A Distributed Environment for Development and Deployment of Ontologies of Knowledge-Based Systems

Sameh El-Ansary , Ahmed Rafea
Computer Science Deaprtment
The American University in Cairo (AUC),
113 Kasr El-Aini Street
Cairo, Egypt
sansary@sics.se, rafea@aucegypt.edu

Abstract:

Creating ontologies for scientific, natural or business domains is a process by itself while the development of knowledge bases that depend on those ontologies is another important process. Nevertheless, the research focus in the field of ontology is on the first process, namely, creating ontologies. The aim of the presented research is to improve the second process, namely, building knowledge bases that depend on well-established ontologies. Practically, the aim of the work is to enhance the process of knowledge-based systems construction by providing a common integrated environment for the ontologies development and deployment for knowledge based systems. The environment covers many aspects related to the content, the editing facilities and exchange of knowledge. The eXtensible Markup Language (XML) was chosen as the medium for persisting and exchanging knowledge.

1 INTRODUCTION

The presented research work focuses on enhancing the process of knowledge-based systems construction by providing an integrated ontology development and deployment environment for knowledge based systems. The environment covers many aspects related to the content, the editing facilities and exchange of knowledge. The eXtensible Markup Language (XML) was chosen as the medium for persisting and exchanging knowledge, as it became a de facto standard for information exchange in many disciplines.

An ontology provides a set of concepts and terms for describing some domain, while a knowledge base uses those terms to represent what is true about a real or hypothetical world [1]. For example, a common ontology defines a vocabulary with which queries and assertions could be exchanged among a collection of knowledge-based systems, which share vocabulary in a coherent and consistent manner. Despite of that, they do not necessarily need to share the same knowledge base [2].

The main motivation behind the presented research work was the problems that were faced during constructing knowledge bases in the Central Laboratory for Agricultural Expert Systems (CLAES). Several agricultural expert systems have been developed using commonKADS methodology [3],[4]. These expert systems have the same basic ontology. However, we found difficulty in sharing or reusing this ontology because parallel efforts were initiated to build different expert systems and hence we ended up with ontologies for each of these expert systems with inconsistent terminology. So, the first problem we were facing was what ontology we should use when we start building a new expert system and what to do if we want to integrate existing expert system in one package. Another problem faced was the lack of collaboration during developing the expert systems as each developer is adding, modifying and/or deleting in the ontology individually. A third problem was that we are using several knowledge representation languages [5] and hence how to integrate two expert systems developed using different languages.

The presented research work concentrates on making use of ontology repositories in contributing to the solution of the previous problems Thus, the research goal could be summarized as follows: Improving the performance of Knowledge Bases construction via an XML-based distributed environment for Ontology development and deployment.

2 RELATED WORK

The most relevant area of research to the presented work is the field of ontological engineering, which is primarily concerned with the methodology or the process of building ontologies [6]. The ontology editing environments and the ontology modeling in particular are two aspects that represent the focus of the presented work in the field of ontological engineering. For the ontology editing environments, there are numerous systems of varying degrees of complexity. Some of those are: The Ontolingua system [7] of Stanford University. It provides a repository of ontologies and designed to allow several users to cooperate in developing an ontology collaboratively. Ontolinuga is also the name of an ontology modeling language. WebOnto [8] is another collaborative tool for browsing, creation and editing of ontologies.

Several other non-collaborative environments exist such as Protégé [9] and ODE (Ontology Design Environment) [10]. For the ontology modeling languages, there are three general trends [5]: The first trend is to used well-established knowledge representation languages and extend them if needed and this is the most widely used trend. Examples of that trend include: Ontolinuga [7], KIF [11],[12], Loom [13], [14] Cycl [15] and Flogic [16]. The second trend is to use mark-up languages as a format for representing knowledge. Examples of that trend include: XOL [17], OML [18], Onto-RDF [19], OIL [20]. The reader is referred to [6] for a survey of existing research in the field of ontology.

3 THE PRESENTED APPROACH

The research approach in the presented work draws a clear line between ontology development and ontology deployment. Thus, the strategy of the presented environment is to continue having an ontology-editing environment like Ontolingua and WebOnto and add to this environment a second aspect, which is Ontology deployment. For Ontology modelling, XML was used to encode ontological entities.

The adopted approach solves the problems listed in section 1. Having a common ontology for various knowledge-based systems provides a vehicle for constructing consistent knowledge bases in terms of terminology and design. This also contributes in decreasing the repeated efforts exerted to build knowledge bases from scratch. In the proposed research, we propose an ontology server for providing the common ontologies in a distributed environment. The intended form of representation for the repository of ontologies is XML structures. Due to the structural nature of XML and its high interchangeability, representing ontologies in XML would provide syntactical standardization well suited for ontologies cross-language exchange when there is an agreement on their intended semantics. The research work provides a protocol based on XML also to act as an API for any other software components that need to work with the repository of ontologies. The protocol supports querying the ontology and taking portions or complete snapshots of ontologies residing on the server.

Ontology translation was a straightforward outcome of the approach of adopting the XML as a storage medium. XML parsing is achieved via the DOM API, which is available in mostly all programming languages, thus, translating to other representation languages is a relatively easier process.

4 ENVIRONMENT ARCHITECTURE

As illustrated in figure 1, the environment consists of a central server, where both ontologies and knowledge bases reside, in addition to a number of clients that are used by the environment users. The central server serves two types of users: the *ontology engineers* and the *knowledge engineers*, each working on his corresponding side of the server. The side of the server that the ontology engineers interact with is called the *development* side. On the other hand, the knowledge engineers interact with the *deployment* side. The word "side" in that context means a group of services offered to clients of the server.

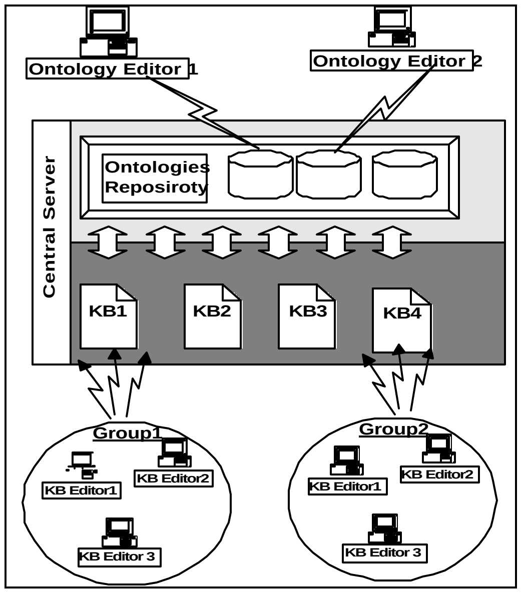


Figure 1: Architectural Overview of the Presented Environment.

4.1 Ontology & Knowledge Bases Server

The Ontology and Knowledge Bases Server (OKBS) is the core of the presented environment. It provides the development and deployment services to its clients over the network. The development side is a self-contained system that could be used independently like Ontolingua [7] or WebOnto [8]. The advantage of the environment is that the development side is integrated with a deployment side in order to enhance the process of constructing knowledge bases using the ontologies developed on the development side. From another perspective, the deployment of ontologies is the first phase in developing knowledge bases.

The OKBS comprises various subsystems that work in coordination with each other to provide the services offered by the OKBS. The following figure depicts those different subsystems and lays out the general scheme of interaction between them.

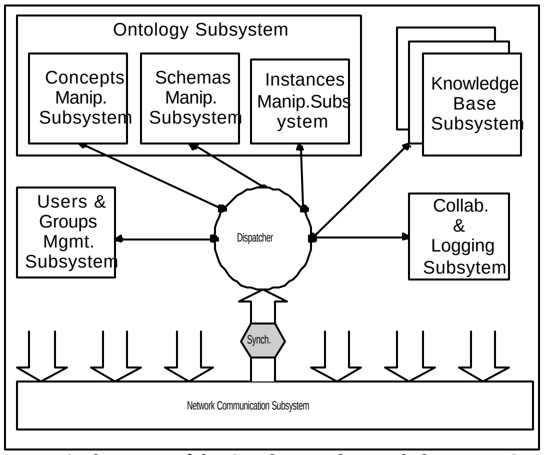

Ontology Subsystem
Concepts Manip. Subsystem
Schemas Manip. Subsystem
Instances Manip.Subsystem
Knowledge Base Subsystem
Users & Groups Mgmt. Subsystem
Dispatcher
Collab. & Logging Subsytem
Synch.
Network Communication Subsystem

Figure 2: Architecture of the Ontology and Knowledge Bases Server

The users and groups management subsystem is responsible for authenticating the clients of the OKBS and authorizing them to the appropriate resources. It manages its activities through a database of users and groups. In order to standardize the data persistence form in the OKBS, this database is also encoded in XML. **The network communication subsystem** is responsible for making the clients of the OKBS able to communicate with the server asking for its services. This subsystem depends on an XML-based protocol that is inspired by the notion of XML-RPC [21]. The design of the protocol is discussed separately in section 4.4. All the incoming requests from the client are forwarded to the dispatcher and any targeted subsystem could respond back via the network communication subsystem. This subsystem keeps a list of all clients who are working online and this list is used in broadcasting messages to all clients. **The dispatcher subsystem** is responsible for receiving incoming requests from the network communication subsystem and deciding which subsystem is the target of the request. This subsystem also guarantees that all the requests are serialized and do not interleave if necessary. Thus, all the clients requests to access shared resources is synchronized by this subsystem. **The ontology subsystem** is the core of the development side because it offers all the services for editing the ontology repository. Once the OKBS is launched, this subsystem loads the ontology repository from the persistent format and becomes ready for processing ontology editing requests. **The knowledge base subsystem** is responsible for all the operations that target the knowledge base. Multiple instances of that subsystem are created for each loaded knowledge base. This subsystem is the core of the deployment side and an instance of it is created on-demand when the first knowledge engineer requests creating/editing a particular knowledge base that is not already loaded. **The collaboration & logging subsystem** role is to make all the online users of the system aware of the other online peers of the OKBS and the transactions taking place at the OKBS. The subsystem also serves in transaction logging because it persists all the transaction information that the OKBS broadcasts to the online users in transaction logs. The transaction logs record also unbroadcasted information such as login failures and internal transaction errors.

4.2 Ontology Development Clients

In the development side, we find the first type of clients for the ontology server, namely, ontology editors. An ontology editor is a software component used by the ontology engineer. The ontology engineer is a knowledge engineer with a long experience in knowledge modeling since entities in the ontology are static facts that should be designed once and the designer should have in mind all the aspects needed to model the domain of discourse. The environment provides on this side three main software components: (1) The Concepts editor, (2) The Relation Schema Editor, (3) The Relation Instances Editor.

4.2.1 Concepts Editor

The Concepts Editor is a facility for the ontology engineer to edit the first component of the ontology, namely, the hierarchy of concepts. In some environments, where ontology is only a taxonomy of concepts, this is considered the main ontology editing facility. The Concept editor provides for the ontology engineers services such as: Loading the Concepts' hierarchy, Showing the details of a certain concept, Showing the facets of a certain property, Adding a concept, Deleting a concept, Adding a property, Deleting a property, Editing a property, etc..

4.2.2 The Relation Schema editor

The relation schema editor is one of the contributions of this research work and its goal is to give a facility for the ontology engineer by which he/she can define an abstraction of relations that are then instantiated later. This server does many transparent tasks to ensure integrity of and prevent redundancy in the ontology. The relation schema editor provides for the ontology engineers services such as: Adding relation schemas, deleting relation schemas, finding relation Schemas, etc. The idea of relation schema is not new, KADS methodology [3] has a similar definition for relation schema, but using it in the ontology development environment to enhance the redundancy checking of an ontology is a contribution of this research.

4.2.3 The Relation Instance editor

The relation schema editor is the extension of the relation schema editor, its goal is to give a facility for the ontology engineer by which he/she can instantiate relations schemas that were previously created. This editor does many transparent tasks to ensure integrity of and prevent redundancy in the ontology. The relation instance editor provides the ontology services such as: Adding relation instances, deleting relation instances, etc.. No instance can be created if it does not belong to a relation schema. This will increase the efficiency of checking the redundancy in the ontology

4.3 Ontology Deployment Clients

In the deployment side, we find the second type clients who are deploying the ontologies in building knowledge bases. The persons responsible for operating this side are the knowledge engineers. The facilities provided in that side could be described more by being "*picking*" facilities rather than "*editing*" facilities. This side of the environment avails for the knowledge engineer a concepts' picker, a relation schema picker and a knowledge-base snapshooter that he/she can use to build knowledge bases. Before the knowledge engineer uses the picking facilities, he should select a knowledge base to work with that was already created by the administrator and that he has the privilege to edit it. This picking facility is similar to SENSUS [22] with one exception. In SENSUS, the developer could upload the ontology after changing it. In our tool, it is only permitted to download the ontology to use it in the knowledge base but it is not allowed to upload it again. Any modification to the ontology must be done by the ontology administrator(s).

4.3.1 Concepts Picker

The concept picker provides the knowledge engineer with the facility of extracting a subset of concepts from the ontology to formulate the foundations of a new knowledge base. The hierarchy constructed with this concepts' picker is not the one that is going to be used in a final knowledge base because a knowledge base would typically include other non-ontological information. That is why the concepts' picker is used to formulate just the "foundations" or "the nucleus " of the knowledge base, i.e. the ontological aspects of the concepts' hierarchy.

4.3.2 Relation Schema Picker

The relation schema picker provides the knowledge engineer with the facility of extracting a subset of relation schemas from the ontology. The relation schemas selected are restricted to concepts that were previously picked in the knowledge base, i.e. a relation cannot be picked in a knowledge base if one of the concepts in its domain or range is not picked in that knowledge base. Once a schema is picked, all of its instances are picked with it as well.

4.3.3 Knowledge Base Snapshooter

The Knowledge Base Snapshooter is a facility for obtaining a snapshot of a knowledge base as a physical file. In the proposed environment, the knowledge base is stored in a primary knowledge base information file and then the knowledge base physical file is generated from it. The file that is obtained from the snapshooter could then be translated to other formats and that is basically rendered to that it is encoded in XML so, there are no parsing efforts in translation.

4.4 The XML-based Communication Protocol

The development and deployment environment is collaborative and distributed. Consequently, the design of a communication protocol is crucial to accomplish the goals of the system. Given the different software components in the system, certain requirements need to be satisfied.

4.4.1 Requirements of the Protocol

Power of Expression: The used protocol should be capable of handling complex data such as hierarchies of concepts and relation schemas. In addition, in many cases where knowledge is queried, a number of queries and their results need to be passed between the different software components. Moreover, other administrative messages need to be sent such as: login messages, chat messages, messages indicating the current logged on users, and project management messages for creating and manipulating knowledge base projects. Thus, the communication protocol is required to have enough expression capabilities to convey the various functional needs of data exchange between the server and the different clients.

Efficiency: *Efficiency of communication is another functional requirement of the protocol since many users are collaborating on ontology/knowledge base editing, thus, an acceptable response time should be provided for those collaborators. Collaboration could occur in a LAN-based environment but WAN access also should be supported, as is the case in most of the already-existing ontology servers.*

Openness: Ontology servers have the special property of the heterogeneity of their clients. This property necessitates the openness of the communication protocol. That is to say, no commitment to implementation languages, platforms or any other commitments that violate the heterogeneity of the collaborating clients should be enforced.

4.4.2 Design of the protocol

Given that the ontology server uses XML to encode different knowledge components, thus it is convenient to adopt a protocol that is based on XML. The idea of XML-based communication is not novel and is implemented in many systems. The protocol implemented in the presented work is inspired by the idea of XML-RPC [21]. One of the most mature implementation of that trend is the Simple Object Access Protocol [23].

The following summarizes the features of the protocol:

▪ All communication between the server and his clients is abstracted as remote calls to the other party's procedures.

▪ The calls are characterized by having variable-length number of arguments where the arguments are un-typed.

▪ For the communicating parties, each has a communication subsystem by which it can call the other party's procedure without being aware of the encoding details. A simple function call could be something like:

```
Login("Sameh","x123jk")
```

Where "Login" is the name of the procedure, "sameh" and "x123jk" are the arguments for the procedure.

▪ The communication subsystem encodes (or rather marshals) messages in an XML encoding that is ready to be sent to the other party. The following is a synopsis for an encoding of a simple procedure call:

```
<MSG>
      <CMD>Login</CMD>
       <ARG>Sameh</ARG>
      <ARG>x123jk</ARG>
 <MSG>
```

The <MSG> tag comprises the remote call and the first element in it is the name of the procedure to be called which is the content of the <CMD> tag, the following elements represent a list of arguments each comprised in an <ARG> tag.

The protocol is built on standard TCP/IP sockets in a three-tiered structure:

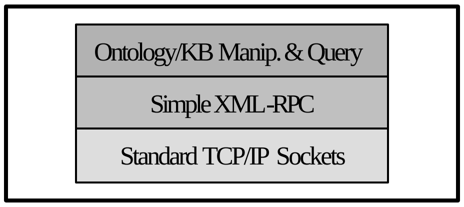


Figure 3: Three-Tiered structure of the comm. Protocol

Despite the simplicity of the protocol, it satisfies the functional requirements of expression, efficiency and openness.

▪ The fact that the arguments are un-typed and that they are only textual strings may seem to contradict with the power-of-expression requirement. To the contrary, this fact is actually the empowering feature of the protocol because the contents of the XML <ARG> tag could recursively contain XML-tagged structures. Thus, if there are any data structures that are already encoded in XML and need to be exchanged over a network, which is the case in the presented work, then there is no limit to the complexity of the arguments of the procedure call as long as they are XML-encoded.

▪ Concerning efficiency, the protocol adds a very small overhead, which is the tags for XML encoding.

▪ The Openness requirement is also satisfied because the protocol is text-based and could be implemented easily in any implementation language on any platform.

5 ONTOLOGY MODELLING

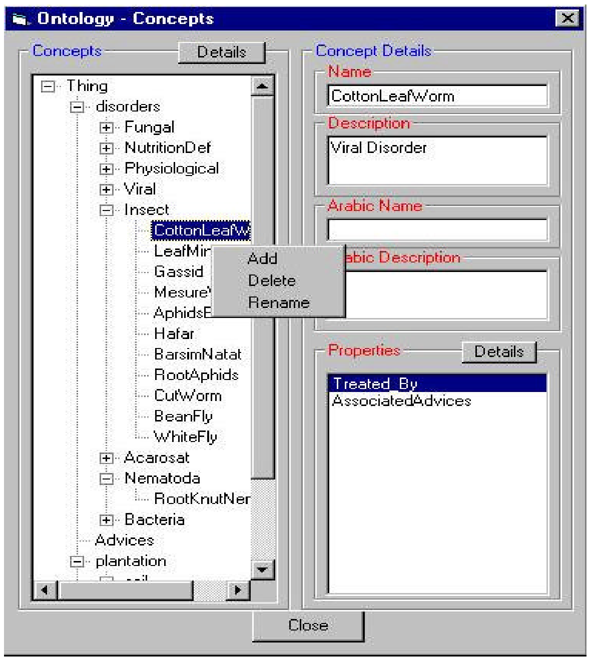
The presented research work adopts the broader sense of ontology of not just being a taxonomy of concepts but also it considers other entities. Thus, the main three components that are used in ontology are: The concepts' hierarchy, the relation schemas and the relation instances. The presented modeling for the concepts' hierarchy is similar to XOL [17], nevertheless relation/function encoding is not supported in XOL.

5.1 Concepts Hierarchy

This is a traditional hierarchy of concepts related mostly by the "IS-A" relationship. A concept represents a logical or physical entity in the domain of knowledge. The analysis of each domain could change the validity of whether a certain entity is a concept or not. However, as mentioned earlier, ontology comprises elements of knowledge that do not depend on personal opinions or perspectives. Thus, deciding whether a certain entity is a concept or not should be a decision taken by experienced ontology engineers in coordination with domain experts in order for a consensus to exist on the analysis of the domain to be represented in the hierarchy. Each **Concept** has the following elements: **Name**, **Description, Super-Concept, Property list.** Each property has some meta-attributes that describe it and they are called facets. The following is a list of selected facets for the properties of the concepts: **Name, Description**, **Type**, **Cardinality, Legal Values, Minimum** & **Maximum.** Figure 4 depicts a snapshot of the concept editor.

5.2 Relation Schemas

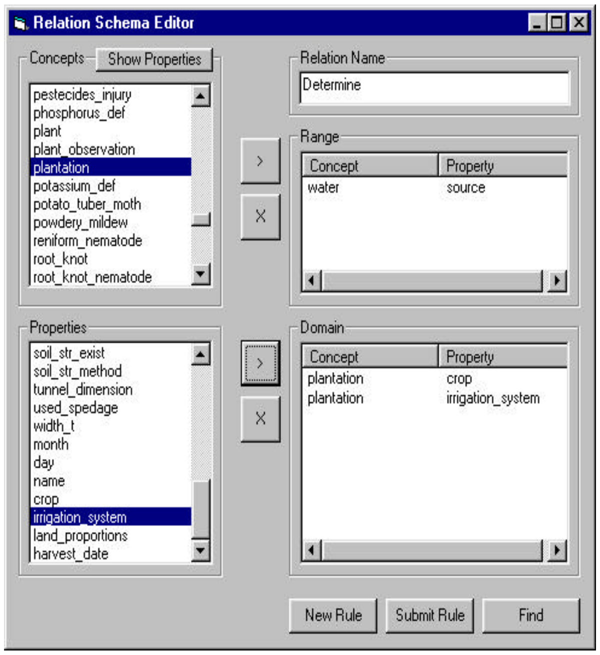
Normally relations are entered directly into ontologies without classification. In the proposed ontology model, relation schemas are provided first. They act as templates for creating relations. This feature aims at providing a way for checking redundancy of relations and ensuring more reusability and maintenance functionalities. A relation represents a type of interaction between concepts. It is analogous to a mathematical relation and the terms "Domain" and "Range" of a relation are borrowed. Thus, a relation schema as an ontology component is composed of three things: ***Name***, ***Domain*** & ***Range***, where ***Name*** denotes an identifier describing the connection between the concepts found in the ***Domain*** and those in the ***Range***.



Thing
disorders
Fungal
NutritionDef
Physiological
Viral
Insect
CottonLeafW
LeafMir
Gassid
Mesure
AphidsE
Hafar
BarsimNatat
RootAphids
CutWorm
BeanFly
WhiteFly
Acarosat
Nematoda
RootKnutNer
Bacteria
Advices
plantation

Name: CottonLeafWorm
Description: Viral Disorder
Properties: Treated_By, AssociatedAdvices

Figure 4: The Concepts Editor Main GUI



Concepts: pestecides_injury, phosphorus_def, plant, plant_observation, plantation, potassium_def, potato_tuber_moth, powdery_mildew, reniform_nematode, root_knot, root_knot_nematode

Properties: soil_str_exist, soil_str_method, tunnel_dimension, used_spedage, width_t, month, day, name, crop, irrigation_system, land_proportions, harvest_date

Relation Name: Determine

Range:

Concept	Property
water	source

Domain:

Concept	Property
plantation	crop
plantation	irrigation_system

Figure 5: Relation Schemas Editor

For example a relation in the agricultural domain could have the name **determine**, the domain is the **soil PH, and the water salinity**, and the range is the **plantation variety and plantation date**.

The following points summarize the features of a relation schema:

- The relation schema must have a unique identifier.
- The relation schema name is not unique and no relation can be created without having a name.
- Each entry in the domain and the range is of the form "Concept.Property" and no concepts can be found in an entry without specifying a property.
- Each relation schema must have at least one entry in the domain and at least one entry in the range.
- No two-relation schemas can share both the same domain and the same range. , e.g.:

  R1:C1.P1 --> C2.P2 & R2:C1.P1--> C2.P2 is illegal

Where C1 and C are concepts and P1 and P2 are properties in C1 and C2 respectively

- A "Concept.Property" entry cannot exist in both the domain and the range of the same relation unless "Property" is different, e.g.:

  R1:C1.P1xC2.P2 --> C1.P1 is illegal
  R1:C1.P1xC2.P2 --> C1.P2 is legal

- A "Concept.Property" entry can be repeated in the domain if "Property" is different among all entries, e.g.:

  R1:C1.P1xC1.P1 --> C3.P3 is illegal
  R1:C1.P1xC1.P2--> C3.P3 is legal

- A "Concept.Property" entry cannot be repeated in the range unless all properties of the range belong to the same "Concept", e.g.:

  R1:C1.P1--> C3.P1 x C3.P1 is illegal
  R1:C1.P1--> C3.P1 x C3.P2 is legal

Figure 5 depicts a snapshot of the relation schema editor.

5.3 RELATION INSTANCES

A relation instance is an element instantiated from a certain relation schema. The actual logic of the relation is encapsulated inside it. Thus, many instances could be sharing the same relation schema but the content of the relation itself is different. To illustrate this, consider the relation schema **soil PH and Water Salinity determines Plantation Variety and Plantation Date** it can have many instances as shown in table-1 :

Thus, the instance has the extra component of the value for each property in the relation schema. Notice also that there is also an operator between the property and its value and this operator is mostly a relational operator (=, <, >, !=,..) or a set operator (in, not in, ….), the relational operators could be combined together to specify ranges

Relation Name: Determine					
Domain			**Range**		
1st Instance					
Concept	Property	Value	Concept	Property	Value
Soil	PH	=5	Plantation	Variety	Orange
Water	Salinity	<20%	Plantation	Date	Spring
2nd Instance					
Soil	PH	>5	Plantation	Variety	Bean
Water	Salinity	=10%	Plantation	Date	Winter
…					
…					

Table 1: Instances of a Relation schema

As illustrated in Figure 6, we have the <INSTANCE> element that comprises the details of the instance. This element has one attribute, which is the SCHEMA_ID that refers to the relation schema that this instance is instantiated from. The details of the instance contain two types of elements: <DOMAIN> and <RANGE> elements and those correspond in order and number to the domain entries of the relation schema respectively. Each one of those elements has the OPERATOR attribute that specifies which relational or set operation is needed In addition, they comprise the <VALUE> element that includes the value of the property. This element is repeated in case the property has a cardinality of multiple.

```
<INSTANCE SCHEMA_ID=”3587” >
      <DOMAIN OPERATOR  = ”=” >
                  <VALUE>5 </VALUE>
      </DOMAIN>
      <DOMAIN OPERATOR  = ”<” >
                  <VALUE>0.02 </VALUE>
      </RANGE>
      <RANGE OPERATOR  = ”=” >
                  <VALUE>Orange </VALUE>
      </RANGE>
      <RANGE OPERATOR  = ”<” >
                  <VALUE>Spring</VALUE>
      </RANGE>
 </INSTANCE>
```

Figure 6: Example XML Encoding of a Relation Schema Instance

6 CONCLUSION

In the presented research work, we concentrated on improving the knowledge bases construction process via having a common environment where ontology is developed and then deployed to build knowledge bases. In this environment, we introduced a new knowledge encoding based on the eXtensible Markup Language (XML) that was used as the medium for knowledge persistence and exchange. The environment was designed to allow the collaboration of knowledge and ontology engineers in a distributed fashion.

The ontology development environment is similar in principle to previously developed environment but is simpler as only needed constructs in the agriculture domain are addressed. The idea of picking an ontological element to be included in the knowledge base is similar to SENSUS environment, but in our environment uploading a modified ontology on the client side is not permitted. In SENSUS [22], it is permitted to upload a modified environment to the central ontology after being checked for consistency. It should be mentioned in this regard that our main intention was to build a common ontology for knowledge engineers such that they can use it when building a new knowledge based system or modify an existing one by picking the ontology elements they want. Once they get the required ontology elements they can add more knowledge related to their application. Therefore, using developed ontology tools available on the Web was not sufficient to satisfy our goal.

At this stage of research we could not guarantee a100% synchronization between the ontology contents and the knowledge base and this was not our intention. There is some work here for future research to synchronize the developed knowledge base with the ontology. For example, if changes are made to the ontology and will affect the knowledge base, message should be sent automatically to the registered knowledge base and maintenance tool must be provided on the client side to take necessary action. It should also be mentioned that if the developed knowledge bases are to be shared and reused a knowledge base server is to be built. This server must include all the developed knowledge bases in a consistent and integral manner. This is another future research that may use the same approach used in this paper but to include the knowledge base primitives as well and to permit the knowledge base developer to upload their knowledge bases after being checked for consistency.

The work presented here is under evaluation in the Central Laboratory for Agricultural Expert Systems. An ontology of the cucumber expert system [4] was uploaded and edited to be considered the unified ontology. Administrator of this ontology was assigned. The new expert systems that will be built will use this unified ontology. In the mean time the different functionality of the developed environment was tested especially the translators from XML to KROL [24] and from KROL to XML and work satisfactorily [5] . The plan of evaluation is to measure the time saving in building new expert system by comparing the time the new expert systems takes and the old ones. Secondly, we will measure the integration efforts taken to integrate subsystems of an expert system of one crop developed using the unified ontology with the integration effort we are doing now. The comments of the developers will also be considered for enhancing the environment.

7 REFERENCES

[1] Swartout, W. T., Austin (1999). “Ontologies.” IEEE Intelligent Systems & their application 14(1): 18-19

[2] Gruber, T. R. (1993). “A Translation Approach to Portable Ontology Specifications.” Knowledge Acquisition(5): 199-220

[3] Schreiber, A. T. W., B. J.; Hoog, R. de; Akkermans, J. M.; Velde, W. Van de (1994). “A principled approach to knowledge-based system development.” IEEE Expert 9(6): 28-37

[4] Rafea, A. A. E-A., Sayed; Ibrahim, Iman; Edres, Soliman; Mahmoud, Mostafa (1995). Experience with the Development and Deployment of Expert Systems in Agriculture. IAAI, Canada

[5] El-Ansary, Sameh (2000). “Distibuted development and deployment of ontologies for knowledge-based systems” M.Sc. Thesis, The American University in Cairo, Cairo, Egypt

[6] Gomez-Perez, A. B., V. (1999). Overview of Knowledge Sharing and Resue Components: Ontologies and Problem-Solving Methods. IJCAI99' workshop on Ontologies and Problem-Solving Methods, Stockholm

[7] Fraquhar, A. F., R.; Rice, J (1997). “The Ontolingua Server:Tool for Collaborative Ontology Construction.” IJHCS 46(6): 707-728

[8] Domingue, J. (1998). Tadzebao and WebOnto: Discussing, Browsing, and Editing Ontologies On theWeb. Eleventh Workshop on Knowledge Acquisition, Modeling and Management, KAW'98, Banff, Canada

[9] Eriksson, H. F., R.; Shahar, Y; Musen, M.A. (1995). “Task modeling with reusable problem-solving methods.” Artificial Intelligence 79: 293-326

[10] Fernandez, M. G.-P., A.; Pazos, J.; Pazos, A. (1999). “Building a Chemical Ontology Using Methontology and the Ontology Design Environment,.” IEEE Intelligent Systems & their application 14(1): 37-46

[11] Gruber, T. R. (1995). “Toward Pricnciples for the Design of Ontologies Used for Knowledge Sharing.” International Journal of Human-Computer Studies(43): 907-928

[12] Genesereth, M., Fikes, R (1992). Knowledge Interchange Format. Technical Report. Computer Science Department. Stanford University. Logic-92-1

[13] Perez, A. G. B., V. Richard (1999). Overview on knowledge sharing and reusecomponents: Ontologies and Problem-Solving methods. IJCAI-99 workshop on Ontologies and Problem-Solving Methods, Sweden

[14] MacGregor, R. Inside (1991). the LOOM clasifier. SIGART bulletin. #2(3):70-76..

[15] Cycorp (1999). Features of CycL (http://www.cyc.com).

[16] Kifer, M., Lausen, G., Wu, J.(1995) Logical Foundations of Object-Oriented and Frame-Based Languages. Journal of the ACM.

[17] Karp, P. D. C., Vinay K.; Thomere, Jerome (1993). An XML-Based Ontology Exchange Language. Menlo Park, Pangea Systems

[18] Ontologos (1999). The Information Flow Foundation for Conceptual Knowledge Organization. Pullman, The Ontology Consortium (http://www.ontologos.org).

[19] Staab, S. M., A. (2000). Ontology engineering beyond the modeling of concepts and relations. ECAI’2000 Workshop on Application of Ontologies and Problem-Solving Methods, Amesterdam, IOS Press

[20] Fensel, D. C., M.;Van Harmelen,F.;Horrocks I. (2000). OIL & UPML: A Unifying Framework for the Knowledge Web. ECAI’2000 Workshop on Application of Ontologies and Problem-Solving Methods, Berlin, IOS Press

[21] XML-RPC (1999). (http://www.xm-rpc.com)

[22] Swartout, B.; Ramesh P.; Knight, K.; Russ, T.(1997) "Toward distributed use of large-scale ontologies". Symposium on Ontological Engineering. American Association for Artificial Intelligence (AAAI). Stanford (California).Marzo

[23]MICROSOFT(2000).Simple Object Access Protocol (SOAP) (http://msdn.microsoft.com/workshop/soap).

[24] Shaalan, K., Rafea, M, &.Rafea, A.(1998), "KROL: A Knowledge Representation Object Language on Top of Prolog, Expert Systems with Applications", An International Journal, Vol. 15, pp. 33-46, Elsevier Science Ltd,