Available online at www.isas.org.in/jisas
JOURNAL OF THE INDIAN SOCIETY OF
AGRICULTURAL STATISTICS 67(1) 2013 107-120

Diseases and Pests Identification in Maize – A Multilingual Scenario

Sudeep Marwaha[1], Punam Bedi[2] and V.K. Yadav[3]
[1]*Indian Agricultural Statistics Research Institute, New Delhi*
[2]*University of Delhi, Delhi*
[3]*Directorate of Maize Research, New Delhi*

Received 13 August 2012; Revised 05 January 2013; Accepted 15 January 2013

SUMMARY

Pests and diseases cause major economic losses to the farmers. The estimate loss per annum due to diseases and pests in a country like India touches to billions of Rupees. Most of the time farmers use over dosages of pesticides and fungicides to save their crop and thus cause environmental hazards. The presented expert system is designed to help farmers to identify diseases and insects attacking maize crop which is neither feasible nor practical by conventional system of extension. Traditional expert systems are based on rules and facts whereas the knowledgebase of this expert system is built using ontology – the latest knowledge representation technique. OWL is the W3C specifications for building ontologies. It is based on XML and Unicode. Moreover, rule-base knowledgebase is not inherently based on Unicode and thus lacks support for internationalization or for regional languages. The system acts as a tool for transferring the site and crop specific knowledge of various domain experts to the farmers. The system is integrated with the Maize Agridaksh. Agridaksh is a tool for developing online expert system of crops. India being a multilingual society with over 16 major languages and most of the farmers across the country understands their local language only. This system is multilingual and at present contains knowledge in English and Hindi languages.

Keywords: Ontology, OWL, Diseases, Pests identification, Maize, Protégé, Semantic web, SPARQL.

1. INTRODUCTION

The Internet was opened to general users in 1994 and this new era of information and communication technology has played an important role in the field of expert systems. The Web technologies allowed the knowledge engineers and domain experts to build the expert systems that were having dynamic knowledgebase capabilities (Marwaha *et al.* 2002). The domain experts could update the knowledge at the central server and the users had an access to the recent knowledgebase through a Web interface. But the increase in use of computers, Internet, number of people that have got access to computing resources and number of applications in almost every field of life, has created another problem. Domain experts have the pressure to create or update the knowledgebase in number of experts systems or applications that use knowledge in some sense. These systems use some common knowledge about the domain or mix knowledge from different domains to create their specific knowledgebase. It becomes harder for the domain experts from different fields to sit together for creating and updating the knowledgebase for each of these systems. So, there is a need to shift the knowledgebase to a knowledge representation technique that allows domain experts to code their knowledge once that can be reused in many systems. Ontology is the latest knowledge representation technique that allows the domain experts to code their knowledge in a specific

Corresponding author : Sudeep Marwaha
E-mail address : sudeep@iasri.res.in

domain. Ontology is devised for the Web based systems and provides them with the semantics of the concepts in the specific knowledge domain. They have the potential to be used in a distributed environment like Internet and provide the dynamic and reusable capability to the knowledgebase.

Pests and diseases are major causes for the damages in the crops and results in the economic loss to the farmer. The estimate loss per annum due to disease and pests in a country like India touches to billions of Rupees. Most of the time farmers use over dosages of pesticides and fungicides to save their crop and thus cause environmental hazards.

In this paper a Semantic Web approach is presented to design and develop ontology based multilingual expert system. The system is designed to help farmers to take appropriate decisions and disseminate need based research findings to millions of the farmers at a time, which is neither feasible nor practical by conventional system of extension. In conventional architecture of expert system, the knowledge base is built by some knowledge engineers along with domain experts and these expert systems are known to be working within a narrow domain of knowledge. But for building the expert system in agriculture for a vast and diverse country like India, the conventional approaches fail to meet their objectives. The systems that can meet this requirement must have dynamic and portable knowledgebase. Moreover in today's dynamic world the solution loses its relevance if it is not based on the recent knowledge. So, there is need of the regular updating of knowledgebase by the team of experts from their desktops using Web technologies. Semantic Web with its enriched set of tools promises for such a solution. With technologies and standards such as XML, RDF/RDFS, OWL, Ontologies, Protégé and Agents in place, creation of such an expert system is possible.

1.1 Review of Literature

Initially, expert systems were built from scratch and there is no abstraction in terms of knowledgebase and inference engine. As the research in the field matured researchers have developed generic inference engines that can be applied to any knowledge domains.

Gruber (1993) defined ontology as "an explicit specification of a conceptualization" and describes objective criteria to guide and evaluate designs that are founded on the purpose of the resulting artefact, rather than based on a priori notions of naturalness or truth. He proposed a preliminary set of design criteria for ontologies whose purpose is knowledge sharing and interoperation among programs based on a shared conceptualization.

The Semantic Web envisioned by Berners-Lee (Berners-Lee *et al*. 2001), provides automated information access based on machine-processable semantics of data and heuristics that use these metadata. The explicit representation of the semantics of data, accompanied with domain theories (that is, ontologies), enables a Web that provides a qualitatively new level of service. It has the capability to weave together an incredibly large network of human knowledge complemented with machine processability.

Brickley and Guha (1999) proposed Resource Description Framework (Schema) as a W3C standard. The proposal later became the W3C recommendation in 2000. The motivation for the RDF standard was primarily the description of metadata about Web-based resources. RDF has XML based syntax and it allows making statements about things or resources in the form of triples. Berners-Lee *et al*. (2001) presented the Semantic Web Architecture; a revision of the architecture was also proposed (Kifer *et al*. 2005) to realize Semantic Web's full potential growth. Dean (2003) described the Web Ontology Language (OWL) and its design goals. Horrocks *et al*. (2003) discussed the formalism of OWL, rooted in Description Logic. OWL, designed for different uses on the Web, has universal expressive power, with support for syntactic and semantic interoperations. OWL described the structure of a domain in terms of classes, properties and restrictions. OWL has three increasingly-expressive sublanguages: OWL Lite, OWL DL and OWL Full designed for use by specific communities of implementers and users. OWL Lite supports those users primarily needing a classification hierarchy and simple constraints. It should be simpler to provide tool support for OWL Lite than its more expressive relatives, and OWL Lite provides a quick migration path for thesauri and other taxonomies. OWL Lite also has a lower formal complexity than OWL DL. OWL DL supports those users who want the maximum expressiveness while retaining computational completeness (all conclusions are guaranteed to be computable) and decidability (all computations will finish in finite time). OWL DL includes all OWL language constructs, but they can be used only under certain restrictions (for example, while a class may be a subclass of many classes, a class cannot be an instance of another class). OWL Full is meant for users who want maximum expressiveness and the syntactic freedom of RDF with no computational guarantees. Protégé (Gennari *et al*. 2003, Golbeck *et al*. 2003) is an integrated software tool used by system developers and domain experts to develop knowledge-based systems. Protégé can be characterized as an ontology development environment. It provides functionality for editing classes, properties, and individuals. At its core is a frame-based knowledge model (Noy *et al*. 2000, Noy *et al*. 2001) with support for metaclasses. These metaclasses can be extended to define other languages on top of the core frame model. The OWL Plug-in (Knublauch *et al.* 2004) is an extension of Protégé. The OWL Plugin can be used to edit OWL ontologies, to access description logic (DL) reasoners, and to acquire instances for semantic markup. As an extension of Protégé, the OWL Plugin profits from the benefits of a large user community, a library of reusable components, and a flexible architecture. The OWL Plugin therefore has the potential to become a standard infrastructure for building ontology-based Semantic Web applications.

The applications of the semantic web in domains like agriculture are much needed as this domain has a very diverse knowledge that depends upon the geographic locations. A knowledge base solution that works very efficiently in one geographic location may be the worse solution if applied in another location. The solutions and the knowledge bases in the agriculture domain are believed to be much more complex than the business domain. Also, this domain is neglected one as compared to other domains where much of the work is already done. We believe that the presented framework will boost the thinking process as well as building of location specific ontologies and knowledge base systems in the agriculture domain (Bedi *et al.* 2005).

Section 2 discusses the designing of crop ontology. Section 3 describes the development of crop ontology. Section 4 gives the architecture of the presented ontology based expert system. Section 5 and 6 present the implementation of expert system for Disease Diagnosis and Insect identification of maize crop respectively. Section 7 concludes the paper with merits of the system.

2. DESIGNING ONTOLOGY FOR DISEASE DIAGNOSIS AND IDENTIFICATION OF INSECTS FOR MAIZE

Ontology creation from the scratch is a complex process and is expensive, as many iterations are required to relate each and every concept with one another. Ontology engineering requires in depth knowledge of the domain as well as expertise in building knowledge representation and organization techniques, a rare combination that can be found for any domain. However, due to the standardization of Ontology Web Language and tools such as Protégé 3.3.1/Web Protégé, building ontologies have become significantly easier. But for domain experts, especially in the fields that are not much concerned with the computers and information technology, it is still harder to use these tools and standards. On the other hand, for the knowledge engineers, the problem of getting knowledge from the domain expert remains unsolved. In this section, efforts for devising an approach to create ontology for disease diagnosis for maize (Bedi *et al*. 2004) are presented. This requires some knowledge source in the requisite domain and two booklets (Leon 1996) and (Ortega 1987) are the right choice. Protégé/Web Protégé is used to demonstrate the implementation of generating OWL Ontology for Maize. This makes it easy to integrate with other ontologies or agent based systems that make Semantic Web a realization. List of steps for designing the ontology are as follows:

1. Study the two booklets and list the major entities i.e. the maize diseases and insect-pests of maize. Put them in the hierarchical format.
2. These entities will be the classes in the ontology under owl: Thing and under diseases sub-classes are created.
3. Create different classes under owl: Thing class in a hierarchy corresponding to the maize diseases and insect-pests of maize as they appear in the taxonomy hierarchy.
4. List different properties that are applicable to the instances or individuals of the classes defined in step 2 and 3. These properties are assigned appropriate values at the time of creation of

individuals. By default, these properties are inherited to the individuals of the classes that are derived from them.

5. Decide the range and data type of values for each property of the class. In OWL, one can have two types of properties Data type Property and Object property. The Data type property can have data types defined by XML Schema such as Boolean, Float, Integer, String, and Symbol values while Object property can point to other class or individual of the class.

6. Repeat Step 4 and Step 5 for all the classes and sub-classes.

7. Create individuals for all the classes and sub-classes of the ontology and assign values to the different properties for the individuals of different classes of the ontology.

8. Refine and enhance the ontology by adding appropriate restrictions and other annotation properties.

3. BUILDING ONTOLOGIES FOR MAIZE DISEASES AND INSECT-PESTS IN PROTÉGÉ

Ontologies have come to be seen as a critical component of the Semantic Web. Ontology provides a common vocabulary to support the sharing and reuse of knowledge. When two parties agree to use the same ontology, they agree on the meanings for all terms from that ontology and their information can be combined easily. Protégé is a free, open-source platform that provides a growing user community with a suite of tools to construct domain models and knowledge-based applications with ontologies. At its core, Protégé implements a rich set of knowledge-modeling structures and actions that support the creation, visualization, and manipulation of ontologies in various representation formats. Protégé can be customized to provide domain-friendly support for creating knowledge models and entering data. This section describes the development of Crop ontology that acts as a knowledgebase for the presented expert system and is used for querying and reasoning on user's request. By using the above approach, the Ontologies for Maize Diseases and Insect-Pests in Protégé with OWL plug-in is implemented. However, the approach is generic and can be implemented with any tool that provides support to implement ontology in OWL.

3.1 Creating Classes

First step is to identify different classes that are required to define Ontologies for Maize Diseases and Insect-Pests.

- Since, crop_ontology is at the top of the hierarchy, the crop_ontology class is created as the base class under owl:thing class from which other classes are derived. Crops class is the first sub class, cereals class is the next sub class.
- Similar to Crops class other sibling classes like Crop_Stage, PartsAffected, Diseases and Symptoms are created in the hierarchy. For Diseases class four sub-classes are created like Bacterial_disease, Fungal_disease, Nematode_disease and Virus_disease (Fig. 1).

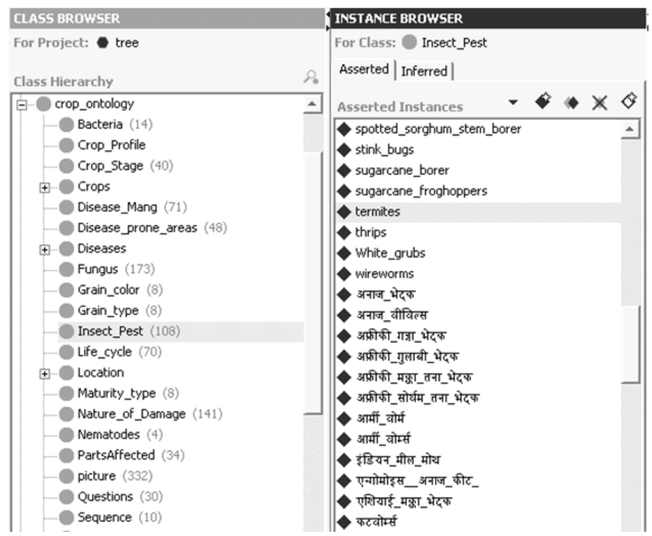

Fig. 1: Class Hierarchy and Individuals of Insect_Pest class in crop_ontology

- For Insect-Pests three new class Insect_Pest, Nature_of_Damage and Life_cycle under crop_ontology are created.
- Also, the rdfs:subClassOf relationship is established as a necessary condition between the parent classes and corresponding sub classes.

3.2 Creating Individuals

After creating all classes individuals of all classes of crop_ontology were created through Individual Editor of Protégé. *e.g.* Under crop_ontology class the subclass Insect_Pest has over 100 individuals like

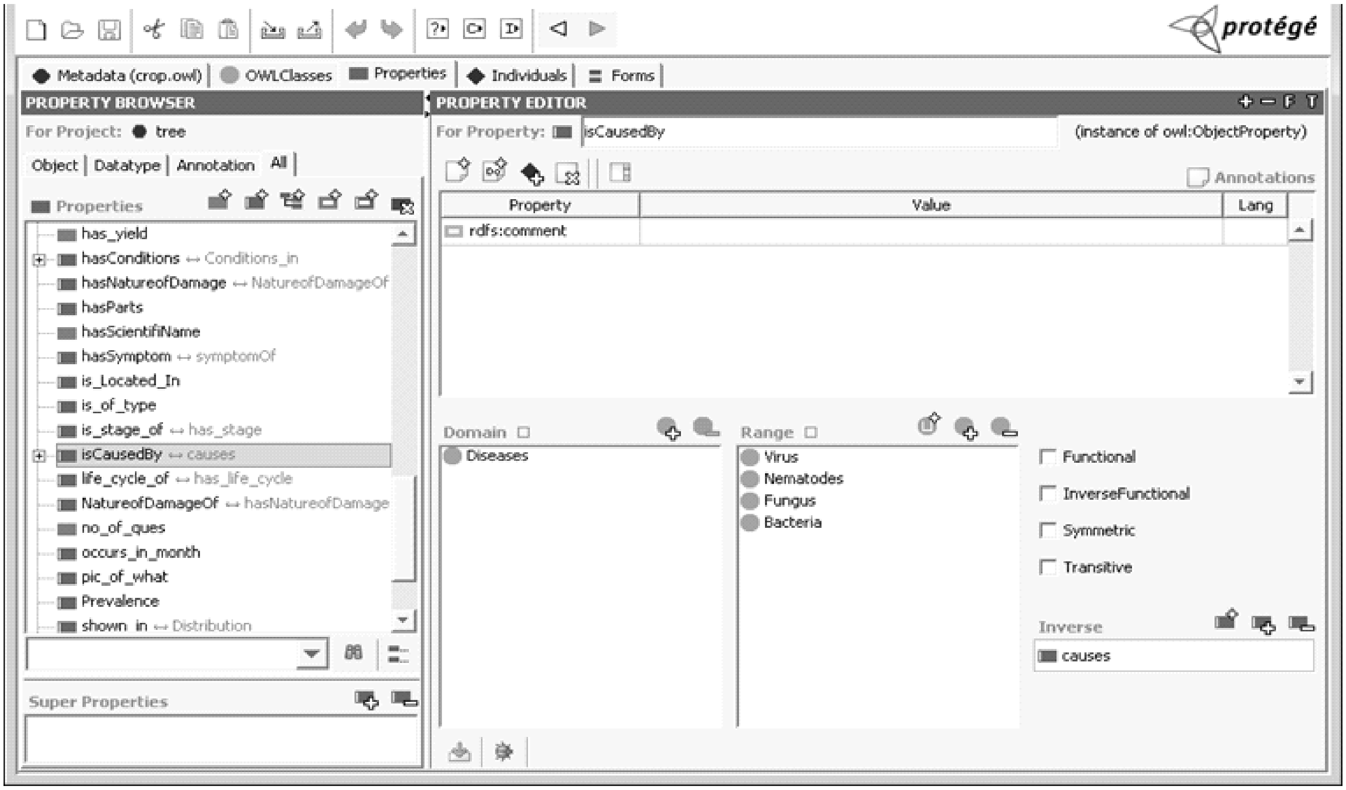

Fig. 2: List of properties in crop_ontology

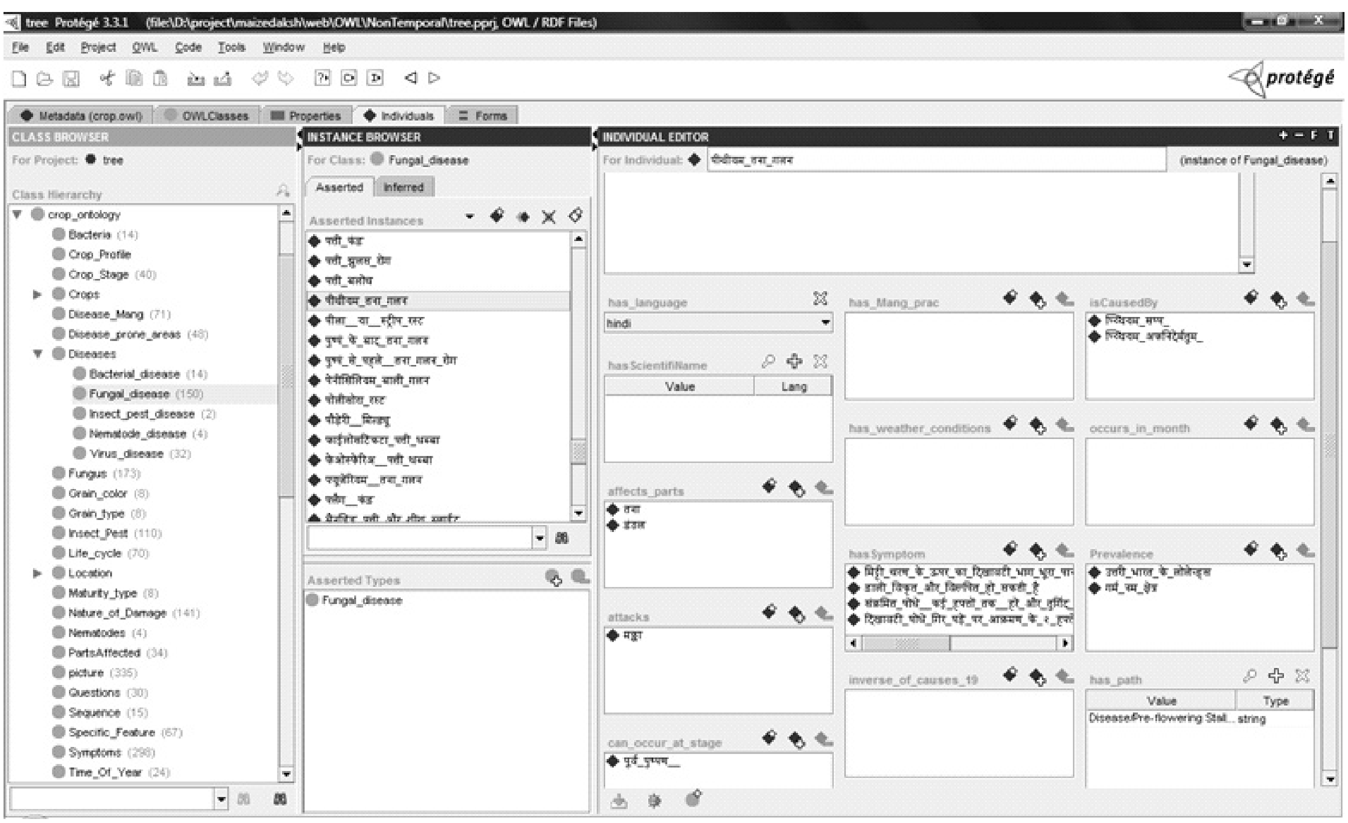

Fig. 3: Fungal disease class with its individual and properties in crop_ontology

african_maize_stem_borer, african_pink_borer, african_sugarcane_borer, termites etc. Similarly individuals of other classes of crop_ontology are created. As the system supports multilingual the following snapshot shows class Insect_Pest and its individuals both in English and Hindi (Fig. 1).

3.3 Creating Properties

After creating classes and individuals properties for classes and their individuals in crop_ontology are created. Properties were created through Property Editor of Protégé, where *domain* and *range* for each property were declared *e.g. hasSymptom* is a object property it's domains is *Diseases* class and range is *Symptoms*. Similarly has_language is a datatype property whose domain is crop_ontology and range is string with allowed values English and Hindi (Fig. 2 and Fig. 3). Other properties were created and their domain and range were also specified.

3.4 Creating Restricted Property

In addition to designating property characteristics, it is possible to further constrain the range of a property in specific contexts in variety of ways, called *property restrictions*. All restrictions are made under a tag *owl:Restriction*. The *owl:onProperty* element indicates the restricted property. Similarly, all required restrictions of other property classes are applied.

4. ARCHITECTURE OF ONTOLOGY BASED EXPERT SYSTEM

The presented expert system is based on the n-tier model of the web applications. This model allows different components of the system to be built by different experts, specialized in their domain. Fig. 4 shows how the components of the system interact with each other. Each of these components can exist on the different machines or anywhere on the web. Knowledge base and inference engine are the two most important components of an expert system. The basic principal of the separation of the knowledge from its treatment is of prime importance in the building of every expert system. The building and maintenance of an expert system is greatly facilitated by trying to adhere to this principal as closely as possible.

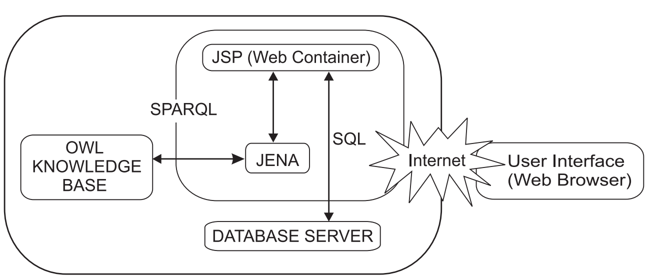

Fig. 4: The N-tier architecture of the software

- **The Knowledge Base Layer (KBL):** The knowledgebase is built using OWL ontology. It contains knowledge about Maize varieties and diseases and insect-pests.
- **The Database Layer (DBL):** This layer is implemented using MS SQL Server 2008 database. This contains the authorisation information about users and crop specific information.
- **The Reasoning Engine:** The reasoning engine accepts user input queries and responses to questions through the I/O interface and uses this dynamic information together with the static knowledge stored in the knowledge base. The knowledge in the knowledge base is used to derive conclusions about the current case or situation as presented by the user's input. JENA is used here for this purpose.
- **Server Side Application Layer (SSAL):** Application layer is built using Java Server Pages (JSP). The JSP provides the web developers with a framework to create dynamic content on the server using HTML, XML, Java classes, which is secure, fast and independent of server platform.
- **Client Side Interface Layer (CSIL):** It will be implemented using Hyper Text Markup language (HTML), CSS and JavaScript. The CSIL consists of forms for accepting information from the user and validation those forms using JavaScript. It also provides the explanatory interface to the users of expert system.

5. DISEASE DIAGNOSIS IN MAIZE

The software "Ontologies based Expert System for Maize" is integrated with Maize AgriDaksh. This section explains about the class hierarchy, properties and relations between classes and properties about Maize Diseases.

Bacteria: This is a class for bacterium that affects crops and causes diseases. The class Bacteria is a subclass of owl: Thing and its individual have three restrictions namely, they have exactly one language, they affect at

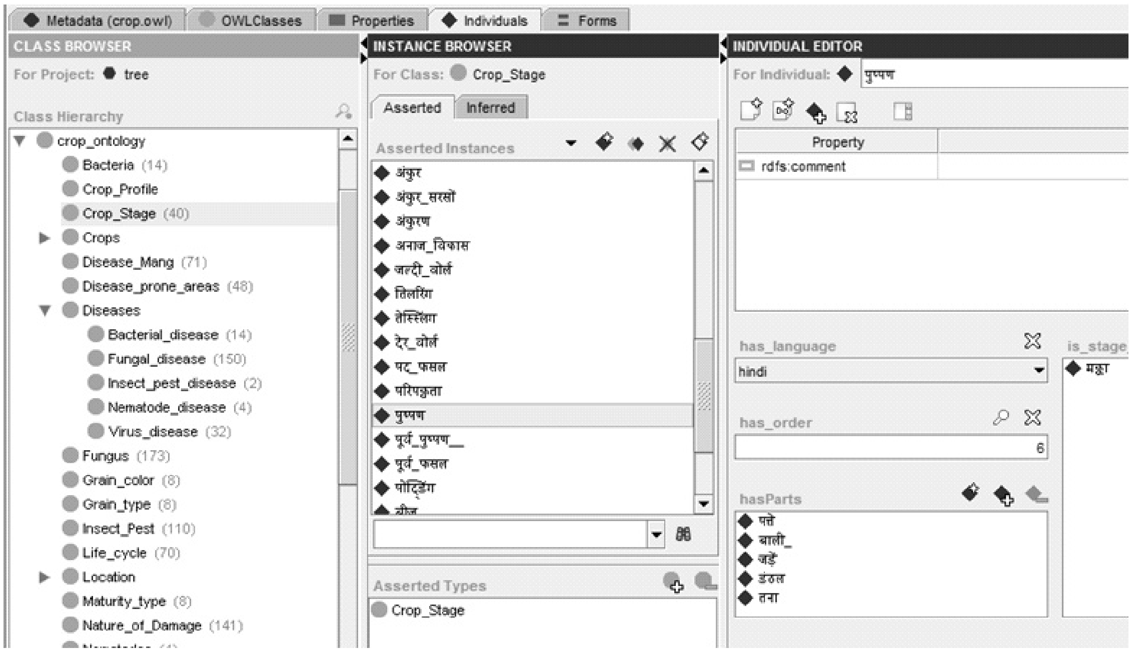

Fig. 5: Class Crop_Stage and its individuals with an example named Flowering

least one crop and cause at least one disease. The following snapshot shows the class view for bacteria and individual view for bacteria named *Erwinia carotovora p.v. zeae*.

Crop_Stage: Crop_Stage is a subclass of owl:Thing. It is defined to store growth stages for different crops as different crops have different crop stages. Crop_stage is also domain for datatype property is_stage_of where range has allowed values from class Crops. It has one object property namely, hasParts which shows different parts of maize plant that may be affected by diseases belong to a particular stage of a crop and one datatype property namely, has_order to show the stage of the crop in such a manner that appears in plant as it grows. The following snapshot shows class Crop_Satge and its individual view of a stage called Flowering [Fig. 5].

Crops: It is a class created to store different crops and it is a subclass of owl: Thing. It has different sub-classes like cereals, fruits, Oilseeds, vegetables etc. Under class cereals there are again sub-classes for rice, wheat and maize. Class maize has its individuals which have two bject properties has_Stage and is_Located_in.

Diseases: It is a class for diseases that affects crops adversely. Its sub-classes are bacterial_diseases, fungal_diseases, virus_diseases, and nematode_ diseases. Individuals of this class have two restrictions namely, they have at least one symptom and caused by at least one of bacteria, fungus, nematodes or virus. It also has many properties like attacks and affects_parts whose domain are Crops and PartsAffected respectively. Besides this, it has other properties namely, has_Mang_prac, can_occur_at_stage etc.

Fungus: It is a Class for fungus that causes fungal disease. Individuals of this class have two restrictions namely, they affects at least one crop and causes at least one disease.

PartsAffected: PartsAffected is a sub-class of owl: Thing. This class contains instances which represent different parts of crops which are most commonly affected by bacteria, fungus etc. Parts to be added are created as individuals of this class. Following snapshot shows a part leaves being added to the knowledgebase.

Virus: Class for viruses that effect crops. Individuals of this class have two restrictions namely, they affects at least one crop and causes at least one disease.

Type: It is a sub-class of owl: Thing and it is for indication of type of symptoms whether it is initial or final.

Symptoms: Class for common symptoms for crops. Individuals of this class have restriction that they are symptom of at least one disease. The individuals of this

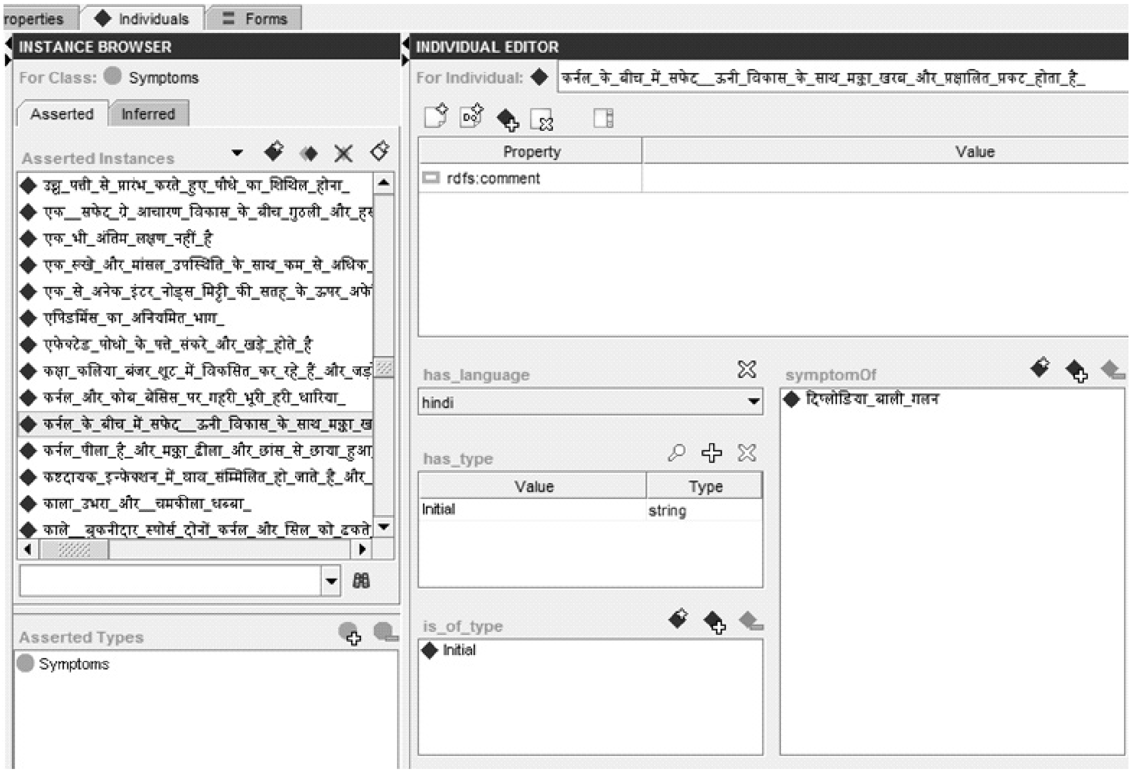

Fig. 6: Class Symptoms and its individuals

class are involved in property symptomOf which related disease with its symptoms. It has a object property is_of_type which is used to indicate whether a symptom initial or final type [Fig. 6].

When the project is run the home page of Maize AgriDaksh appears [Fig. 7]. The presented work is integrated with this project through the Problem Identification option that appears on the left frame. Fig. 8 shows the page that downloads when the Problem Identification link is clicked.

After selecting Disease Diagnosis, a series of questions are asked and the user selects the appropriate answer for each one of them. For each question, the answer options are supported with relevant photos. After every question, a SPARQL query is dynamically generated and executed. The execution of query results

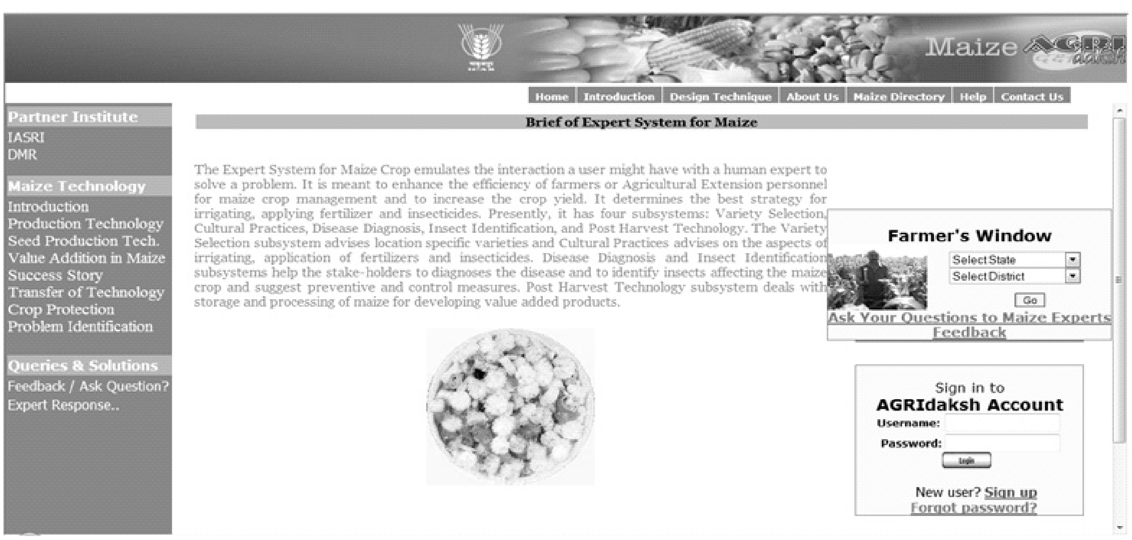

Fig. 7: Home page of Maize AgriDaksh

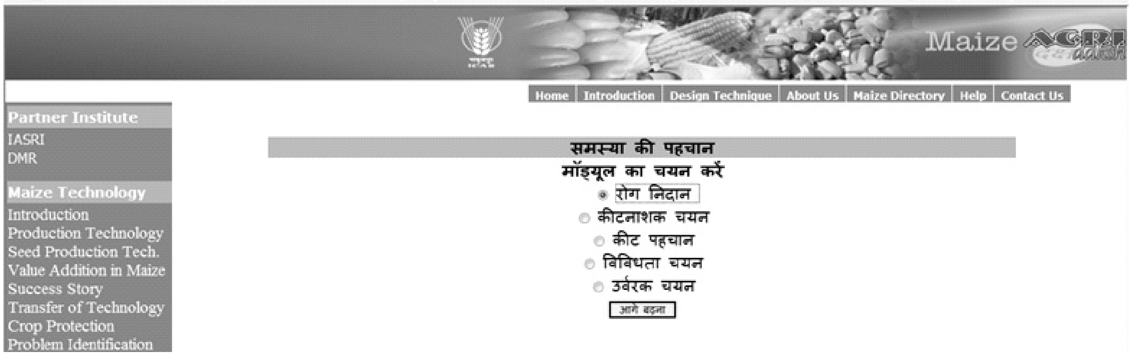

Fig. 8: Problem Identification Module

Table 1. SPARQL queries generated for disease diagnosis module

No.	SPARQL Query	No.	SPARQL Query
1	SELECT?tree WHERE {?tree : :has_sequence ? sequence. ?tree :has_language “hindi”}	2	SELECT ?select_part ? query_clause ?ques WHERE {:रोग_अनुक्रम :has_ques1 ?qid. ?ques : has_id ?qid. ?ques:has_select_clause? select_part. ?ques :has_query_clause ?query_clause. ?ques :has_language “hindi” }
3	SELECT distinct ? Crop ?path WHERE {?probable_pest : attacks ?Crop. ?Crop : has_language “hindi”. ?pic :pic_of_what ?Crop. ?pic :has_path ?path.	4	SELECT distinct ?Crop_Stage? path. WHERE { ?probable_pest :attacks ? : मक्का :has_stage ?Crop_Stage. ?Crop_Stage :has_language “hindi”. ?pic :pic_of_what ?Crop_Stage. ?pic :has_path ?path} ?Crop_Stage:has_order ?A.} order by (?A)
5	SELECT distinct ?Parts_Affected ?path w { ?probable_pest :attacks : मक्का . मक्का : has_stage : पुष्पण . पुष्पण : hasParts?Parts_Affected. ?Parts_Affected : has_language “hindi”. ?pic :pic_of_what ?Parts_Affected. ?pic :has_path ?path}	6	SELECT distinct ?Initial_Symptom?path WHERE { ?probable_pest :attacks : मक्का . :मक्का :has_stage : पुष्पण . ?probable_pest :affects_parts : पत्ते . ?Initial_Symptom :is_of_type :Initial. ?probable_pest :hasSymptom ?Initial_Symptom. ?Initial_Symptom :has_language “hindi”. ?pic :pic_of_what ?Initial_Symptom. ?pic :has_path ?path}
7	SELECT distinct ?Final_Symptom ?path WHERE { ?probable_pest :attacks : मक्का मक्का : has_stage : पुष्पण . ?probable_pest :affects_parts: पत्ते ?Initial_Symptom :is_of_type :Initial. ?probable_pest :hasSymptom:(नसों_के_साथ_छोटे_हरे_घाव_कम_उम्र_की_पति यों_पर_मुख्य_रूप_से_एक_विशिष्ट_स्ट्रिपिंग_उत्पादित_करते_हैं ?Final_Symptom :is_of_type :Final. ?probable_pest :has Symptom ?Final_Symptom. ?Final_Symptom :has_language “hindi”. ?pic :pic_of_what ?Final_Symptom. ?pic :has_path ?path}	8	SELECT distinct ?probable_pest?path WHERE { ?probable_pest :attacks : मक्का मक्का : has_stage : पुष्पण . ?probable_pest :affects_parts: पत्ते ?Initial_Symptom :is_of_type :Initial. ?probable_pest :hasSymptom:(नसों_के_साथ_छोटे_हरे_घाव_कम_उम्र_की_पति यों_पर_मुख्य_रूप_से_एक_विशिष्ट_स्ट्रिपिंग_उत्पादित_करते_हैं ?Final_Symptom :is_of_type :Final. ?probable_pest :has Symptom: गंभीर_क्षति_के_तहत_शीर्ष_पत्ते_टेसल_रोटिंग_में_परिव र्तित हो जाते हैं क्योंकि टेसल मृत पत्तियों से घिरा है ?probable_pest :has_language “hindi”. ?pic :pic_of_what ?probable_pest. ?pic :has_path ?path}

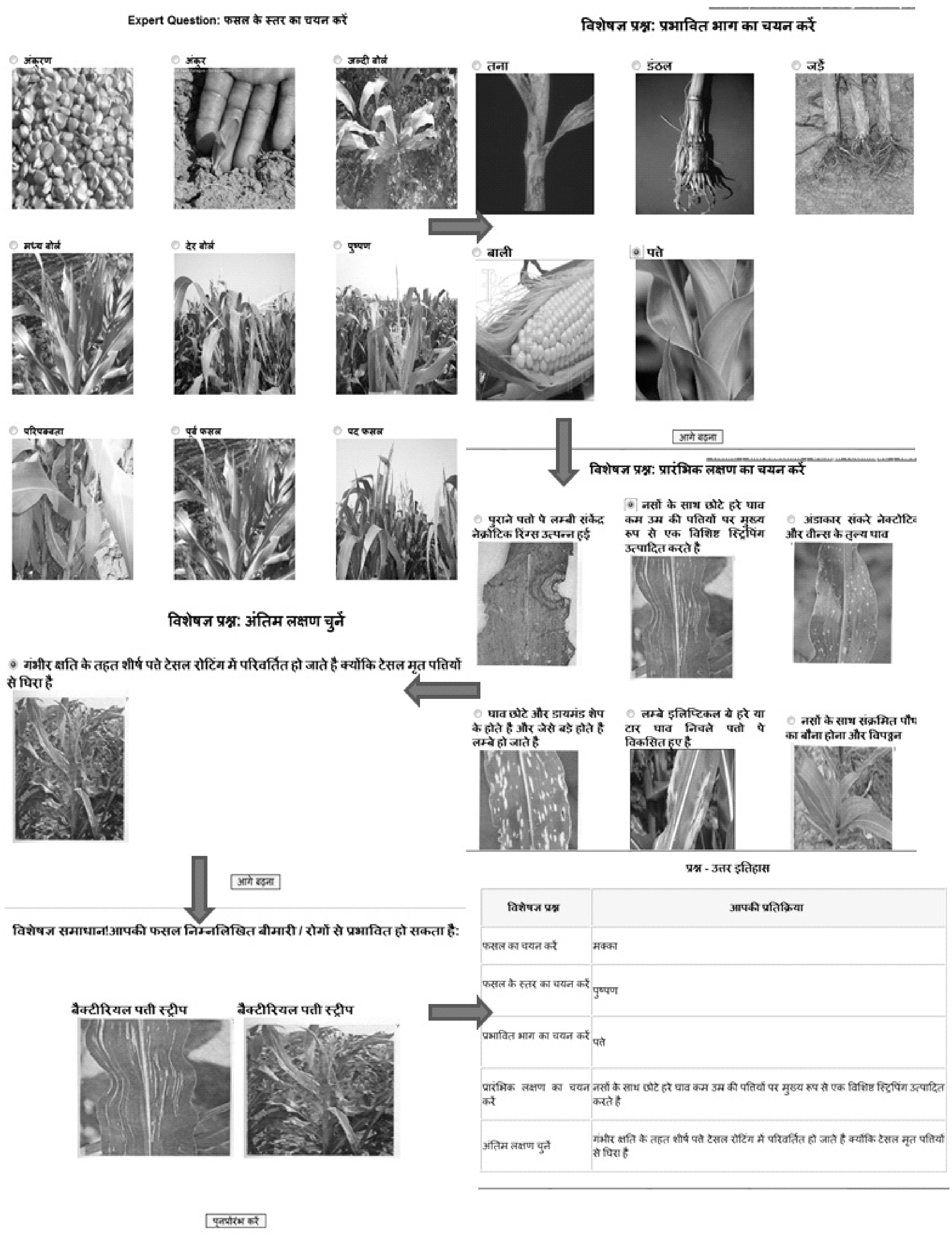

Fig. 9: Question answer session for disease diagnosis in Maize

Table 2. SPARQL queries generated for insect identification diagnosis module

No.	SPARQL Query	No.	SPARQL Query
1	SELECT?tree WHERE {?tree : has_sequence ? sequence. ?tree :has_language “hindi”}	2	SELECT ?select_part ? query_clause ?ques WHERE {:कीट;अनुक्रम :has_ques1 ?qid. ?ques : has_id ?qid.?ques:has_select_clause ? select_part. ?ques :has_query_clause? query_clause. ?ques :has_language “hindi” }
3	SELECT distinct ? Crop ?path WHERE {?probable_pest : attacks ?Crop. ?Crop : has_language “hindi”. ?pic :pic_of_what ?Crop. ?pic :has_path ?path.	4	SELECT distinct ?Crop_Stage? path. WHERE { ?probable_pest :attacks ? : मक्का : मक्का :has_stage ?Crop_Stage. ?Crop_Stage :has_language “hindi”. ?pic :pic_of_what ?Crop_Stage. ?pic :has_path ?path} ?Crop_Stage:has_order ?A.} order by (?A)
5	SELECT distinct ?Parts_Affected ?path WHERE { ?probable_pest :attacks : eDdk - मक्का : has_stage : पुष्पण - पुष्पण : has Parts?Parts_Affected. ?Parts_Affected : has_language “hindi”. ?pic :pic_of_what ?Parts_Affected. ?pic :has_path ?path}	6	SELECT distinct ? Intitial_Nature_of_Damage? path WHERE { ?probable_pest :attacks : मक्का - मक्का :has_stage : पुष्पण - ?probable_pest :affects_parts : बाली_- ?Initial_Nature_of_Damage :is_of_type : Initial. ?probable_pest :has Nature of Damage ? Initial Nature of Damage. ?Initial_Nature_of_Damage :has_language “hindi”. ?pic :pic_of_what ?Initial_Nature_of_Damage. ?pic :has_path ?path}
7	SELECT distinct ?Initial_Nature_of_Damage?path WHERE { ?probable_pest :attacks : मक्का मक्का : has_stage : पुष्पण - ?probable_pest :affects_parts: ckyh_. ?Initial_Nature_of_Damage :is_of_type:Initial. ?probable_pest :hasNatureofDamage ?Initial_Nature_of_Damage. ?Initial_Nature_of_Damage :has_language “hindi”. ?pic:pic_of_what ?Initial_Nature_of_Damage. ?pic :has_path ?path}	8	SELECT distinct ?Final_Nature_of_Damage?path WHERE { ?probable_pest :attacks : मक्का मक्का : has_stage : पुष्पण - ?probable_pest :affects_parts: ckyhs_. ?Initial_Nature_of_Damage :is_of_type:Initial. ?probable_pest :has Natur of Damage : युवा_लार्वा_अंडे_मास_के_आसपास_ के_पत्त_पर_फीड_करना_शुरू_करते_है_और_बाद_में_वोल_के_ भीतर_और_पुराना_लार्वा_डंठल_में_बोर_हुआ_पत्ते_शीथ_के_पीछे_नोड_में ?Final_Nature_of_Damage :is_of_type :Final. ?probable_pest :has Nature of Damage? ?Final_Nature_of_Damage. :has_language “hindi”. pic :pic_of_what ??Final_Nature_of_Damage. pic :has_path ?path}
9	SELECT distinct ?probable_pest?path {?probable_pest :attacks :मक्का मक्का : has_stage : पुष्पण ?probable_pest :affects_parts :बाली ?Initial_Nature_of_Damage :is_of_type:Initial. ?probable_pest :hasNatureofDamage: युवा लार्वा अंडे मास के पत्ते पर फीड करना शुरू करते है और बाद में वोर्ल के भीतर और पुराना लार्वा डंठल में बोर हुआ पत्ते शीथ के पीछे नोड में ?Final_Nature_of_Damage :is_of_type :Final. ?probable_pest :has Nature of Damage: लार्वा बोर किया जाता है और विकसित हो रहे टेसल मक्के और मध्य पसलियों को नुकसान पहुचाता हैं ?probable_pest :has_language “hindi”. :pic_of_what ?probable_pest. ?pic :has_path ?path}		

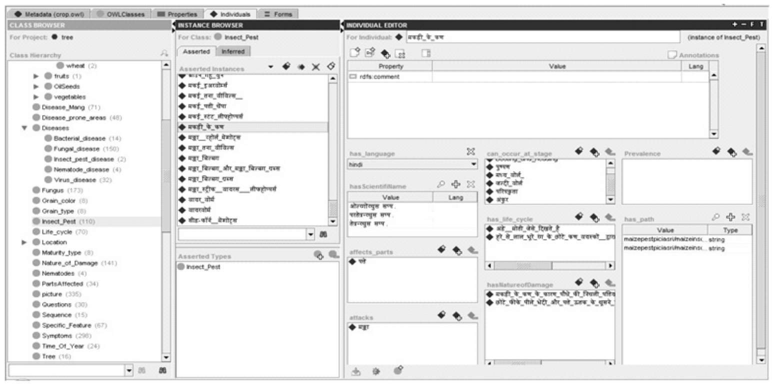

Fig. 10: Class Insect_Pest with its individuals showing various object and Data type properties

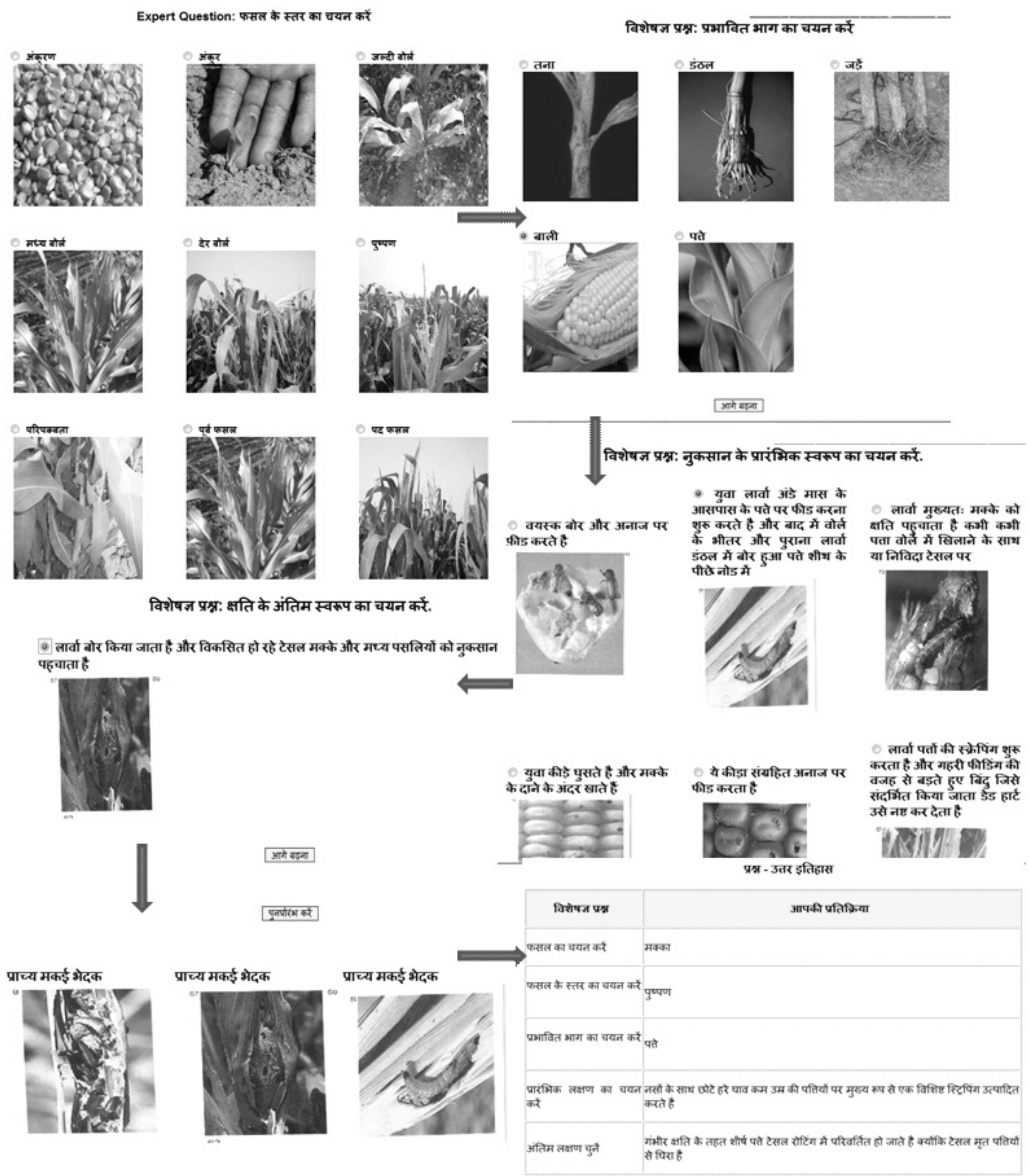

Fig. 11. Question answer session for insect identification in Maize

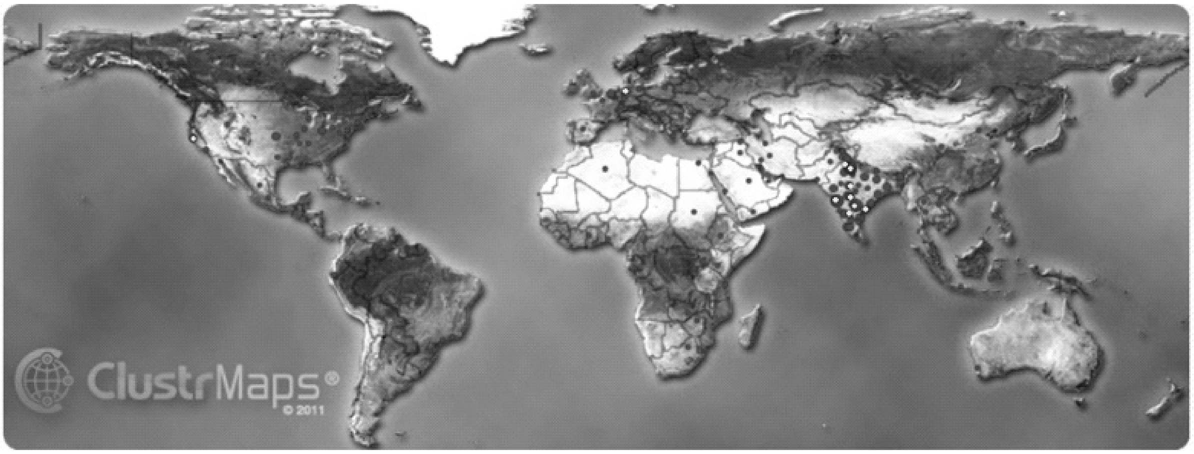

Fig. 12: Usage of Maize Agridaksh

into the subsequent relevant question or a final answer. Table 1. shows the sequence of queries generated when a user traverses through a question answer session. Query number 7 uses all previous input given by user, i.e. crop, crop stage, parts affected and initial symptoms to extract the most likely final symptoms, leaving the irrelevant options. Query 8 finally results into the diagnosis of disease. Fig. 9 shows the series of questions and available options that user may choose to diagnose the disease.

7. IDENTIFICATION OF INSECTS FOR MAIZE

In this section, the class hierarchy, properties, relation between classes and properties created in ontology are described. The classes Crop_stage, Crops, PartsAffected are same as Disease Diagnosis. Instead of Classes like Diseases and Symptoms, classes Insect_Pest and Nature_of_Damage are created.

Insect_Pest: It is a class for insects that affects crops adversely. Individuals of this class have one restriction namely, they have at least one nature of damage. It also has many properties like attacks and affects_parts whose domain are Crops and PartsAffected respectively. Besides this, it has other properties namely, hasScientificname, can_occur_at_stage etc. Fig. 10 shows an individual, asian maize borer being created. The individual editor shows various object and datatype properties in which an individual of this type is involved.

Nature_of_Damage: Class for common nature of damage for crops describes attacks by various insects. Individuals of this class have restriction that they are nature of damage of at least one insect. The individuals of this class are involved in property NatureofDamageOf which related to Insect_Pest. It has an object property is_of_type which is used to indicate whether damage is of initial or final type.

Identification of insect can be done in a similar fashion as disease diagnosis by appropriately selecting the option from the page that appears after clicking Problem Identification link. The user enters into question answer session and follows the path as shown in Fig. 11. A SPARQL query is generated dynamically after each question. Table 2 shows the queries generated for insect identification in Maize.

The presented system is available online at http://agridaksh.iasri.res.in and Fig. 12 shows its usage from across the world. It has got over 2000 hits from March 2012 to Jan 2013.

8. CONCLUSION

The Ontology based Expert System for identification of Maize diseases and insect-pests has been presented in this paper. Ontology being the latest knowledge representation technique allows the expert system to maintain a dynamic knowledge base and thus provides the user with the result based on the recent knowledge. This framework does not pose any restriction on the domain experts for regular updating and modification of the underlying ontology. This expert system utilizes Ontology as knowledgebase because populating knowledge in ontology is easy using available Ontology editors such as Protégé/Web Protégé. In traditional rule based systems, rules were to be extracted manually and then fed into the expert system whereas here it is not needed. This system is scalable and knowledge about number of diseases and insects can be added that attacks other crops and the system will be able to do insect and disease identification for all of them. The Ontology based Expert System for Maize Diseases and Insect-Pests has the capacity to carry large amount of research work done by agriculture scientists and act as a tool for transfer of knowledge from scientists to the farmers and helps in enhancing the efficiency, crop yield and productivity.

REFERENCES

Bedi, P. and Marwaha, S. (2005). Framework for Ontology based expert systems: Disease and pests identification in crops - A case study. In: *International Conference of Artificial Intelligence* (*ICAI-05*), 256-259.

Bedi, P. and Marwaha, S. (2004). Designing Ontologies from traditional taxonomies. In: *International Conference on Cognitive Science*, Allahabad, India.

Berners-Lee T., Hendler, J. and Lassila, O. (2001). *The Semantic Web. Scientific American*, **284(5)**, 34-43.

Brickley, D. and Guha, R.V. (1999). Resource Description Framework (RDF) Schema Specification 1.0. W3C Proposed Recommendation, http://www.w3.org/TR/1999/PR-rdf-schema-19990303

Dean, M., Connolly, D., Harmelen, F.V., Hendler, J., Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F. and Stein, L.A. (2003). OWL Web Ontology Language Reference. W3C Working Draft, http://www.w3.org/TR/2003/WD-owl-ref-20030331

Gennari, J.H., Musen, M.A., Fergerson, R.W., Grosso, W.E., Crubezy, M., Eriksson, H., Noy, N.F. and Tu, S.W. (2003). The evolution of Protege: An environment for knowledge-based systems development. *Intern. Jour. Human Computer Studies,* **58(1)**, 89-123.

Golbeck, J., Fragoso, G., Hartel, F., Hendler, J., Parsia, B., and Oberthaler, J. (2003). The national cancer Institute's thesaurus and ontology. *Jour. Web Semantics*, **1(1)**, 7-26.

Gruber, T.R. (1993). *Toward Principles for the Design of Ontologies Used for Knowledge Sharing.* KSL-93-04, Knowledge Systems Laboratory, Stanford University, Available at http://ksl-web.stanford.edu/

Horrocks, I., Patel-Schneider, P.F. and Harmelen, F.V. (2003). From SHIQ and RDF to OWL: The making of a Web Ontology Language. Journal of Web Semantics: *Science, Services and Agents on the World Wide Web*, **1**, 7-26.

Kifer, M., Bruijn, J.D., Boley, H. and Fensel, D. (2005). A realistic architecture for the semantic web. *Lecture Notes in Computer Science,* **3791**, 17-29.

Knublauch, H., Musen, M.A. and Rector, A.L. (2004). Editing description logic ontologies with the Protégé OWL plugin. In: *2004 International Workshop on Description Logics (DL2004)*, **104**. Whistler, British Columbia, Canada.

Leon, C.D. (1996). Maize diseases-a guide for field identification. Centro International de Mejoramiento de Maiz y Trigo International Maize and Wheat Improvement Center (CIMMYT).

Marwaha, S., Kumar, V. and Bahal, R. (2002). Web enabled expert system of extension: A need of time in agriculture for developing countries. In: *Intern. Agronomy Conference, IARI*, New Delhi.

Noy, N.F. and Musen, M.A. (2000). PROMPT: Algorithm and tool for automated ontology merging and alignment. *Proceedings of the Seventeenth National Conference on Artificial Intelligence* (*AAAI-2000*), Austin, TX.

Noy, N.F. and McGuinness, D.L. (2001). Ontology development: a guide to creating your first ontology. Stanford Knowledge Systems Laboratory Technical Report KSL-01-05, Stanford Medical Informatics Technical Report SMI-2001-0880.

Ortega, C.A. (1987). Insects pests of maize–A guide for field identification. Maize program, CIMMYT.