mercato*. In a similar way, the class *AB_cross_mercato* is defined with the necessary and sufficient condition that it collects the business areas offering at least one "cross market" item.

The business items are divided into four disjoint subclasses: *Prodotto* (Product), which can be developed by a *Partner*, *Soluzione* (Solution), *Servizio* (Service), *Attivitá_professionale* (Professional Activity). These classes are related with class *Ambito_Tecnico* (Technical Area) describing at high level the IT area they refer to (for example "Client Server Application", "Mobile Application" and so on).

The relationships between the classes described above are shown in Figure 5.

The documents that must be classified are divided into different types, modeled with the classes (subclasses of *Documento* (Document)):

- Proposal (class *Documento_offerta*), which describes a business proposal to a costumer;
- Technical attachment (class *Allegato_tecnico_offerta*), which is a technical attachment describing at least one *Business_item* and one *Ambito_Funzionale*;
- Presentation (class *Presentazione*), that describes the Sempla business proposal to a specific Market, and that must be related to at least one *Business_item*.

Class *Ambito_Funzionale*, with its three subclasses, represents a further classification of the business services from a commercial viewpoint. With respect to [1], the subclasses of *Documento* have been slightly simplified.

In Figure 6, a detailed view of the relationships among *Proposal* and the other classes is shown: class *Proposal* has one string property ("Id_Proposal") not shown in Figure 6, which uniquely identifies the proposal. In Figure 7, an overview of the relationships among the different types of documents is reported.

Class *Partners* represents a list of possible third parties companies involved in the proposal.

We only show Object properties in figures and do not report instances of every class. In Section V, the reader can find some of these instances shown in figures and tables, while they are used by the EC^{2}M interface to help user tagging a document.

C. Exploiting the ontology

As described before, the main goal of the ontology is to formally model the domain and to store the common information needed to tag documents, using the ontology instances, in the system.

But the ontology has been also used to store other information related to Classes and Instances that will be used by the GUIs and by the automatic classification module.

Regarding the exploitation of the ontology by the GUIs, please consider that the EC^{2}M system was created to help different users, from different companies, to share documents. It is possible that users speak different languages: consequently, the GUIs should adapt to the desired language. Similarly, some classes or instances names suffer of a formalism that is due to OWL, but that could create confusion to the end users. Since GUIs read from the ontology the possible tags and have to show the users them plus the ontology class the tags refer

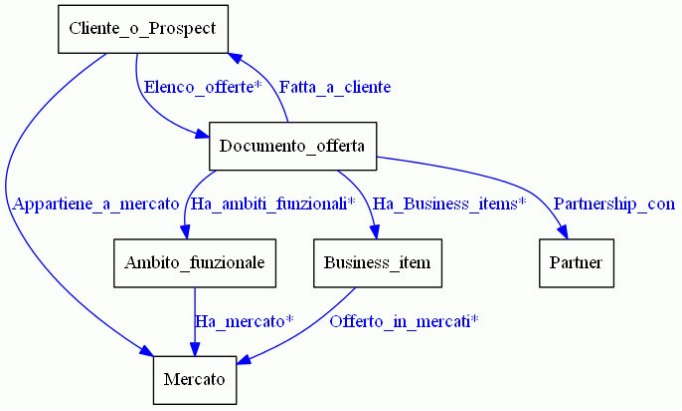

Figure 6. Class *Documento_offerta* (*Proposal*) and its main relationships.

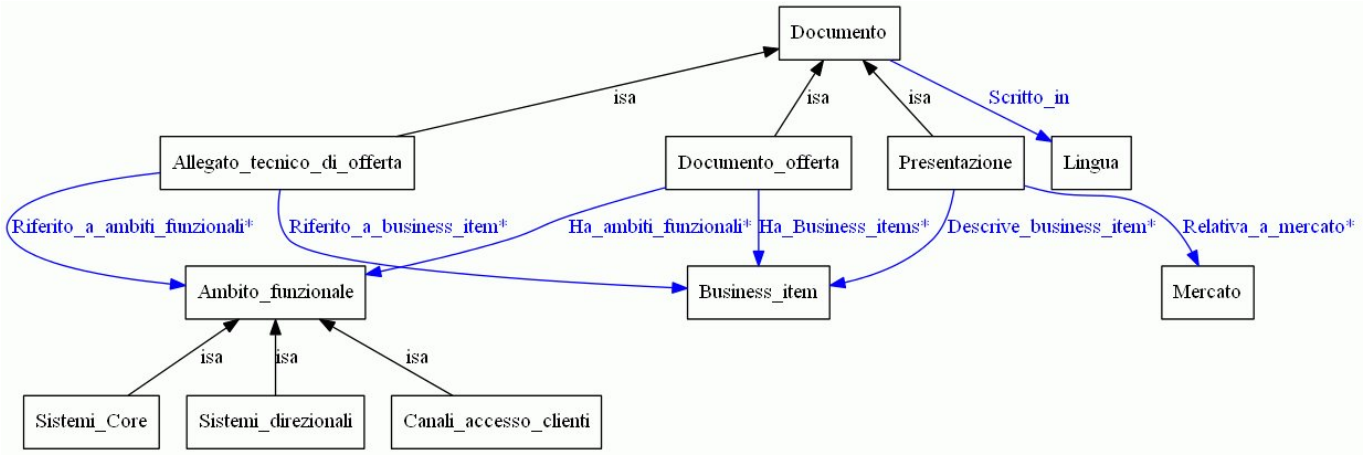

Figure 7. Relationships among classes starting from *Documento* (*Document*) and its subclasses.

to, we added to ontology classes some rdfs:labels to store the value that must be visualized in the GUIs. For example, as shown in Section V and Section VI, to Italian users the class *Business_Item* is shown with the name "Elemento d'offerta", read from the label associated to that class.

Regarding the exploitation of the ontology by the automatic classification module, we improved the information stored in the ontology by adding more synonyms, acronyms, equivalent definitions and generic information to classes and instances, storing them into rdfs:labels: these terms are the elements of the "synsets" that will be read and used by the automatic classification module, as shown in Section VI.

V. THE RUNNING SYSTEM

A first prototype of the EC^{2}M system has been developed at the end of 2012, based on the ontology described in Section IV of [1], and has been tested concretely by Nacon, Nis and Sempla to manage the documents created in 2012 (3200 documents), with 31 users. Now the system is completely updated and running (using the new version of the ontology presented in this paper), serving more than 600 users, and manages a documents set that grows each year approximatively of 3000 new items.

Some images of the GUIs are reported to give an example of usage. The GUI form in Figure 8 is the one where the user can add a new document, specifying: what kind of document he is inserting (a "Presentazione"), the market (in this case "Insurance") and the business items the document is about (selecting these values using drop down lists that group the business item instances in their subclasses). In the example, the user chooses the instances "User_Experience", "Web_development" and "Digital_media_strategy" from class *Attivitá_professionale* (instances of class *Attivitá_professionale* are shown in Table II). Finally, the user chooses the language ("Italiano").

Figure 8. First GUI form for inserting a new document with some first tags. Screen shot.

In Figure 9, the system presents to the user the list of the current tags that he selected plus the tags that have been automatically added: in this case only the tag "Digital_Marketing_Design" from class *Business_Area*, because all the selected business items refer to this area, as shown in Table III.

Note that in the ontology, labels have been added to classes to store the term to be shown in the GUIs, because sometime the class names are not "good to be visualized" (for example they contain the "_" character, or are in a different language from the GUI one). For example, in Figure 9 the tags from class *Business_Item* are called "Elemento d'offerta" instead of the standard class name.

Then, the system asks the user if he wants to add some more tags, choosing from those connected to the already selected ones. In the example, the user chooses to add more tags starting from the business area "Digital_Marketing_Design". So the system shows the user the possible tags, choosing from the instances related to "Digital_Marketing_Design" (considering the properties with domain *Business_Area*) in the ontology (Figure 10). The user can add some of those tags and then saves the document.

TABLE II. INSTANCES OF THE *Attivitá_professionale* CLASS (SUBCLASS OF *Business_Item*).

User Experience	Web development	Digital media strategy	ADV Campaign
Program management consulting	IT infrastructure management	Project portfolio management	Enterprise architecture planning
ITIL Implementation	Business driven development	Delivery Methodologies	IT Processes Organization
Application Modernization	Digital Marketing	Feasibility Study	Visual Graphic Design
BPR	Brand Identity	Time Material	Movie Design

TAG	CATEGORIA
Insurance	Mercato
User Experience	Elemento d'offerta
Web development	Elemento d'offerta
Digital media strategy	Elemento d'offerta
Italiano	Lingua
Digital_Marketing_Design	Area_business

Figure 9. List of a manually selected tags plus those automatically added. Screen shot.

TABLE III. *Business_Item* INSTANCES ASSOCIATED TO THE "DIGITAL_MARKETING_DESIGN" BUSINESS AREA.

User experience	Movie Design	ADV Campaign	Brand Identity
Digital Media Strategy	Visual Graphic Design	Digital Marketing	Web Development

Scegli ulteriori tag da associare al documento

TAG	CATEGORIA
Community	Ambito tecnico
Forum	Ambito tecnico
Portale	Ambito tecnico
Sito web	Ambito tecnico
Sito mobile	Ambito tecnico
Product	Mercato
Financial Services	Mercato

TAG	CATEGORIA
Insurance	Mercato
User Experience	Elemento d'offerta
Web development	Elemento d'offerta
Digital media strategy	Elemento d'offerta
Italiano	Lingua
Digital_Marketing_Design	Area_business

Figure 10. Possible new tags (left) and already associated tags (right). Screen shot.

VI. THE AUTOMATIC CLASSIFICATION ALGORITHM

The EC^{2}M system is equipped with the Loader/Classifier module, able to automatically classify a document given as input: it is based on Lucene [34] (to interact with Alfresco for getting the indexes and to analyze the documents, searching for terms) and on Jena [35] (to read the ontology classes, instances and labels). It follows an algorithm based on [36], and is used to automatically identify the type of the input document and the possible related tags (chosen from the ontology). These automatically associated tags are then presented to the user who can confirm or change them.

The UML activity diagram in Figure 11 (created following the methodology described in [37], as for that in Figure 12) shows the Classifier's algorithm devoted to the identification of the document type:

- It asks the Ontology Manager (that uses Jena) the list of Document Types (that are the subclasses of the ontology class *Documento* (Document)) with their related synsets (that are the sets of rdfs:labels associated to each class);
- For each document type, it:
  - searches in the document title the terms in the document type synset;
  - searches in the document body the terms in the document type synset;
  - If at least one correspondence is found, the type is considered in the list of possible candidates, and a score is calculated based on the frequency of the synset terms found in the previous steps. If the term is found in the title, it is multiplied for a predefined "boost value".
- At the end of the loop, the document type with the highest score (that is, the uniquely found document type or that with a score exceeding of a predefined value, called "DELTA", those of the other document types) is associated to the document. If there are many types with similar scores, none type is associated to the document.

The algorithm that automatically extracts the semantic tags is described with an activity diagram in Figure 12:

- It asks the Ontology Manager (that uses Jena) the properties list of the document type chosen in the previous phase of the algorithm (if any);
- For each datatype property, it calculates the possible value analyzing with Lucene the document, searching for the property name (or for the terms in its synset, stored again as rdfs:labels related to the property) followed by a value of the correct property type. This couple is added to the set of indexes related to the document (indexes are "not semantic" standard tags that can be related to a document in a CMS: they are managed directly by Alfresco apart those created in this step, which are automatically calculated as described and added to this set);
- For each Object Property, it asks the Ontology Manager the instances of that Class with their related synsets (that are the sets of rdfs:labels associated to each instance) and for each instance:
  - searches in the document title the terms in the synset;
  - searches in the document body the terms in the synset;
  - If at least one correspondence is found, the instance is considered in the list of possible candidates, and a score is calculated based on the frequency of the synset terms found in the previous steps. If the term is found in the title, it is multiplied for a predefined "boost value";
- At the end of the loop, if the property has single cardinality, the instance with the highest score (that is, the uniquely found instance or the instance with a score exceeding of a predefined value, called "DELTA", those of the other instances) is associated to the document while, if it has multiple cardinality, all the instances with a score higher than a predefined "threshold" are associated to the document;
- Lastly, considering each instance associated to the document in the previous steps, its data properties with a value are added as indexes, and other instances (if any) related by object properties are added as tags to the document too.

The "boost value" is a parameter used to give some more importance (that, in this case, is an higher score) to a term if it has been found in the title of a document. Its value may range between 1 and 2. After many tests to identify the most

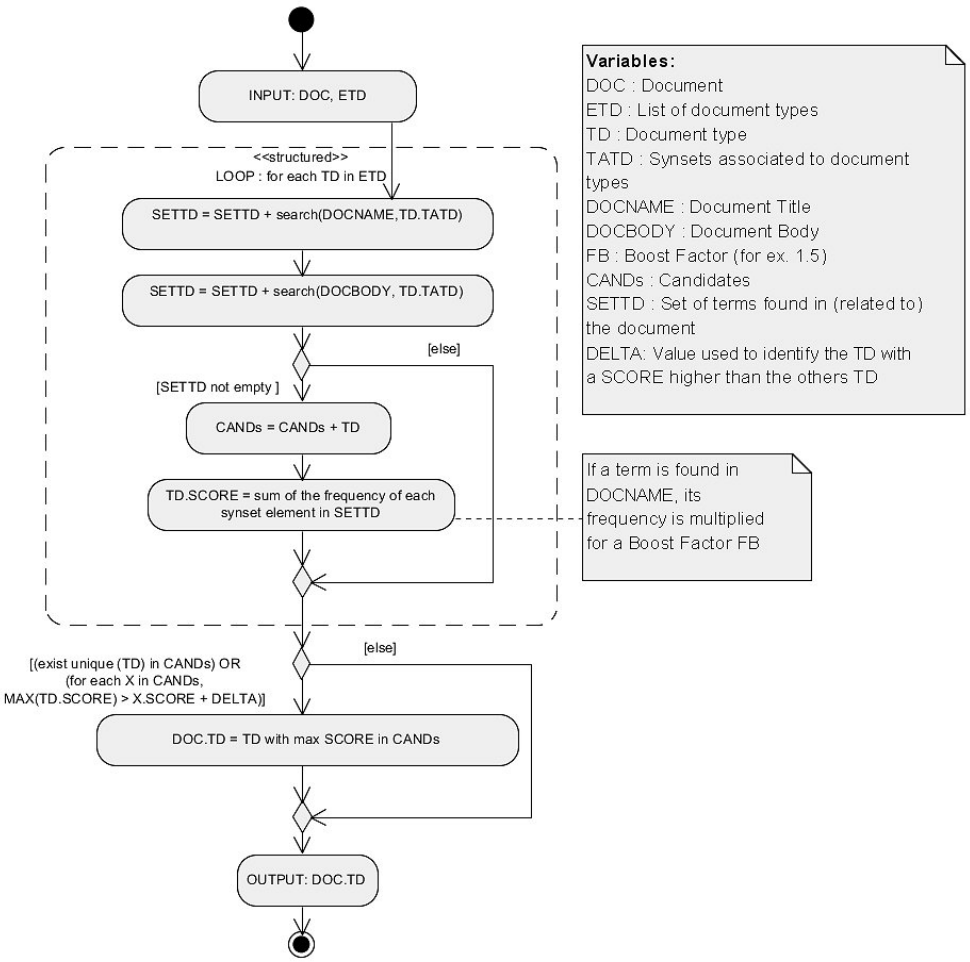

Figure 11. Activity Diagram describing the algorithm to identify the document type.

correct value for this parameter, we now use it with value 1.5. In general, if you have a domain with documents with relevant titles, an higher value of this parameter is recommended.

The "DELTA" parameter, as explained before, is used to select only the instance with a score really higher then that of the other candidates, and again has been tuned during the tests: its value may range between 0 and 1, and we use it set to 0.2.

Lastly, the "threshold" parameter is used to select only a subset of the identified candidates to be proposed to the user: at the end of the algorithm we have a list of candidates with a score, which ranges from 0.1 (the lowest value) to 1 (the highest one), reflecting the confidence that the candidate is really associable to the document. We use the "threshold" parameter to choose, between them, only those with a reliable score. Its value may range between 0 and 1, and currently this value is set to 0.3 (as explained later in this Section).

An example is reported in Figure 13: the document in input, "06-Portale del Credito-CR2 - Revisione PDC GOR - MS.pptx", is automatically correctly identified as a *Presentazione* (Presentation).

This class has two properties, as shown in Figure 7: "Scritto_In (Written In)" (single, with range *Lingua* (Language)), and "Descrive_Business_Item (visualized as "Describe" in Figure 13)" (multiple, with range *Business_Item*). The algorithm correctly identifies the language (instance "Italian") and selects three related *Business_Item* for the property "Describe" ("Credito al consumo", "Credito lab", "Credit and risk management").

As last step, considering that all the three *Business_Items* are related to the "Business Solution" instance of *Business_Area* class (see Table I) thanks to the property "Item_appartiene_ad_areaB" (see Figure 5), the tag "Business Solution" is associated to the document.

The first tests, presented in [1], gave already promising results: considering the automatic tags extraction, those with a threshold higher than 0.3 were correctly associated with the document in the 95% of the executed tests. In the following months the system has been tuned to achieve better results: the algorithm was not modified, while the ontology has been changed to create more complete synsets (in fact many wrong tags associations were due to poor synsets). Now that the ontology has been updated and revised, the percentage of correct tags is 100% with threshold higher than 0.5, and 97% with threshold between 0.3 and 0.5.

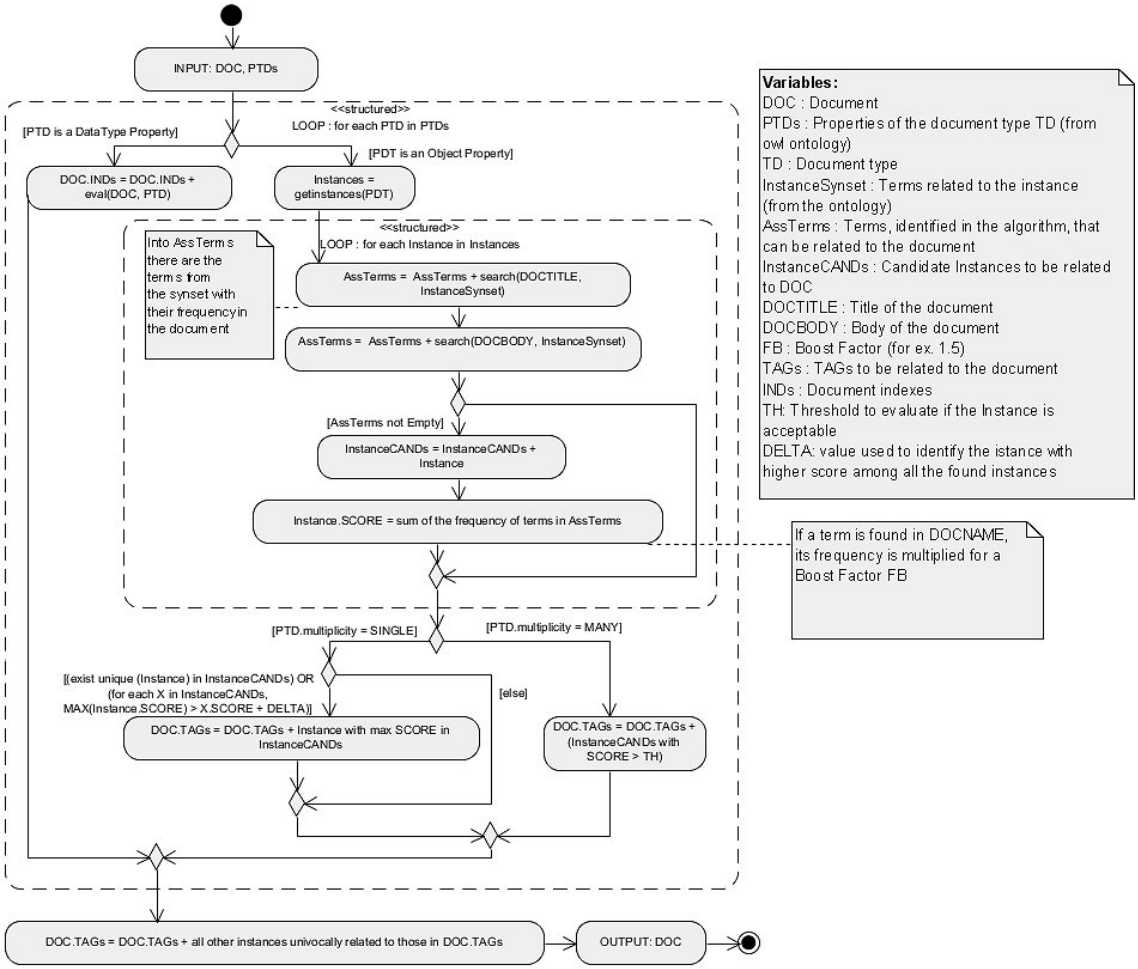

Figure 12. Activity Diagram describing the algorithm to identify the document tags.

Document type	Score
Presentation	1.0
Proposal	0.2
Technical Attachment	0.1

Property	Cardinality	Class	Instance	Score
Describe	Multiple			
		Product	Credito_lab	1.0
		Solution	Credit e risk management	0.83
		Solution	Credito al consumo	0.5
		Time material	Organizzazione_processi_IT	0.07
		Solution	Gestione sinistri e reporting	0.05
		Time material	Web development	0.03
		Solution	Controlli e compliance	0.03
		Solution	Reporting e business intelligence	0.03
		Time material	Digital media strategy	0.02
		Service	Application Management	0.01
		Time material	Studio fattibilità	0.01
		Time material	Web Content Management	0.01
		Solution	Business travel management	0.01
Written in	Single			
		Language	Italian	1.0

Class	Instance
Business area	Business solution
Solution	Credit e risk management
Language	Italian
Solution	Credito al consumo
Product	Credito_lab

Figure 13. Automatically identified tags (highlighted rows) in the document used for example.

VII. Conclusion and Future Work

The solution that we adopted is based on a mesh-up of many different technologies and softwares, where an ad hoc ontology is integrated into an open-source CMS and then deployed over a cloud platform, following the new trends in different research areas. As shown in Sections I and II, many solutions exist that are in some way similar to ours (considering the adoption of a formal representation of the domain, or for the similar overall structure) but are really far for other aspects (different domain, different semantic representation and expressiveness, or simply being only a study, not a real and applied tool). It is impossible to make a precise comparison with the other mentioned systems, because they are commercial or not available to the community, or because a completely new system, using the other approaches or technologies, should be developed to be tested and compared, and this solution is not feasible. As far as we were able to do, we adopted the open-source softwares recognized by the users and by the research community as very reliable and efficient ones (for example Lucene and Alfresco), while our industrial partner chose the cloud platform making again a detailed analysis of the available platforms (*Eucalyptus* and *Amazon Web Services* by Amazon, *OpenStack* by Rackspace, *AppScale* by Google), to adopt the best one for our project. To develop our ontology, considering that it was the core of the system, we adopted OWL, since it is one of the most expressive language for developing ontology and that some well working libraries to interface and manipulate it from an external program exist (like Jena).

The EC^{2}M project has been fully tested in 2012 and 2013, and showed very good results with respect to a standard ECM system: it really helped users from different companies better collaborate, exploiting a semantic classification of their documentation and consequently offering a simpler searching phase and a better support in sharing information, which was impossible to obtain without a similar system.

The ontology is now complete and models all the document types and contents. It has been equipped with labels to store both "visualization information" (that are the labels that must be visualized in the GUIs) both "synsets". This solution was definitely good for our system because:

- it lets the system administrators simply modify some parts of the GUIs only modifying the ontology, reducing the chance of an misalignment between the terminology and the GUIs, and reducing the time-to-market too
- it lets simply modify the instances and their synsets, so that the automatic classification algorithm can consequently improve its results thanks to a more complete ontology

keeping the independence between the domain representation, the GUIs and the automatic classification algorithm itself.

The notification of new documents to subscribed users is done in quasi real time, as requested by these types of applications, for the remote and mobile users too. Furthermore, with the deployment over the cloud platform, the performances can be enhanced with a new purchase of cloud services: with this architecture and deployment solution the system is really scalable.

The system has been adopted by GFT (previously Sempla), and related companies, as their new content management system, since it showed very good performances (thanks to the cloud architecture), great accessibility (thanks to the mobile access and context awareness module), high reliability in the automatic tagging process, and high adaptability (thanks to the adoption of the ontology as an independent domain representation). So, it is not only a prototype, but a real and running system that shows how these solutions coming from the research fields (ontologies, context awareness, cloud and CMS composed together) can actually and effectively work in real industrial scenarios.

Furthermore, the adoption of open source technologies and the chance of exploiting the system as a service over the cloud platform (with a pay-per-use solution) allowed a reasonable initial budget and a high scalability in the overall architecture, which from the industrial viewpoint is a good "Return Of Investment": that is another evidence of the applicability of these technologies in industrial systems.

As last future work, we will investigate how dealing with the scenarios where the nodes in the EC^{2}M network use different ontologies to describe and tag their documentation: in this case, the common ontology must be anyway chosen and defined, but ontology matching techniques may be adopted to align the common and private ontologies before tagging and when receiving notifications from the other nodes, to let them keep on using their private ontology but also being able to share documents with common tags.

Acknowledgment

The research described in this paper has been funded by the "EC^{2}M system (Enterprise Cloud Content Management)" Programma Operativo Regionale (POR) project.

The authors would like to thank Maurizio Ferraris from Nacon and Viviana Mascardi and Gianna Reggio from DIBRIS that led the industrial and the DIBRIS academic components, respectively, involved in the EC^{2}M project. Thanks also to Maurizio Leotta for the activity Diagrams shown in this article.

A special thanks goes to Dante Laudisa, from Sempla, who actively participated in the domain explanation and in the ontology definition.

References

[1] D. Briola, A. Amicone, and D. Laudisa, "Ontologies in Industrial Enterprise Content Management Systems: the EC^{2}M Project," in COGNITIVE 2013, The Fifth International Conference on Advanced Cognitive Technologies and Applications, 2013, pp. 153–160.

[2] Association for Information and Image Management, "What is Enterprise Content Management (ECM)?" 2010, URL: http://www.aiim.org/what-is-ecm-enterprise-content-management.

[3] A. P. Sheth and C. Ramakrishnan, "Semantic (Web) Technology In Action: Ontology Driven Information Systems for Search, Integration and Analysis," IEEE Data Eng. Bull., vol. 26, no. 4, 2003, pp. 40–48. [Online]. Available: http://dblp.uni-trier.de/db/journals/debu/debu26.html#ShethR03

[4] R. Andersen, "The Rhetoric of Enterprise Content Management (ECM): Confronting the Assumptions Driving ECM Adoption and Transforming Technical Communication," Technical Communication Quarterly, vol. 17, no. 1, 2008, pp. 61–87.

[5] "The Alfresco Homepage," 2014, URL: http://www.alfresco.com/ [accessed: 2014-03-01].

[6] "The Plone Homepage," 2014, URL: http://plone.org/ [accessed: 2014-03-01].

[7] "The Sensenet Homepage," 2014, URL: http://www.sensenet.com/ [accessed: 2014-03-01].

[8] "The smartlogic Homepage," 2014, URL: http://www.smartlogic.com [accessed: 2014-03-01].

[9] "The H-Dose Homepage," 2014, URL: http://dose.sourceforge.net/ [accessed: 2014-03-01].

[10] "The OpenCalais Homepage," 2014, URL: http://www.opencalais.com/ [accessed: 2014-03-01].

[11] "The Apache Stanbol Homepage," 2014, URL: http://stanbol.apache.org/ [accessed: 2014-03-01].

[12] R. García, J. M. Gimeno, F. Perdrix, R. Gil, and M. Oliva, "The Rhizomer Semantic Content Management System," in WSKS (1), ser. Lecture Notes in Computer Science, M. D. Lytras, J. M. Carroll, E. Damiani, and R. D. Tennyson, Eds., vol. 5288. Springer, 2008, pp. 385–394. [Online]. Available: http://dblp.uni-trier.de/db/conf/wsks/wsks2008.html#GarciaGPGO08

[13] C. Frank and M. Gardoni, "Information Content Management with Shared Ontologies-at Corporate Research Centre of EADS," Int. J. Inf. Manag., vol. 25, no. 1, Feb. 2005, pp. 55–70. [Online]. Available: http://dx.doi.org/10.1016/j.ijinfomgt.2004.10.009

[14] D. M. Le and L. M. S. Lau, "An Open Architecture for Ontology-Enabled Content Management Systems: A Case Study in Managing Learning Objects," in OTM Conferences (1), ser. Lecture Notes in Computer Science, R. Meersman and Z. Tari, Eds., vol. 4275. Springer, 2006, pp. 772–790. [Online]. Available: http://dblp.uni-trier.de/db/conf/otm/otm2006-1.html#LeL06

[15] G. Laleci, G. Aluc, A. Dogac, A. Sinaci, O. Kilic, and F. Tuncer, "A Semantic Backend for Content Management Systems," Knowl.-Based Syst., vol. 23, no. 8, 2010, pp. 832–843. [Online]. Available: http://dblp.uni-trier.de/db/journals/kbs/kbs23.html#LaleciADSKT10

[16] H.-C. Chu, M.-Y. Chen, and Y.-M. Chen, "A Semantic-based Approach to Content Abstraction and Annotation for Content Management," Expert Syst. Appl., vol. 36, no. 2, Mar. 2009, pp. 2360–2376. [Online]. Available: http://dx.doi.org/10.1016/j.eswa.2007.12.067

[17] A. Bechini and A. Vetrano, "Management and Storage of in Situ Oceanographic Data: An ECM-based Approach," Inf. Syst., vol. 38, no. 3, May 2013, pp. 351–368. [Online]. Available: http://dx.doi.org/10.1016/j.is.2012.10.004

[18] B. Thönssen, "An Enterprise Ontology Building the Bases for Automatic Metadata Generation," in MTSR, ser. Communications in Computer and Information Science, S. S. Alonso and I. N. Athanasiadis, Eds., vol. 108. Springer, 2010, pp. 195–210. [Online]. Available: http://dblp.uni-trier.de/db/conf/mtsr/mtsr2010.html#Thonssen10

[19] J. Corchado, D. Tapia, and J. Bajo, "A Multi-Agent Architecture for Distributed Services and Applications," Computational Intelligence, vol. 24, 2008, pp. 77–107.

[20] D. Griol, J. M. Molina, and Z. Callejas, "Providing Personalized Internet Services by means of Context-Aware Spoken Dialogue Systems," JAISE, vol. 5, no. 1, 2013, pp. 23–45. [Online]. Available: http://dblp.uni-trier.de/db/journals/jaise/jaise5.html#GriolMC13

[21] G. Blázquez, A. Berlanga, and J. M. Molina, "inContexto: Mobile Phone Multi-Sensor Architecture to Obtain People Context," Journal of Ambient Intelligence and Smart Environments, vol. 5, 2013, pp. 23–45.

[22] R. Fuentes-Fernández, J. Gómez-Sanz, and J. Pavón, "Understanding the Human Context in Requirements Elicitation," Requirements Engineering, vol. 15, no. 3, 2010, pp. 267–283. [Online]. Available: http://dx.doi.org/10.1007/s00766-009-0087-7

[23] J. L. R. García and A. L. Tello, "Ontotv: an Ontology-Based System for the Management of Information about Television Contents," Int. J. Semantic Computing, vol. 6, no. 1, 2012, pp. 111–. [Online]. Available: http://dblp.uni-trier.de/db/journals/ijsc/ijsc6.html#GarciaT12

[24] A. Aiello, M. M. Furnari, A. Massarotti, S. Brandi, V. Caputo, and V. Barone, "An Experimental Ontology Server for an Information Grid Environment," International Journal of Parallel Programming, vol. 34, no. 6, 2006, pp. 489–508. [Online]. Available: http://dblp.uni-trier.de/db/journals/ijpp/ijpp34.html#AielloFMBCB06

[25] G. Reggio, F. Ricca, and M. Leotta, "Improving the Quality and the Comprehension of Requirements: Disciplined Use Cases and Mockups," in Proceedings of the 40^{th} Euromicro Conference on Software Engineering and Advanced Applications (SEAA 2014). IEEE, 2014.

[26] "The Liferay Homepage," 2014, URL: http://www.liferay.com/ [accessed: 2014-03-01].

[27] "The OWL Language Overview Homepage," 2014, URL: http://www.w3.org/TR/owl-features/ [accessed: 2014-03-01].

[28] "The Protégé Homepage," 2014, URL: http://protege.stanford.edu/ [accessed: 2014-03-01].

[29] "The Drools Homepage," 2014, URL: https://www.jboss.org/drools/ [accessed: 2014-03-01].

[30] I. Bisio, F. Lavagetto, M. Marchese, and A. Sciarrone, "GPS/HPS- and Wi-Fi Fingerprint-Based Location Recognition for Check-In Applications Over Smartphones in Cloud-Based LBSs," Multimedia, IEEE Transactions on, vol. 15, no. 4, June 2013, pp. 858–869.

[31] T. R. Gruber, "A Translation Approach to Portable Ontology Specifications," Knowl. Acquis., vol. 5, no. 2, Jun. 1993, pp. 199–220. [Online]. Available: http://dx.doi.org/10.1006/knac.1993.1008

[32] N. F. Noy and D. L. Mcguinness, "Ontology Development 101: A Guide to Creating Your First Ontology," Tech. Rep., 2001.

[33] "The Bibliographic Ontology Homepage," 2014, URL: http://bibliontology.com/specification [accessed: 2014-03-01].

[34] "The Apache Lucene Homepage," 2014, URL: https://lucene.apache.org/ [accessed: 2014-03-01].

[35] "The Jena Homepage," 2014, URL: https://jena.apache.org/index.html [accessed: 2014-03-01].

[36] K. S. Jones, "A Statistical Interpretation of Term Specificity and its Application in Retrieval," Journal of Documentation, vol. 28, 1972, pp. 11–21.

[37] G. Reggio, M. Leotta, F. Ricca, and E. Astesiano, "Business Process Modelling: Five Styles and a Method to Choose the Most Suitable One," in Proceedings of the Second Edition of the International Workshop on Experiences and Empirical Studies in Software Modelling, ser. EESSMod '12. New York, NY, USA: ACM, 2012, pp. 8:1–8:6. [Online]. Available: http://doi.acm.org/10.1145/2424563.2424574