GODO: Goal driven orchestration for Semantic Web Services

Juan Miguel Gómez [1], Mariano Rico[2], Francisco García-Sánchez [3], Rodrigo Martínez Béjar[3] , Christoph Bussler[1]

[1] Digital Enterprise Research Institute (DERI). National University of Ireland, Galway
Galway, Ireland.
{juan.gomez,chris.bussler}@deri.org
[2] Universidad Autónoma de Madrid
Madrid, Spain.
Mariano.rico@uam.es
[3] Universidad de Murcia
Murcia, Spain
{fgs2@alu.um.es,rodrigo@dif.um.es}

Abstract. Current orchestration models rely on workflow specifications and old-fashioned hard wired means. With the advent of Semantic Web technologies, a goal driven approach provides a basis to establish a ubiquitous technical platform for eCommerce interactions based on customer wish and needs. Ideally, a customer would like to specify their goals in natural language and wait for them to happen. In this paper, we present GODO, a best of breed ultimate engine which uses natural language processing and mapping techniques for orchestrating goals and achieves them by means of web services.

1. Introduction

The Internet is going trough several major changes. Recently, it has become a new vehicle for business transactions and information exchange rather than just a repository of information. Companies are challenged to publish and share services on the web. Furthermore, integration of services through different companies would foster the development of Business-to-Business (B2B) [1] and Business-to-Consumer (B2C) interactions by sharing costs and reusability.

As the technologies associated to eCommerce interactions, such as web services gain momentum, the need for an application to fulfill the customer expectations becomes increasingly important. The goal of current Semantic Web Services initiatives such as OWL-S [9] or WSMO [2] is to actually add semantics to current web services to match the expectations of the consumer of the web service. The most prominent compliant implementation is the WSMX [3] platform. In the WSMX architecture, a goal [6] specifies the objectives that a client may have when he consults a web service and it also contains a list of preferences [7]. These preferences represent constraints on non-functional properties of a web service (i.e., they narrow the scope of the selection spectrum of a web service).

In B2C, the real challenge is therefore to commit ad maximum with the customer's wish. This is the main impetus behind this work. Goal driven orchestration is inspired in the principles of a semantically enhanced goal-driven approach in which a customer writes down its wishes and they immediately turn into goals which will be processed by WSMX giving as a result the final expected overall result.

In this paper, we propose the Goal Driven Orchestration, for short, system, and depict its architecture and technical details about its implementation. Finally, we detail a use case which sheds light on the advantages and interest of our approach.

This paper is structured as follows. In section 2, we present a general introduction of GODO and its architecture. Section 3 shows an example and some advantages of our approach. Finally, we present future and related work in section 4.

2. Goal driven orchestration

2.1. Making the dreams of users come true

In May 2001, Tim Berners-Lee, James Hendler and Ora Lassila, depicted their vision of the semantic web in their famous Scientific American article [18] . They showed us a future world where the software agents could deal with our wishes, exchanging information with other agents, offering us the best options and executing our final decisions. Current efforts envision reaching that ideal world, and a critical aspect is the way these agents interact with us. The current state-of-the-art is based on the concept of goal and how to describe it, but, at the end, we have always a human being, without any idea about computer science or the existence of agents. It is clear that a user wants to express his wishes in a more familiar language: natural language.
Expressing the user wishes is hence the first step in the chain. Voice recognition systems are restricted to specific requisites such as headphones, noiseless environments, etc. Furthermore, most of the user interaction is through the web navigator. Ideally, the user would interact with the agent through this well-known device and write down there his wishes. In a second step, his goals could be extracted, searched and finally, achieved. Our proposal goes in the direction of this second step with the help of Semantic Web technologies [14]. These technologies are bout adding machine-processable semantics to data so that the computer can understand the information and process it on behalf of the human user. Finally, the ultimate goal is to have an automation process of the aforementioned tasks required for a web service. For this, the new paradigm of Semantic Web Services promises to enable users to locate, select, employ, compose, and monitor Web-based services automatically [13].
Our approach is to first extract the goals from the natural text introduced by the user. For this, we will be using Natural Language processing techniques. A language analyzer will filter the different concepts and relationships in the text, creating a so-called lightweight ontology. Lightweight ontologies are a key-enabling technology for the use of the semantic web [15].

Orchestration comes once those goals have been found. In our approach, a goal is seen as a single execution step. A sequence of goals is composed and sent to the WSMX platform. Spoken on layman terms, the user will have to write down its text and GODO will find the way of extracting the goals, composing them and make sure they reach WSMX, where they will match web services capabilities and will execute certain web services.

2.2. The GODO Architecture

Software is everywhere. It is an integral part of our society that is becoming increasingly larger and more complex. Software development is a difficult activity and many software projects take longer than expected, have large operational failures or are simply cancelled [10]. These problems are one of the main concerns of the software engineer community and a way of decreasing it is to build software models. A model is a simplification of the reality that still retains the elements relevant to our problem. The purpose of the software model is to explain these concepts in a easy way and ignore all unimportant details. There are several models for describing the architecture of software systems, based on the use of multiple, concurrent views, such as the Krutchen "4+1" View Model [11] and those based on the description of each of their components. We will follow the latter approach to capture the gist of the GODO architecture.

The aim of this section is to describe the functionality of the components in the architecture. Loose coupling and reusability of components have been the major intentions behind the architectural decisions and implementation. Some of these details are reflected in the particular components to make them more understandable. Figure 1 depicts the big picture of the architecture.

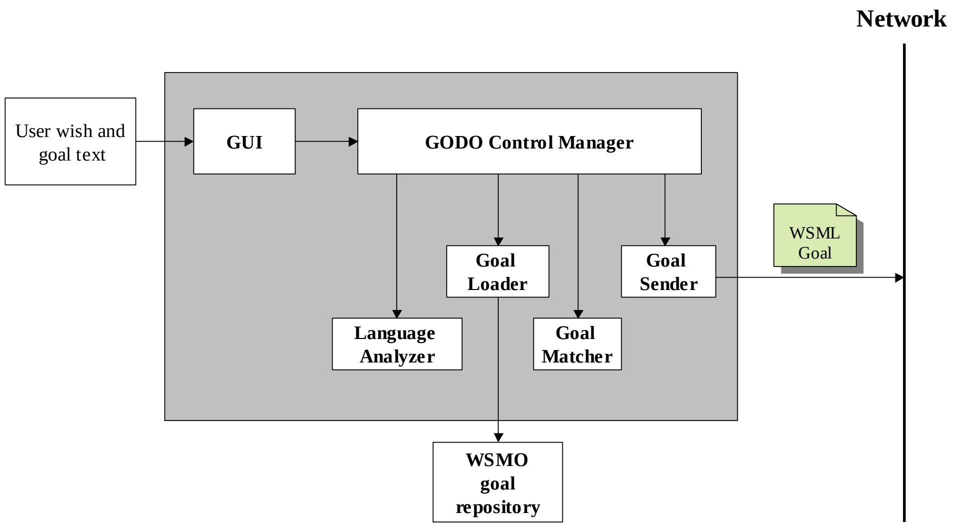


Figure 1. The GODO architecture

2.2.1. Language Analyzer

The task of the Language Analyzer is to filter and process the input introduced by the user in natural language and determine the concepts (attributes and values) and relations included in it.

The Language Analyzer used in this work is based on the methodology presented in [16] to get knowledge from text. This methodology, which uses ontologies and one incremental knowledge acquisition technique termed MCRDR [12], is based on the idea that relationships between concepts are usually associated to verbs in natural language. This methodology uses the mentioned technique and the grammar category of the words in the current sentence to infer other knowledge entities (e.g., concept, attributes and values) in order to create an ontology from a text fragment.

It is necessary a previous phase of training in which the experts establish all the patterns that will use the analyzer to get the concepts and relations within the sentence introduced by the user. This training process is based on the study of a set of texts in natural language related to a specific domain. In this phase, the experts indicate the concepts, concepts' attributes, attributes' values and relations among concepts, and the system stores all this information in the database. After this, when a real user introduces a sentence the system must be able to get the concepts, attributes, values and relations within this sentence taking into account all what it has learned in the training process.

This knowledge acquisition process is divided into three sequential phases, namely the "*POS-Tagging*", "*Concept* search" and "*inference*" phases.

The main objective of the *POS-Tagging* phase is to obtain the grammatical category of each word in the current sentence; for this purpose, the POS-tagger described in [17] is used. In the *Concept search* phase, linguistic expressions representing concepts are identified. The associations between linguistic expressions and concepts have been stored in a conceptual knowledge base obtained in the (previous) training phase of the system. As a result of this phase we obtain all the expressions of the fragment, which are already in the conceptual knowledge base. The last phase, *inference*, is based on the idea that, in natural language, relationships between concepts are usually associated to verbs. The MCRDR component used to obtain the relationships between concepts is formed by a knowledge base containing linguistics expressions representing generic conceptual relationships, and by a subsystem that infers the participants in these relationships.

The modus operandi of this process is as follows. Firstly, the verb in the current sentence is identified. Then, the system searches for the type of semantic relationship associated to that verb. Once the type of relation associated to the main verb in the current sentence has been found, the MCRDR sub-system is applied to extract knowledge by means of the grammatical category of the words, their position in the current sentence, and the type of relation associated to the verb, if any. We will include the following example for further clarification.

Suppose that the Language Analyzer component receives a sentence in natural language as a parameter and returns the concepts (attributes and values) within this sentence and the relations among these concepts (a lightweight ontology). As a result, the ontology resulting from the sentence "I want to buy a plane ticket from Galway to Madrid, then book a hotel in the center and rent a C class car" would be like the one shown in Figure 2. In this figure we can distinguish four concepts ("Subject", "Planet ticket", "Hotel" and "Car") and three relations ("buy", "book" and "rent"). The concept "Subject" is a more general concept than "I" and is directly inferred by the system. The other three concepts have attributes such as "from", "to" and values for these attributes ("Galway", "Madrid")

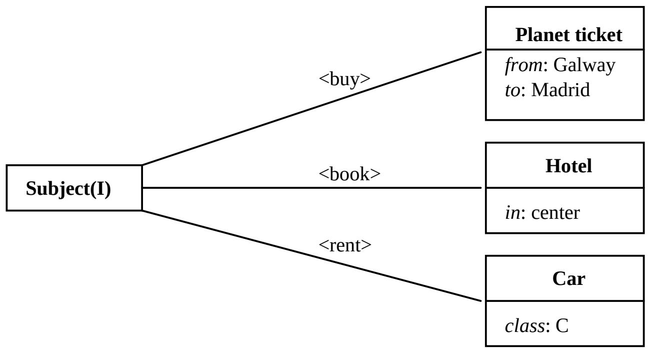


Figure 2. A resulting ontology

2.2.2. Goal Loader

This component looks for the goals in the WSMO goal repository or in an internal file where the goals are stored. This component is outside the architecture so that anybody may plug in his/her own goal repository.
The syntax of these goals is expressed in the Web Services Modeling Language (WSML) [7] used by WSMO as the underlying logical formalism. From our point of view, these goals are just syntactically important given that, at this stage of the implementation, we will be using the inline concepts for our matching.

2.2.3. Goal Matcher

Matching is a widely-used term which is limited in our case to a syntactical scope. The Goal Matcher compares the concepts extracted from the natural language analysis and the ones included in the WSML goals. From this matching, several goals are selected that are composed by the Control Manager in order to build up the orchestration.

2.2.4. Goal Sender

This component sends the different goals to WSMX. Its functionality is quite simple since the orchestration is predefined in the GODO control manager. The goals are sent sequencially, without taking into account any other workflow constructs.

2.2.5. GUI

This is the component that interacts with the user. It collects the users request and presents the results obtained to them.

The software tool developed is a Web application. The application architecture is based on the Model 2 approach in a similar way to that specified in "*The Apache Struts Web Application Framework*"[1]. In fact, it is a variation of the classic Model-View-Controller (MVC) design paradigm. This framework also indicates the way in which the different web technologies must be applied to build a web application.

In this way, the *Controller* is a Servlet, which receives all petitions, determines the action that has to be applied and returns the appropriate view. The *View* is based on the Java Server Pages (JSP) and the user browser. The JSPs get data from the model and compose the HTML page that will be viewed by the user. Finally, for the *Model* the system can interact with several technologies for accessing data like JDBC and EJB.

Briefly, the execution model can be summarized as follows (Figure 3 shows a general scheme of this model): the Controller examines the request of a user, and determines the action to be applied and start its execution. The actions conform the system model. These actions interact with the "Business Logic", retrieve the information asked by the user and pass this information to the View (i.e. the JSP pages) by means of JavaBeans. The JSP page, using that information and (possibly) tags developed to support the graphic designer work, creates the HTML document which will be shown to the user by the browser.

[1] http://struts.apache.org/

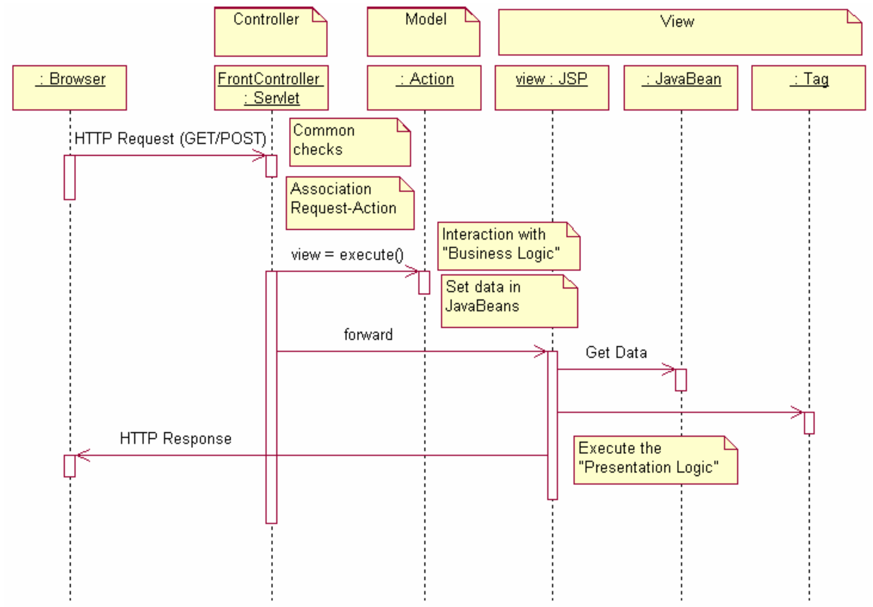


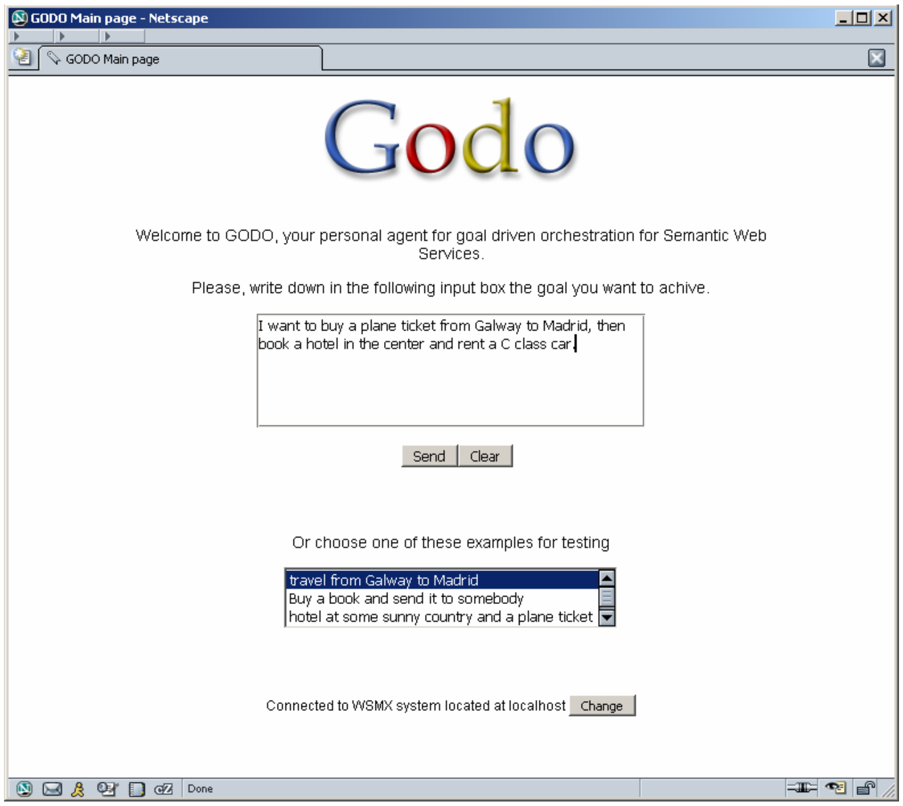
Figure 3. Sequence diagram of the GUI

2.2.6. Godo Control Manager

This is the main component of the architecture. It manages the different interactions among the components. Firstly, it is informed by the GUI that the wish of the user has been described in plain text. Then, it instructs the Language Analyzer to attempt the recognition of the major concepts in the text and communicates with the Goal Loader and the Goal Matcher to orchestrate the different goals that will be sent to WSMX through the Goal Sender. Then, it communicates with the GUI so that the user receives a view of the selected goals and decides if they are correct and comply with his/her expectations. Finally, if the user approves them, they are sequentially sent.

3. Use case

In this section, we will illustrate the advantages of our approach. First of all, we will present a real-world scenario in the next section. In a simple B2C scenario, let us suppose that a customer wants to arrange a trip. In an ideal case, he would specify this wish in natural language, so he will interact with the web-based GODO GUI and fill in with a sentence like this: “I want to buy a plane ticket from Galway to Madrid, then book a hotel in the center and then rent a C class car". A first view of GODO is shown in figure 4.


GODO Main page - Netscape

GODO Main page

Godo

Welcome to GODO, your personal agent for goal driven orchestration for Semantic Web Services.

Please, write down in the following input box the goal you want to achive.

I want to buy a plane ticket from Galway to Madrid, then book a hotel in the center and rent a C class car.

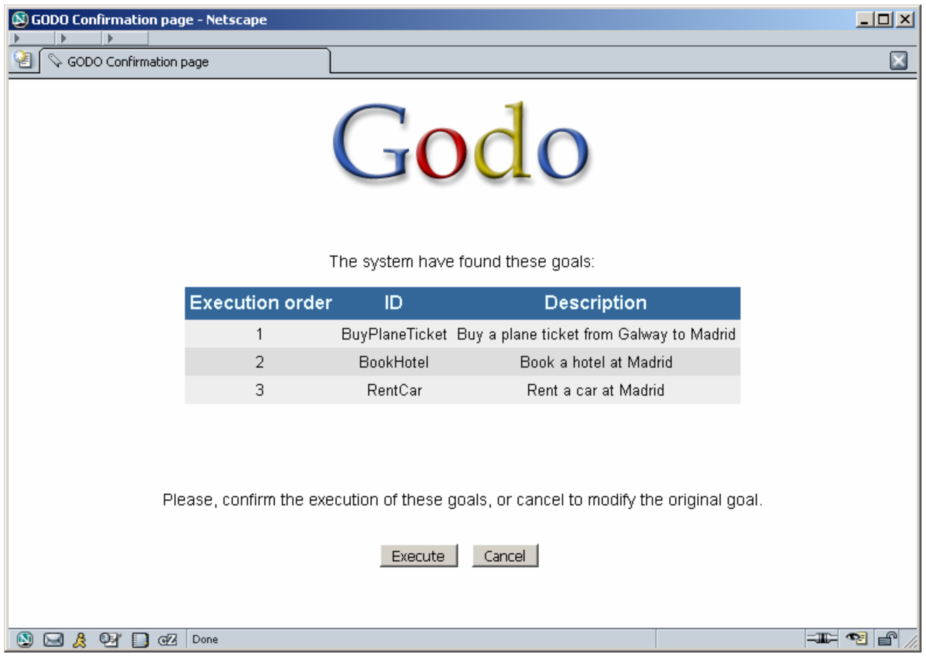
Or choose one of these examples for testing

travel from Galway to Madrid
Buy a book and send it to somebody
hotel at some sunny country and a plane ticket

Connected to WSMX system located at localhost

Figure 4. GODO interaction with the user

Current technologies such as Google [20] would just index these words in a meaning-less way and try to find pure syntactical relationships among them. However, GODO goes further and classifies the goals loaded from the WSMO repository. These goals are chained and orchestrated as depicted in figure 5.



The system have found these goals:

Execution order	ID	Description
1	BuyPlaneTicket	Buy a plane ticket from Galway to Madrid
2	BookHotel	Book a hotel at Madrid
3	RentCar	Rent a car at Madrid

Please, confirm the execution of these goals, or cancel to modify the original goal.

Figure 5. GODO interaction with the user (II)

Finally, these goals will be sent and performed by WSMX once the user has verified and confirmed that this is what (s) he wanted.

4. Conclusions and related work

A similar application is described in [19]. This is an e-commerce system where a user agent uses a natural language processing tool to study a sentence introduced by the user expressing his wishes. When the agent gets the concepts and relations within the sentence, it tries to design a plan with the actions needed to achieve the user final goal. In this case, these actions involve the communication with other intelligent agents that act on behalf of supplier companies. It also uses the Natural Language Processing resources mentioned in section 2.

There is much related work with regards to NLP resources transforming natural language text into semi-ontological structures. TextoOnto[22] supports semi-automatic creation of ontologies by applying text mining algorithms. Currently it includes a term extraction algorithm, a concept association extraction algorithm and an ontology pruning algorithm. This tool aids users in creating and maintaining ontologies through the application of text-mining algorithms in such a way that it helps detecting concepts and relationships that the ontology engineer has initially missed, but that may be inferred from texts about the domain being modeled. ASIUM [4] is a clustering and cooperative methods-based system, which takes the syntactic analysis of sentences as the input such that the phrases are identified ASIUM creates an ontology given a text as input and a set of case frames where the semantic features are filled by ontology concepts. Also, it uses a graphic interface to allow the user to inspect, validate and refine the knowledge learned at each step of the learning process. There are some other efforts to combine interactive extraction and extension of ontologies from text, like OntoLT [21]. This application provides a Protégé plugin and an environment for the integration of linguistic analysis in ontology engineering through the definition of mapping rules that map linguistic entities in annotated text collections to concept and attribute candidates (i.e. Protégé classes and slots). A relevant work in this area is the OntoText Knowledge Information Management (KIM) [8] platform. KIM enables Semantic annotation of text. It also allows for automatic ontology population and open-domain dynamic semantic annotation of (unstructured or semi-structured) content for Semantic Web and Knowledge Management applications where indexing, retrieval, query and exploration of formal knowledge is feasible.

There are several approaches regarding orchestration. In WSMO terms, orchestration [2] decomposes a capability (i.e., the functionality of a web service) in terms of functionality required by other services, namely other providers view. This could be applied to current business process execution initiatives. In our approach, we have worked with an extended vision of orchestration.

Finally, our future work will focus on determining the feasibility of a semantic match of lightweight ontologies extracted from natural language text and ontologies defining goals in particular domains. This work is related to existing efforts about ontology merging and alignment. Future version of GODO will be oriented towards that direction. Also, ways to overcome the limitations of having one single step execution per goal will be explored.

5. Acknowledgment

The work is funded by the European Commission under the projects DIP, Knowledge Web, Ontoweb, SEKT and SWWS; by Science Foundation Ireland under the DERI-Lion project; and by the Austrian government under the CoOperate Programm.

We thank the Spanish Ministry for Science and Technology for its support for the development of the system through projects TIC2002-03879, FIT-110100-2003-73 and FIT-150500-2003-503. This work has also been partially funded by the Spanish Ministry of Science and Technology, in the context of the project TIC2002-1948

6. References

[1] C. Bussler: B2B Integration, Concepts and Architecture. *Springer-Verlag*, 2003, ISBN 3-540-43487-9.

[2] D. Roman, H. Lausen, U. Keller: Web Service Modeling Ontology Standard. WSMO Working Draft v02, 2004. Available from http://www.wsmo.org/2004/d2/v02/20040306/.

[3] E. Oren, M. Zaremba, M. Moran: Overview and Scope of WSMX. WSMX Working Draft, 2004. Available from http://www.wsmo.org/2004/d13/d13.0/v0.1/20040611/.

[4] Faure D. and Nédellec C. *Knowledge acquisition of predicate argument structures from technical texts using Machine Learning: the system ASIUM*. In Dieter Fensel Rudi Studer, editor, 11th European Workshop EKAW'99, pages 329-334, Springer-Verlag, May 1999

[5] J. Bruijn, D. Foxvog, E. Oren, D. Fensel: WSML-Core, Working Draft, 2004.

[6] D. Fensel and C. Bussler: The Web Service Modeling Framework WSMF. *Electronic Commerce Research and Applications*, Vol. 1, Issue 2, Elsevier Science B.V.

[7] E. Cimpian, A. Mocan, M. Moran, E. Oren, M. Zaremba: WSMX Conceptual Model. WSMX Working Draft, 2004. Available from http://www.wsmo.org/2004/d13/d13.1/v0.1/.

[8] Popov.B, Kiryakov. A. *Towards Semantic Web Information Extraction.*Human Language Technologies Workshop at the 2nd International Semantic Web Conference (ISWC2003). Florida,USA..

[9] OWL-S: Semantic Markup for Web Services.

[10]W.Wayt Gibbs. Software chronic crisis. Scientific American, pages 72-81. September, 94

[11]P. Krutchen: The “4+1” View Model of Software Architecture. IEEE Software, 12, pages 42-50.November,95

[12]Kang, B (1996). Multiple classification ripple down rules. PhD Thesis, University of New South Wales.

[13]McIllraith,Sheila.. Semantic Web Services. XML-Web Services. ONE conference. June,202

[14]D. Fensel. *Ontologies: A silverbullet for Knowledge Management and Electronic Commerce.* 2nd Edition, Springer 2003.

[15] Gartner Research Note (ID=T-17-5338). “Semantic Web Technologies Take Middleware to Next Level”. Autores: Jim Jacobs y Alexander Lind

[16] Valencia-García R., Ruiz-Sánchez J.M., Vivancos-Vicente P.J., Fernández-Breis J.T., Martínez-Béjar R. (2004) *An incremental approach for discovering medical knowledge from texts. Expert Systems* With Applications, VOL. 26, Nº 3, 291—299

[17] Ruiz Sánchez, J. M., Valencia García, R., Fernández Breis, J. T., Martínez Béjar, R. & Compton, P. (2003). An approach for incremental knowledge acquisition from text. *Expert Systems with Applications*, 25, 77-86.

[18] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. Scientific American, May 2001. http://www.scientificamerican.com/issue.cfm?issuedate=May-01

[19] García-Sánchez, F. (2003).*Diseño e implementación de un entorno para desarrollo de aplicaciones de comercio electrónico basados en teoría de la decisión y tecnologías del conocimiento.* Master Thesis, University of Murcia. (in Spanish)

[20] Google. Inc. (www.google.com)

[21] Protégé Plug-In for Ontology Extraction from Text Based on Linguistic Analysis. Paul Buitelaar, Daniel Olejnik, Michael SintekIn: Proceedings of the 1st European Semantic Web Symposium (ESWS), Heraklion, Greece, May 2004.

[22] TextOnto: http://kaon.semanticweb.org/Members/rvo/Module.2002-08-22.4934

.