2009 International Conference on Signals, Circuits and Systems

Steam Turbine Diagnostic System based on a Domain Ontology Implemented using J2EE Technology

Klai Sihem, Khadir M. Tarek, and John V. Ringwood, Member, IEEE

Abstract— This paper describes the design and the implementation of an aided diagnostic system based on a domain ontology and an expert system. A mono ontology obtained from multiple CCOs (Canonical Conceptual Ontologies) constitutes the knowledge base of the Expert System. The NCCO (Non Canonical Conceptual Ontology) are defined and used for realising inter-CCO mapping and to express the relationships between the COO. An expert system JESS (Java Expert System Shell) is integrated with the ontology. Knowledge inference is then possible to solve maintenance cases by given adequate diagnoses to given symptoms. The second aspect of the paper focuses on the possible enhancement and evolution of the developed ontology in order to take into account new maintenance cases. The complete system is developed following a J2EE (Java 2 Enterprise Edition) architecture.

***Keywords*— Expert System, Ontology Construction, OntologyEvolution, Protégé, OWL, JDBC, J2EE.**

I. Introduction

Knowledge management systems (KMS) are imposing themselves as a very powerful way of transforming knowledge resources into intellectual, machine readable, capital ready to use for a number of targeted applications. Davenport and Prusak [1] define knowledge as a fluid mixture of experience, values, contextual information, and expert insight that provides a framework for evaluating and incorporating new experiences and information. Wache in [2] proposed three main ontology based approaches: single, multiple and hybrid.

- Single ontology approaches uses a global ontology providing a shared vocabulary for the specification of the semantics. It can be applied to integration problems where all information sources to be integrated provide nearly the same view on a domain. But if one information source has a different view on a domain, becomes a difficult task. Also, single ontology approaches are susceptible for changes in the information sources which can affect the conceptualization of the domain represented in the ontology.
- Multiple ontologies: each information source is described by its own ontology. The semantic of an information source is described by a separate ontology.
- Hybrid approaches: To overcome the drawbacks of the single or multiple ontology approaches, hybrid approaches similar to multiple ontology approaches, the semantics of each source is described by its own ontology. But in order to make the local ontologies comparable to each other a global shared ontology is built. The advantage of a hybrid approach is that new sources can easily be added without the need of modification. It also supports the acquisition and evolution of ontologies.

According to a comparative analysis presented in [2], the hybrid approach is the most relevant approach because it allows semantic heterogeneity and flexibility. The mono-ontological approach is simple to realize but it cannot be used for semantically heterogeneous databases. Multiple ontologies and hybrid approaches are confronted to the heterogeneity problem [3]. In this case, it becomes necessary to integrate and make mappings inter-ontologies. In this paper, a mono-ontology constructed from multiple CCOs with integration of semantically heterogeneous data bases, ensuring at the same time implementation flexibility and simplicity, is possible. Each CCO is associated to a data base source. The definition of the NCCO between concepts and the instances belonging to different CCO allows mappings between the various CCOs. The definition of the NCCO consists in seeking and representing correspondences between the various integrated data schemes. It can be made in a manual, semi-automatic or automatic way. An expert system JESS [4] is merged with the ontology to deduce from new relations, starting from the existing concepts, to represent the mappings automatically or in a semi-automatic way. To test the feasibility of the approach suggested, we built a steam turbine domain ontology by integrating two relational data bases with different usage. One relates to the characteristics of the equipment of the steam turbine and the other relates to the various cases of maintenance defining the symptoms, the defects, the causes and the remedies for each case. The objective set in this paper is, on one hand is the proposition of a new mono-ontology multiple-CCO databases integration approach, leading to the construction of steam Turbine ontology. On the other hand the use of the obtained ontology as a knowledge base for knowledge inference and diagnostic purposes. Ultimately, the exploitation and evolution of the obtained ontology is assured by some instructions of JESS language and, it is

Klai S. and Khadir M.T are with the LabGED (Laboratoire de Gestion Electronique de Documents, University of Badji Mokhtar Annaba, Algeria. phone: 0021338872756; e-mail: khadir@ labged.net).
Ringwood J.V. is with NUI Maynooth, Ireland, (Department of Electronic Engineering, NUI Maynooth Co.Kildare; e-mail: john.ringwood@eeng.may.ie).

978-1-4244-4398-7/09/$25.00 ©2009 IEEE

independent from the ontology editor. This is implemented using J2EE technology.

The rest of the paper focuses on a definition and a taxonomy of domain ontologies, Section 2. The proposed approach is detailed in Section 3. The semi automated development of the steam turbine ontology based on proposed approach is detailed in Section 4. Section 5, shows how the ontology may be maintained by evolving in order to suit any new brake down and diagnoses. Finally, conclusions are drawn in section 6.

II. Ontologies

Gruber [5] defines the ontology as: "An ontology is an explicit specification of a conceptualization". The type of an ontology is closely related to its conceptualization objects such as: knowledge representation high level, generic, domain and application. In our case the developed ontology is of domain type, as it contains a number of concepts and a certain vocabulary that defines a targeted domain i.e., the steam turbine and its maintenance aspects.

A. taxonomy of domain ontologies

Domain ontologies, may be divided into two main categories [6]: Linguistic ontologies (LO) with the objective of a multi lingual and conceptual representation. Those ontologies define words or contextual usages of words. Conceptual ontologies (CO), represents the domain objects and proprieties. However within the CO, different ontologies may present different characteristics according to the field of application and the ontology model upon which they were constructed. According to the definition and the distinction between the primitive concepts and the defined concepts, [5] the CO category is divided into 2 categories: Canonical Conceptual Ontology (CCO) which contains only the primitive concepts and Non Canonical Conceptual Ontology (NCCO) which contains the primitive and the defined concepts.

B. A layered model for ontology design

As specified in [7] the previous observations lead to identify some relationships between CCOs, NCCOs and LOs.

- Mappings between CCO might also be defined of equivalence operators of some NCCO;
- NCCO models can use powerful CCO oriented model constructs to define their own primitives concepts;
- Los might define the various meaning of each word of a particular language by reference to a NCCO. This reference would provide a basis for formal and exact reasoning and automatic translation of context-specific terms.

III. Steam turbine ontology

Steam turbines are mechanical devices using supper-heated steam power, and convert it into useful mechanical work. In the studied case, the mechanical work produced is used for electrical production.

Ontologies have been used to represent knowledge and help knowledge inference in the industrial field. As an example, the PROTEUS platform [8] adopted an ontological representation with Cased Based Reasoning (CBR) as model solving problems. Another example is shown in [9] where the authors proposed a collaborative environment allowing users (experts and beginners) to share knowledge through a shared ontology for the maintenance of a steam turbine. In [7], ontologies as a domain model allowing solutions for various issues in data indexing, data exchange and data integration are presented

A. Ontology construction using multiple CCO

We propose a single ontology approaches with multiple CCO for domain ontology construction. Each data base is described by its own CCO. The semantic of an information source is described by a separate CCO. The NCCO are defined and used for realising inter-CCO mapping. The main advantage of this approach is simplicity and flexibility permitting the addition of new data sources without the need of modification. It also permits to integrate two or more semantically heterogeneous databases. A diagram of the described approach is given in Figure 1. It exploits the NCCO capability to define equivalent concepts and thus to integrate several CCO addressing the same domain. Three issues may then be addressed.

Mapping discovery: Given two CCO, how do we find similarities between them, determine witch concepts and properties represent similar notions? We can use various characteristics of CCO, such as their structures, definitions of concepts, instances of classes to find mappings.

Representation of mappings (NCCO): Given two CCO, how do we represent the mappings between them to enable reasoning with mappings? The representation of the inter-COO mappings consists in defining the NCCO which can be the OWL constructor or the equivalence relation defined, starting from the logical rules between the concepts belonging to different CCO. We propose, also, to use inference engine to assert the NCCO automatically starting from the logical rules between concepts belonging to different CCO.

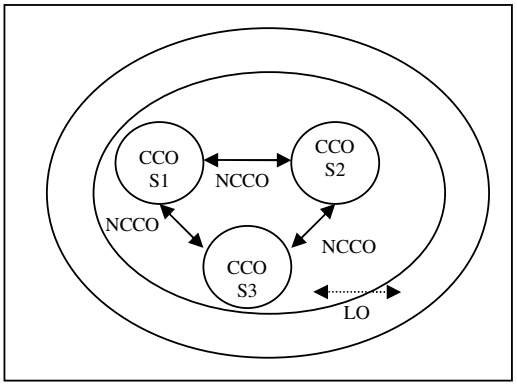

Figure 1. Single ontology approach with multiple-CCOs

Using Mapping (NCCO): Defining the mapping between CCO, either automatically or interactively is not a goal in itself. The resulting mappings are used for various integration tasks: to answer the requests users. In this case, this may be realised by using the JESS instructions language.

.

B. Steam turbine ontology conceptualisation

As for the present work, the system knowledge is contained in two databases. Each data base is described by a CCO, for instance CCOS1 and CCOS2 for the first and second data source respectively. The sum of both CCOs becomes the consensual CCO. In other words, all source data bases available, are merged to form unique domain ontology. The canonical vocabulary of each data base becomes a sub group of the consensual CCO. The NCCO operators in this case, will be are the OWL constructors, the equivalent relation or the specific relation of the domain to be created between concepts belonging to different CCOs.

The implementation of the proposed construction approach is realised using the Protégé Plug-in DaTaMaster [10]. It was developed for importing data base schemes and their contents in Protégé under OWL. DataMaster permits the integration of multiple data bases in a single ontological representation.

The implementation followed the steps below:

- The choice of the connection driver types: Open Data Base Connectivity (ODBC) or Java Databases Connectivity (JDBC) and the data source.
- The selection of a given table activates the visualisation of its content, then the user has the choice of importing the table or not.
- The chosen data base tables are activated and visualised, each table is transferred into one class or sub-class depending on the user's choice.

C. Steam turbine ontology construction

The insertion of OWL constructors (owl: unionof,..), relations as well as annotations participate in the process of semantically enhance data belong to different CCO as well as to solves syntaxes and semantic heterogeneity of integrated systems, improving data exchange between them.

Conflict context, was resolved by assigning a unique space name to each CCO. Thus, all the classes, attributes as well as instances belonging to one CCO are pre-tagged with the same Uniform Resource Identifier (URI).

Name conflict was resolved by defining semantic relationships between concepts. As an example, two classes issued from two different data sources (different CCO), where the first one describes the equipment (code, designation, function, zone etc.) and the second details different maintenance cases (code, defects, causes, symptoms and remedies). Both classes treat the same equipment, but are not structured in the same manner and the semantic of their date are different. A relation "equivalent1" was created between the two classes and the same time between both instances.

This method is manual and may be long for a large ontology. Automatic definition of the NCCO is implemented with JESS instructions, then we can discover that the code is a common attribute, thus we can carry out the following algorithm:

- For each instance of CCO1 and for each instance of CCO2;
- If instance.CCO1.code = instance.CCO2.code then
- Assert (create in JESS) property "equivalent1" between instance.CCO1 and instance.CCO2

The hierarchical structure of the steam turbine ontology is then shown in Figure 2.

D. Steam turbine Ontology operations

JESS, is a rule based reasoning engine that can be used with the ontology instances. The OWL2JESS tool [11], permits the conversion of the OWL ontology code to JESS facts. The semantic predefined rules RDFS and OWL are used to verify the coherence and uniformity of the ontology [12]. This will permit to designed, evaluates and refine the original obtained ontology. The obtained knowledge base encapsulates three layers of knowledge: the ontology model layer, the ontology layer and instances layer. The instance layer is represented in Figure 3, the detail of the others layers are discussed in [13]. The JEES facts are of triplets type given by (Predicate, Subject, Object).

E. Reasoning with mapping using JESS

The exploitation of the ontology is ensured using rules and requests permitting to fetch the knowledge base through a set of JESS commands such as "defrule" and "defquery".

Below an example of using such commands to display the instances equivalent with the instances of the class topo:30BB prefixed with its URI.

```
(defrule instequivalent
(triple(predicate     "http://-rdf-syntax-
ns#type")(subject ?s)
(object"http://jdbc:odbc:topo#_30BB"))
 (triple(predicate"http://owl#equivalent1"
  )(subject ?ss&:(eq ?ss ?s))(object ?o))
=>(printout t ?s " " ?o crlf ))
```

The relation « equivalent1 », being definite transitive, is the rule which allows the inferred triplets as follows:

```
(defrule equivalence
(triple(predicate"http://owl#equivalent1"
)(subject ?x) (object ?y))
(triple(predicate"http://owl#equivalent1"
)(subject ?y) (object ?z))
   => (assert (triple (predicate
"http://www.owl#equivalent1")(subject ?x)
(object ?z))))
```

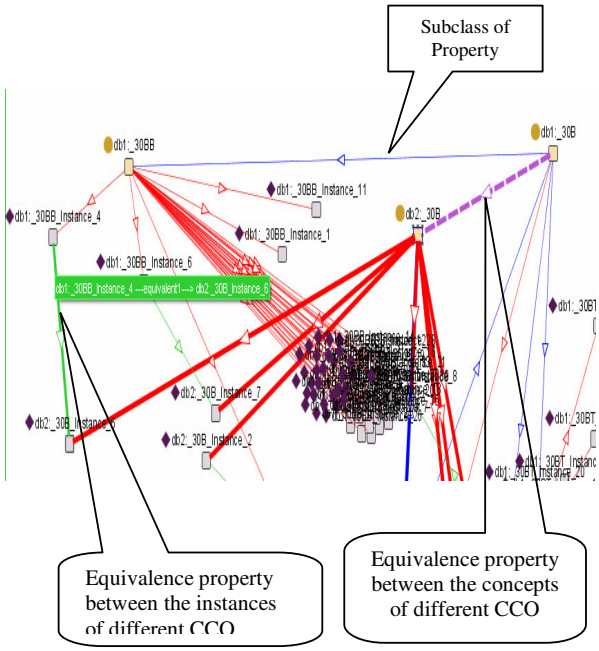

Figure 2. Hierarchical structure of Steam turbine ontology

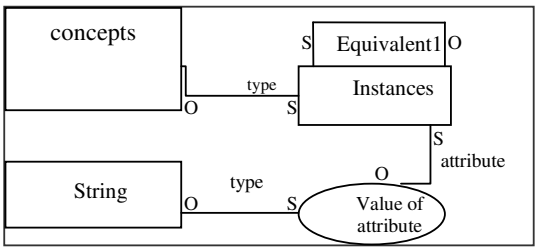

Figure 3. Knowledge representation of instances

IV. ONTOLOGY EVOLUTION

In the ontology cycle, ontology evolution is a crucial aspect that needs to be addressed. Ontology evolution is defined as the process of updating the previous ontology version, in order to take into consideration different evolvement of the domain in its conceptualization or application. The evolution process creates a version *n+1* from the original version *n*. The process is not straight forward and may not be completed manually, as the uniformity and coherence of the ontology must be respected.

➢ *Methodology*

There are no mature explicit methodologies for ontology evolution. However steps can be found in order to elaborate an ontology evolution strategy, given generally by identifying of the domain changes that occurred. There are two main changes identification methods:

- **Descending identification:** Those are imposed by the definition or domain update or the update of the ontology or instances usage.
- **Ascending identification:** Those are changes identified from the ontology analysis it self. For example, by using heuristic rules for ontology optimization.

Editing ontology changes may be complex or elementary.

Elementary changes for a non decomposable change given by a suppression or adding of ontological entities [14] and complex changes composed or 2 or more elementary changes forming together a logical entity [15], e.g., the fusion of a number of concepts in one.

The addition of a new concept must be followed by the insertion of links belongings to the concept along with other existing concepts. The suppression of a concept might cause the isolation of one other concept or more. All conceptual or semantic relations, linking the suppressed concept with other concepts must then be deleted along with linked instances. The addition of a new concept must follow the following rules:

- Checking changes made: This step investigates the effects of the changes made on the ontology coherence.
- Validation of changes: changes must be validated, especially in the case of concept fusion or suppression.
- Changes implementation: This must ensure the archiving of the previous version and the preservation of changes made, and metadata associated at the end of this step. A new version of the ontology n+1 is created.
- Analysing the compatibility between the version *n* and *n+1*. This step identifies incompatibilities caused by changes as well as the effects of changes on the compatibilities of version *n+1* in terms of preservation of the ontology roles.
- Validation of the version *n+1* in the community: this constitute of the collective validation of the new version by the community of practice.

➢ *Evolution of the steam turbine ontology*

The operational steam turbine ontology is saved as a text file (knowledge base) on the forma of a triplet set (Predicate, Subject, Object) presented earlier in section

The evolution of this base consists in the elaboration of the addition, suppression and modification operations on the knowledge base using the JESS language "**assert"** for addition, "**modify"** for modification, and "**retract"** for suppression. The development of a system which takes in charge the propagation of changes automatically is compulsory. In addition the system must guide the user during the operation in order to make changes in a transparent manner. The original ontology may be saved under a name showing the version and the process of the changes made. This way, the history of the different versions of the ontology may be saved. The change may be implemented on different levels:

- On the ontological level: Adding, suppressing and changing a concept, an attribute, a relation or a semantic relation and updating conceptual relations.

- On the instances level: Adding, suppressing and changing an instance. The knowledge representation of instances is given in figure 3.

➢ *Examples of ontology evolution on the steam turbine*

Adding an instance and the changes induced. The JESS instruction for changes at a conceptual level for concepts adding and induced changes presented in [16]. Here is an example of the JESS instructions on order to implement a change at an instances level:

- The user selects the concepts "30BB" and writes the instances to be created:

```
(assert (triple
(Predicate "http://www.w3.org/1999/02/22-rdf-
syntax-ns#type")
(Subject
"http://biostorm.stanford.edu/db_table_classe
s?DSN=jdbc:odbc:diagnostic#_30BB_Instance_40"
)
(object
"http://biostorm.stanford.edu/db_table_classe
s?DSN=jdbc:odbc:diagnostic#_30BB")
))
```

V. DATABASE MAINTENANCE

The last component of proposed system is related to the databases maintenance, which is ensured by updating and semantic enriching.
This database maintenance is executed after an ontology evolution is performed. The objective of this maintenance is to ensure that new instances and property values are consistent with the ontology semantic. Once this is done views may then be created.

Views can join multiples classes belonging to different CCO in to a single virtual table stored in a source databases. This possibility allows each user to have their separate perception of various integrated data bases. It also, offers a large flexibility to user's access.

The proposed system uses JESS (Java Expert System Shell) and JDBC [17]. It follows the following execution steps shown Figure 5:

- Select a class and executes a JESS program which provides the equivalent classes ;
- Executes a JESS program to restore the classes attributes concerned by the first phase, followed by the corresponding instances;
- Establish a connection with the source data base with the Java instruction "DraverManager.getConnection()";
- Once connection is established with the source data base, SQL request for updating and creating of views, starting from the Java program by carrying out the instruction "executeupdate(query) .

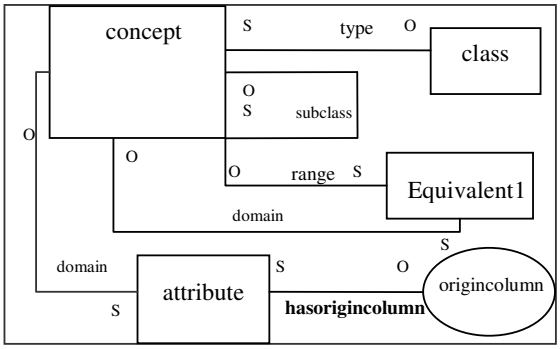

Figure 4. Knowledge representation ofontology

In Figure 4, the property *"hasorigincolumn"* allows a mapping between the ontology and a source database to be updated.

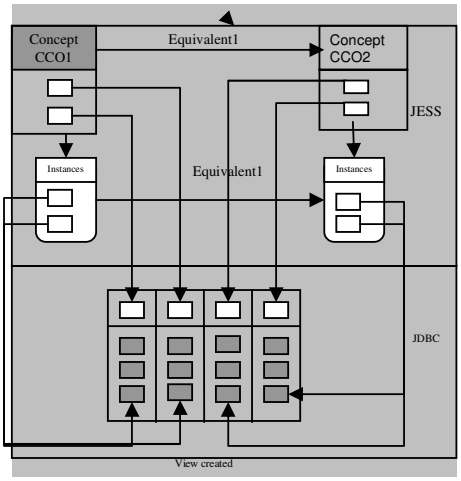

Figure 5. View creation in a source databases from the ontology

VI. J2EE IMPLEMENTATION OF THE SYSTEM

In the development so far, the user must know the JESS language in order to address a requests to the knowledge base. However, this reveals to be very restrictive. It is then paramount to be able to address request in a transparent way. One of the many JESS advantages is that it remains a scripted environment used for the construction of a JAVA based applications. We will take advantage of that propriety to develop JAVA based GUI using the J2EE technology that permits uploading and executing the Expert System operations and then renders results in order to be saved or displayed to the user. The J2EE implementation follows the block diagram given in Figure 4. While the GUI interface is given in Figure 5. The user may browse and requests any diagnostic represented by the previously developed knowledge base and inference rules implemented in the diagnostic and ontology modules. He may as well include new maintenance case by enhancing the existing ontology and creating new instances.

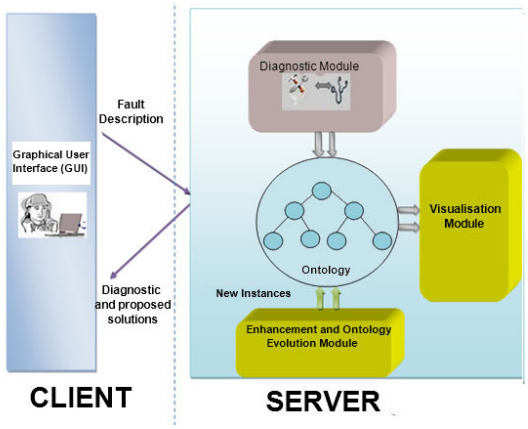

Figure 6. Diagnostic system built atound the steam turbine domain ontology

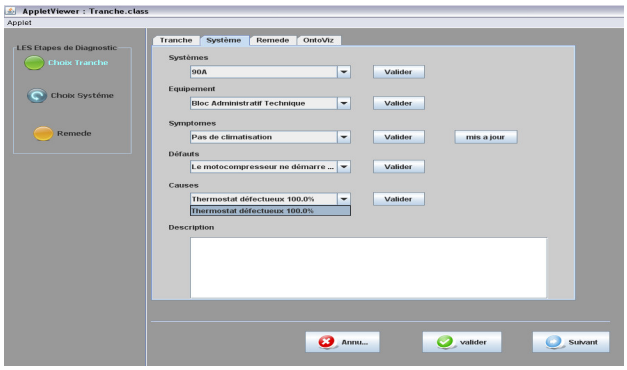

Figure 7. GUI for the aided maintenance diagnostic system

VII. CONCLUSION

Domain ontology is a very powerful knowledge management approach, providing a unified domain conceptualisation. In this paper, a domain ontology with multiple CCOs based on knowledge extraction from two different data sources, which enhanced semantically the final result was presented, and implemented using J2EE technology. The semi-automatic ontology construction helped greatly in terms of time and effort saving. It allowed a faster and more efficient construction than a manual time consuming one. In order to define automatically the NCCO from the logical rules, an expert system is integrated as an inference module. A strategy for ontology evolution at a conceptual, relational and instances levels are presented in order to update the knowledge base for better diagnoses and maintenance, adding new cases in terms of symptoms defects, diagnoses and remedies.

References

[1] Davenport, T.H. and Prusak, L. "Organizations Manage What They Know", Boston, MA: Harvard Business School Press, 1998.

[2] Wache H., Vogele T., Visser U., Stuckenschmidt, H., Schuster, G. Neumann, H., and Hubner, S., "Ontology-Based Integration of Information- A Survey of Existing Approaches". In Stuckenschmidt, H., editor, IJCAI-2001 Workshop on Ontologies and Information Sharing, 2001.

[3] Klein M., "Combining and relating ontologies: an analysis of problems and solutions". In IJCAI Workshop on Ontologies and Information Sharing, pages 53-62, Seattle, Wa, 2001.

[4] Friedman-Hill, E., "*JESS in action* : Rule-Based system in Java", Manning publications co., ISBN:1930110893, 2003, pp. 480.

[5] Gruber, T.R.1, "Translation approach to portable ontology specifications". *Knowledge Acquisit, 5(2)*, 1993, pp. 199-220.

[6] Cullot, N., Parent, C., Spaccapietra, S., and Vangenot, C., "Ontologies": A contribution to the lst International Workshop on semantic Web and Databases (SWDB'03), 2003, pp 109-129.

[7] Jean, S., Pierra, G., Ait Ameur, Y., "A Domain Ontologies : A Database-Oriented Analysis", Lecture Notes in Business Information Processing, Springer Berlin Heidelberg , 2007, pages 238-254.

[8] Rasovska, I., Chebel-Morello, B., Zerhouni, N., () Process of s-maintenance: decision support system for maintenance intervention, *10th IEE E* Conference on Emerging Technologies and Factory Automation, *ETFA 2005,* Catania, Italy, 2005, *pp.8-15.*

[9] Khadir, M-T, and Sellami, M., "A Web-based Collaborative Environment Based on a Shared Ontology for the Maintenance of Steam Turbines". In 7th Computer Information Systems and Industrial Management Applications, Ostrava, The Czech Republic, 2008, pp151-152.

[10] Nyulas, C., O'Connor, M., Samson Tu, "DataMaster – a Plug-in for Importing Schemas and Data from Relational Databases into Protégé Stanford Medical Informatics", Stanford University School of Medicine, Stanford, CA 94305, 2007 .

[11] Mei, J., Bontas, E.P., and Lin, Z., "OWL2Jess: A Transformational Implementation of the OWL Semantics". Lecture notes in computer science, Springer-Verlag, N 3759, 2005, pp. 599-608.

[12] Mei, J., Transformation Implementation of OWL Semantic, 2005, http://www.ag_nbi.de/research/owltrans/

[13] Khadir M.T., Klai S., "A web-Based Fault Diagnostic Maintenance System for steam Turbines Based on a domain Ontology and Expert System", International Conference on Multimedia Computing and Systems (ICMCS' 09) Ouarzazate, Morocco, 2009.

[14] Stanjanovic, L., "Methods and Tools for Ontology Evolution". Ph.D Thesis, University of Karlsruhne, 2004.

[15] Giorgos F., Dimitris P., Grigoris A., (2006). Evolving Ontology Evolution. Proceedings of the 32nd International Conference on Current Trends in Theory and Practice of Computer science (SOFSEM 06).

[16] Khadir M.T, Klai S., Ontology Construction and Evolution for a steam Turbine Diagnostic Maintenance System, (2009) in International Conference on Recent Advances in Intelligent Information Systems (IIS'09), Krakow, Poland, ISBN 978-83-60434-59-8, pages 399-411.

[17] Sun, Java Database Connectivity (JDBC), http://java.sun.com/products/jdbc/,2005.