# Constructing a Semantic-Based Image Retrieving system – Image Semantic Searching System (ISSS)

**Yi Liu[1], Peiyi Xiao[1], and Michael C. Wimberly[2]**
[1]Dept. of Electrical Engineering and Computer Science, South Dakota State University, Brookings, USA
[2]GISc Center of Excellence, South Dakota State University, Brookings, USA

**Abstract -** *Because of the great volume of image resources produced and gathered on the web for public browsing, image retrieving tools and systems play an important role for users in retrieving image resources. A novel image retrieving system, the Image Semantic Search System (ISSS), was designed and implemented based on semantic web concepts and techniques. The paper illustrates the methodology for designing the sematic search system to provide user friendly interfaces and relevant searching results. The paper also presents the architectural design and implementation of the system and uses a case study to demonstrate the application of ISSS.*

**Keywords:** semantic web; image search system; web application

## 1 Introduction

Due to the large volume image resources produced and gathered on the internet, it becomes difficult for end users to retrieve desired resources. Although numerous of image retrieving systems have been developed based on different methodologies, there are still some key problems that have not been adequately resolved. For example, users typically take too many responsibilities for retrieving relevant results. Therefore, in order to provide a good searching experience to the users, there is a need for developing more result relevant and user friendly image retrieving systems.

A prevailing response to this need is semantic-based image retrieving systems. Semantic-based approaches extract the semantic meaning of the image resources using Semantic Web [1] techniques to interpret both the resources and the users in order to reduce users' responsibilities and always provide users with relevant results.

This research aims to construct a novel semantic-based image retrieval system *Image Semantic Searching System* (ISSS) to provide users better experience on searching images. The ISSS is designed for both image seekers, who search for the images, and image producers, who generate or publish the images. For the image seekers, ISSS should have simple and easy-to-use interfaces and always provide relevant results to users without any duplicated or wasted effort. For the image producers, ISSS should have user friendly interfaces and should support reusability in other websites.

Below lists the objectives for the design of ISSS.

Objective 1: The system should provide user friendly interfaces for both image seekers and image producers.

Objective 2: The system should always provide the most relevant results for image seekers.

For each search, the system will provide the most relevant result if it can be found; otherwise, it will return no result.

Objective 3: Although this image searching system was initially designed for searching map images stored in a web atlas, it should provide an easy way to make it reusable in other websites as an image search component.

The paper focuses on the construction of ISSS. Section 2 illustrates the background information that is used in ISSS. Section 3 illustrates the methodology of the design of the ISSS. Section 4 shows architectural design of the ISSS and Section 5 sketches the implementation. Section 6 demonstrates how to plug-in ISSS in a real world application. Section 7 summarizes and discusses the results.

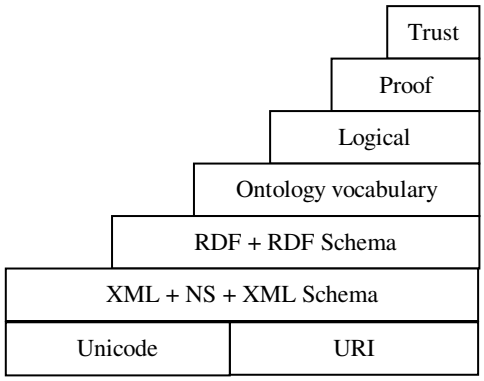

Fig.1. Semantic Web Layers

## 2 Background

### 2.1 Semantic web

The semantic web technique is adopted for developing the ISSS. The semantic web is used to help computers "understand" the information on the web so that they can support richer discovery, data integration, navigation and automation of tasks [2, 3].

The Semantic Web principles are implemented in the layers of web technologies and standards shown in Fig. 1[1]. The Unicode and Uniform Resource Identifier (URI) layer guarantees the international characters sets to be used to provide identical means for resources or objects. The XML layer ensures the Semantic Web definitions can be integrated with other XML based standards. The RDF and RDF Schema layers are responsible for describing each resource or object by make statements about the URI of resources. The Ontology layer defines the relationships between the different concepts of each vocabulary. The Logic layer enables the writing of rules while the Proof layer executes the rules and evaluates together with the Trust layer mechanism for applications whether to trust the given proof or not. At the present time, the Logic, Proof and Trust layers are still under development and have not yet been incorporated into the application.

## 2.2 Dublin Core

The Dublin Core metadata terms [4,5] are a set of vocabulary terms which can be used for the purposes of discovery and generic resource description. The terms can be used to describe a full range of web resources: video, images, web pages, and etc. The simple Dublin Core Metadata Element Set (DCMES) consists of 15 metadata elements [6]:

title identifier source language relation
coverage rights creator subject description
publisher contributor date type format

These metadata terms can represent the characteristics of each resource in different perspectives, and each resource can be described or organized around these terms. Based on different needs, terms from the set can be adopted to describe new resources, or the original term set can be extended to add more terms. For ISSS, we adopt title, subject, coverage, date, and format to describe image resources, and we rename them to be *theme, event, location, period* and *style,* respectively.

# 3 Methodology

In order to satisfy objective 2 to provide the most relevant results to image seekers, ISSS should interpret both users' input and resources correctly. We propose a methodology for describing resources, extracting information, organizing resource properties and inferring resources. The whole interpretation process, as shown in Fig. 2, is divided into three parts: resource description, information extraction, and resource inference.

## 3.1 Resource Description

One of the Semantic Web techniques, Resource Description Framework (RDF), is used to organize and describe resources thoroughly. For interpreting each resource effectively, a resource model with five basic properties (adoption of the Dublin core) – *theme, event, location, period* and *style* is designed and applied into RDF.

It is not reasonable to produce duplicated resources, so we assume that there are no duplicated resources in storage, which means that no two resources have identical properties. In this way, the properties make each resource unique and searchable. For example, a resource whose properties are West Nile Virus (theme), Incidence Rate (event), South Dakota (location), 2011 (period) and JPG (style) differs from another newly produced resource with the properties – West Nile Virus (theme), Incidence Rate (event), South Dakota (location), 2012 (period) and JPG (style). Even though they have similar properties, the difference of their period property makes them different and unique.

By applying the resource description model, each resource can be interpreted by checking and identifying all its associated properties. If all the properties of a resource are exactly match those a user indicates, then the resource is the user's wanted result. To get relevant results, interpreting only resources is not enough. It is not possible to match a resource without users' indication as reference. So, another important issue is how to interpret input from the users.

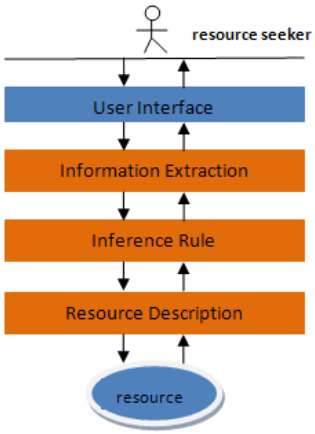


Fig. 2. Interpretation process

## 3.2 Information Extraction

Information Extraction is used to analyze user input and extracts useful information to find out users' target results. To avoid missing any useful information and provide flexibility of user input, the system extracts information in two different ways - Syntax Interpretation and Semantic Interpretation.

The syntactic styles of users' input can vary, based on different spelling habits. For example, "West Nile Virus" is treated as same as "WNV" or "West Nile." For interpreting a word or a phrase in different syntactic styles, it is necessary to collect and organize the different syntactic forms of that word or phrase. Another way to interpret a phrase or a word in a different syntax is to identify its misspelling forms. For example, a misspelled phrase "West Nile Virous" has explicit meaning of "West Nile Virus". Thus, organizing and identifying the misspelled phrases also facilitates interpreting

user input. A Syntax Thesaurus is attached to the system to help the system identify user input in various syntactic forms. The Syntax Thesaurus organizes common words and their spelling and misspelling variations.

The Semantic Interpretation focuses on words' synonyms. The meaning of a word can be represented by identifying and interpreting its synonyms. For instance, "incidence rate" has the same meaning as "incidence proportion". Similar to the process of Syntax Interpretation, a Synonym Thesaurus is constructed to identify the synonyms of the core words used to describe a resource.

### 3.3 Inference Rule

An Inference Rule is the act of inferring the unknown information of a resource based on the interpreted information. Due to the large volume of resources that may accumulate in resource storage, we designed an Inference Rule for inferring the target resources by narrowing down the searching range purpose. Fig.3 shows the Inference Rule.

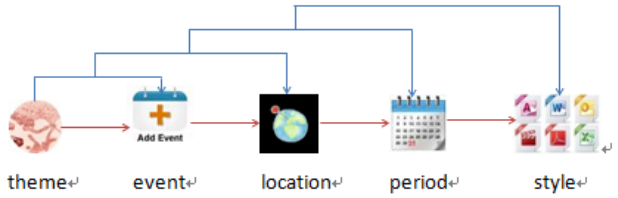


Fig.3. Inference Rule

Consider each resource property in the resource model as theme (T), event (E), location (L), period (P) and style (S), and then we have the following inference logic:

$T \rightarrow E$
$\neg T \rightarrow \neg E$
$(T \rightarrow E) \wedge T \rightarrow L$
$(T \rightarrow E) \rightarrow L \wedge T \wedge E \rightarrow P$
$((T \rightarrow E) \rightarrow L) \rightarrow P \wedge T \wedge E \wedge L \rightarrow S$
$(T \wedge \neg E) \rightarrow \neg L$
$(T \wedge E \wedge \neg L) \rightarrow \neg P$
$(T \wedge E \wedge P \wedge \neg E) \rightarrow \neg S$
$(T \rightarrow E) \wedge (E \rightarrow L) \rightarrow (T \rightarrow L)$
$(T \rightarrow E) \wedge (E \rightarrow L) \wedge (L \rightarrow P) \rightarrow (T \rightarrow P) \vee (E \rightarrow P)$
$(T \rightarrow E) \wedge (E \rightarrow L) \wedge (L \rightarrow P) \wedge (P \rightarrow S) \rightarrow (T \rightarrow S) \vee (E \rightarrow P) \vee (E \rightarrow S) \vee (L \rightarrow S)$
$(T \rightarrow E \wedge E \wedge P \wedge \neg E) \rightarrow \neg S$

A traditional method for inferring resource information is schematizing the content of Resource Description Framework (RDF) by using Resource Description Framework Schema (RDFS) and using RDF query language such as SPARQL [7] to parse the RDF and RDFS files to query and infer resource information. This requires the construction of a resource schema file and the annotation of the relationships between the information. In addition, an extra query language is needed for querying the resource description and resource description schema files. The query processes may be complicated and time consuming based on the construction and format of the parsed files. Our method focuses on constructing a resource information file, which is formatted automatically based on the Inference Rule when resources are generated. There is no need to construct a resource schema file, or use a query language and the associated query rule to get the information. Therefore, our method is more efficient and convenient compared to the traditional method.

## 4 Architectural Design of ISSS

The following assumptions are made to serve as the basis of the architectural design of ISSS.

(1) The resources include many different image types such as JPEG, PDF, PNG, KML, and KMZ and so on. The image resource is stored in the system and available for retrieval.

(2) Two different user interfaces are provided for two types of users – resource seekers and resource providers. Resource seekers search and view the resources stored in the system through a data reading interface, and resource providers generate the resource and store them in the system through a data providing interface.

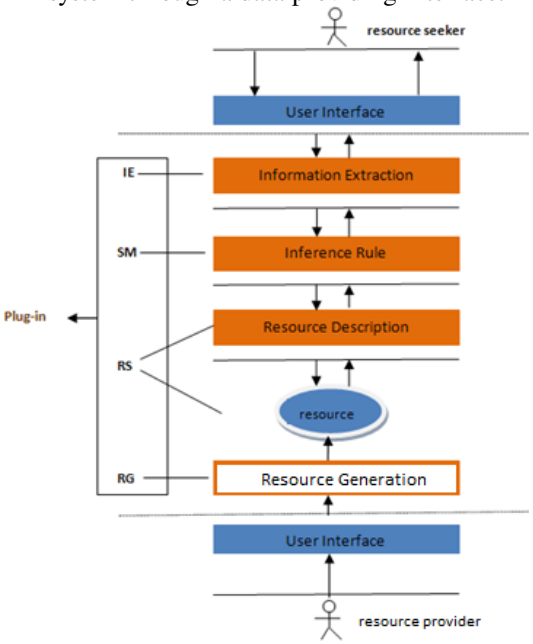


Fig. 4. ISSS Components

ISSS adopts the client-server architecture for the entire system. Resource seekers and resource providers are on the client side and they are provided user interfaces to interact with the server side. Components *Information Extraction (IE)*, *Search Engine (SE)*, *Resource Storage (RS)* and *Resource Generation (RG)*, are sitting on the server side to perform the tasks of storing resources, analyzing the resource seekers' inputs, and conducting searching.

*Information Extraction (IE) Component*

IE maps to Information Extraction that is described in the methodology. IE is responsible for collecting the resource seeker's input and extracting the semantic meaning of the

input through the data reading interface. A Syntax Thesaurus and a Synonym Thesaurus are attached to the IE component to check and interpret the input information.

*Resource Storage (RS) Component*

RS maps to the Resource Description for storing all the resources and monitors the requests from other components. Each resource stored in RS is described and organized based on the applied Resource Description Model. RS also provides the functionality for retrieving resources from storage and sending them to other components. For example, RS will provide the requested resource to SE when it sends a request.

*Search Engine (SE) Component*

SE maps to Inference Rule for searching target resources. It carries users' input and accesses other components to query and check resources, and then provides resource properties information or searching results to users. Theme, event, location, period and style are used as the basic properties to describe each resource. The inference rule specified in section 3 is applied into SE for identifying the resource properties, checking the resource availability and reducing the target resource's range. SE communicates with IE and gets the extracted properties of the resources from it. With the known resource properties from IE, SE provides a user-friendly interface to get the unknown properties from the users through the interface. After all the properties are collected by SE, it interacts with the RS to query resources.

*Resource Generation (RG) Component*

RG allows the resource provider to generate resources and store them into the RS. The operations and interfaces of RG are provided in the resource generator view. It provides three operations, *addForResource(),addForSourceCheck()*and *addForThesaurus()*. After generators finish creating the resource, the operation *addForResource()* stores the created resource into RS, operation *addForSourceCheck()* puts the resource property information into the Inference Rule file, and *addForThesaurus()* sends the resource property synonym information to the Synonym Thesaurus.

Resource providers produce resources through the user interface provided by RG. RG communicates with RS and stores all the produced resources in RS. IE monitors users' searching requests and extracts the information from their input. It then contacts SE to send users' requests and the related data of resources. Finally, SE searches the resources and sends the requests to RS to retrieve and display the search result to users. Their interactions are illustrated in Fig.5.

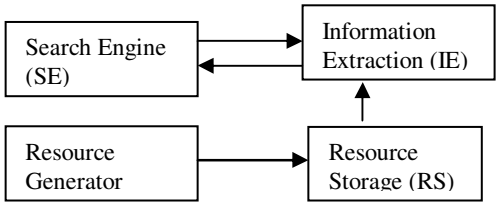


Fig.5. ISSS component interactions

# 5 Implementation

The server side of ISSS was coded in PHP 5[8]. The Client side scripts were developed in HTML for the structure of user interfaces, JavaScript and AJAX [9] for the interactions, and Cascading Style Sheets (CSS) [10] for adding styles to the web page layouts.

*RDF and Inference Rule*

All the resources are defined and stored in the RDF based on the applied Resource Description Model. In RDF, XML is used to express the resource information in form of triples: Subject-Property-Value. User defined tags are used in RDF to describe the attributes or characteristics of each resource. The name/link of each image resource is treated as the Subject. The property of description model is the Property, and the text content of the property is the Value. Fig. 6 shows an example of image resource description. Tag <*regardTo*> is for the property *Theme*, displayed is for *Event,* <*recoding*> is for *Period*, <*occurred*> is for *Location*, and <*madeInto*> is for *Style*. The inference rules for resource descriptions are stored in an inference rule file shown in Fig. 7.

```
<resource>
 <image id="0" name="Timeseries incidence rates from 2002-2011"
      link="/model/maps/asia_trip_n.kmz">
    <regardTo resource="west nile virus" />
    <displayed resource="incidence rate" />
    <recording resource="2002-2011" />
    <occurred resource="northern great plains" />
    <madeInto resource="kml" />
 </image>
</resource>
```

Fig. 6. A sample resource description with one piece of image resource

```
<?xml version="1.0" ?>
- <source>
  - <theme id="west nile virus">
    - <event id="incidence rate">
      - <area id="northern great plains">
        - <time id="2002-2011">
            <type id="kml" />
          </time>
        </area>
      </event>
    </theme>
  </source>
```

Fig. 7. Inference rule file

*Implementation of Information Extraction (IE) Component*

The main operation of the IE component is *extractProperties()*, which tries to extract all the properties information of resources. This operation includes several sub functions, each of which is for extracting one of the properties in the Resource Description Model. During information extraction, IE opens and parses the attached syntax and synonym thesauruses for the syntax and semantic interpretation. The thesauruses are well organized by using XML. Fig. 8 shows the partial codes of IE and Fig.9 and 10 show the partial content of the thesauruses.

```
class InformationExtraction
{ function extractProperties($content, $theme, $event,
        $location, $period, $style)
  {  extractTheme();        extractEvent();        extractLocation();
     extractPeriod();        extractStyle();
  }
  …
}
```

Fig. 8. Partial code of IE component

```
<?xml version="1.0"?>
<syntax from="WIKI MISSPELLING DATABASE">
 <aFile commnet="all the words start with alphabet-a">
  <author>
    <autor />
  </author>
  <authority>
    <autority />
  </authority>
 </aFile>
</syntax>
```

Fig.9. Partial code of syntax thesaurus

```
<?xml version="1.0" ?>
- <synonym from="resource generator">
  - <event orginal="incidence rate">
      <seeAlso>incidence proportion</seeAlso>
      <seeAlso>rate of incidence</seeAlso>
      <seeAlso>proportion of incidence</seeAlso>
    </event>
  - <period orginal="2010-2012">
      <seeAlso>2010 to 2012</seeAlso>
      <seeAlso>20010</seeAlso>
      <seeAlso>2011</seeAlso>
      <seeAlso>2012</seeAlso>
    </period>
  - <location orignal="northern great plains">
      <seeAlso>northern plains</seeAlso>
    </location>
  </synonym>
```

Fig.10. Partial code of synonym thesaurus

*Implementation of Resource Storage (RS) Component*

In the RS component, the method displayResource() identifies resource types by parsing RDF and gets resources for users. RDF is easily parsed using PHP. Because RDF is an XML based file, many XML query techniques such as XPath, Simple XML, XML DOM and so forth that can be used to read and write data to RDF. In the implementation of ISSS, XPath is used for parsing RDF. Fig.11 shows the partial code of the RS component.

```
class ResourceStorage
{ function displayResource($theme, $event, $location,
     $period, $style)
 {//check resource type
  checkResourceType ($style);
  //get resource from the resource storage
  fetchResource($theme, $event, $location, $period, $style);
 }
 function fetchResource($theme, $event, $location,$period,
$style)
  {  ...
    $xmlRDF = new DOMDocument();
    $xmlRDF -> load(RDF);
    $xmlRDFXpath = new DOMXPath($xmlRDF);
    $resource = $xmlRDFXPath->
  query("//theme[@id='".$theme."']/
  event[@id='".$event."']/location[@id='".$location."']/
  period[@id='".$period."']/style[@id='".$style."']");
  ...
  }
}
```

Fig.11. Partial code of RS component

*Implementation of Search Engine (SE) Component*

In SE component, the operation *searchProperties()* parses the inference rule file and infers resource properties via the extracted information or user's indication. After all the needed properties are inferred and collected by SM, it communicates with RS to get the target resource. Fig. 12 shows the partial code of the function searchProperty.

```
class SearchEngine
{  ...
  //search the properties of target resource
  function searchProperty()
  {  searchTheme();       searchEvent();       searchLocation();
     searchPeriod();       searchStyle();
  }
}
```

Fig. 12. Partial code of SE component

*Implementation of Resource Generation (RG) Component*

RG provides a user interface in resource generator's view to let resource generators indicate all the information for each resource property when they are creating resources. The operation *addForResource()* takes resource properties' information and communicates with RS to write the information into RDF. *addForSourceCheck()* opens and parses the Inference Rule file to add the resource information into the file. RG collects synonym information of resource properties from the resource generator, and *addForThesaurus()* adds synonyms into the Synonym Thesaurus. Fig. 13 shows the partial code of the implementation of the RG component.

```
class GenerateResource
{  ...
 //function to add resource to RDF
 function addResource($staticmapfiles, $name, $theme,
  $event, $location, $period)
 {...}
 //add resource to inference
 function addForResourceCheck($staticmapfiles, $theme,
   $event, $location, $period)
 { ... }
 //add resource to thesaurus
 function addForThesaurus($theme, $event, $location,
     $period, $synonym1, $synonym2, $synonym3)
 {...}
}
```

Fig. 13. Partial code of RG component

*Implementation of ISSS Plug-In*

ISSS plug-in class is implemented to address objective 3 for providing reusability. The independent module ISSS

plug-in can be instantiated for plugging ISSS into any existed web site for image retrieving. The plug-in class provides operations for configuring both the source file and the destination file, and then it ships ISSS to the plugged site. Partial code of the ISSS Plug-In is shown in Fig. 14.

```
class ISSS
{ protected $newPath;     protected $sourcePath;
  protected $jsPath;        protected $cssPath;
  protected $pluginFile;    protected $componentName;
   public function ISSSPlugIn()
   {  //config ISSS component
      $this->configApplication();
      //setup ISSS component
     $this->setupModules($this->newPath, $this->sourcePath);
    //plug ISSS component
    $this->PagePlugIn();
   }
  public function configApplication()
  {    // code for setting up the $componentName, $newPath,
       //$sourcePath,  $jsPath, $cssPath, pluginFile
  }
   public function shipFiles($newAppPath, $fwaPath)
   { // code for shipping the needed files to the destination   }
   public function PagePlugIn()
   {  // code for plug the search page to the website }
  }
```

Fig. 14. Partial code of ISSS plug-in

# 6 Use of ISSS

## 6.1 Plug In ISSS

The design and implementation of ISSS supports reusability. It can be plugged into any website as an image retrieval system. To plug ISSS in, the developer needs to (1) configure and setup ISSS by indicating both the ISSS source file path and target path; and (2) plug- ISSS user interface into any web page by indicating the path of that page. The following shows an example of plugging ISSS to an existing website EASTWeb[11]. EASTWeb is a collaborative project involving scientists from South Dakota State University and the USGS Center for Earth Resources Observation and Science (EROS), along with partners from government agencies, and nongovernmental organizations. EASTWeb collects various types of images of public health maps, and it provides public access to the web users to retrieve these image resources. ISSS is plugged into EASTWeb to facilitate web users to retrieve web resources.

i. Indicate all the paths for plugging ISSS into EASTWeb

```
$newPath="eastweb/ISSS";
$sourcePath="web/ISSS";
$jsPath="web/js";
$cssPath="web/css";
$pluginFile="eastweb/homepage.html";
$componentName="eastweb/search";
```

ii. Instantiating the new ISSS

```
$eastwebISSS=new ISSS();
```

iii. Invoking the plug-in method

```
$eastwebISSS->ISSSPlugIn($newPath, $sourcePath,
    $jsPath, $cssPath, $pluginFile, $componentName);
```

After the setup and plug-in of ISSS is complete, the search interface is displayed in the plugged page. EASTWeb ISSS is ready to be used by the end users to search image resource.

## 6.2 Resource Seeker's View

ISSS provides a search text box to users for typing target resource information and a button to start searching resources. Fig.15 displays the EASTWeb searching UI.

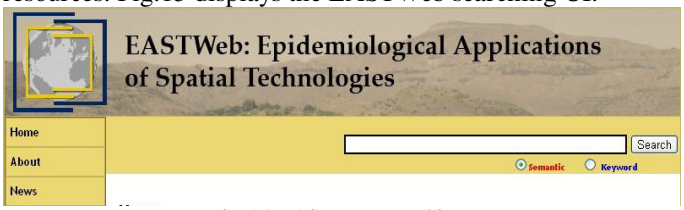


Fig. 15. EASTWEB searching UI

After a user types the content for describing the target resource, ISSS will extract the information and try to get the result by inferring each property of the resource. ISSS provides resource seekers user friendly interfaces with recommended resource properties for indicating "unknown" properties. Fig. 16-19 shows the searching processes of property indication. In the example, the resource information "wnv hot spot kml" is typed in the text box by users. "wnv" is extracted by ISSS to be the theme of the resource, "hot spot" and "kml" are treated as the resource's event and style. For the unknown information, it provides the recommended options of location and period to users.


Fig. 16. EASTWeb resource location indicating UI


Fig. 17. EASTWeb resource period indicating UI

## 6.3 Resource Generator's View

ISSS provides user interfaces for generating resource in resource generator's view. Resource generators can create resources by giving all the information of the resource. As soon as the resource is created by the generator, it is stored in Resource Storage. Fig.20 shows the resource generation

interface. It is the generator's responsibility to provide the information of *theme*, *event*, *location*, *period*, and *style*. When an image is being uploaded, its *style* property can be automatically extracted by the system from its type extension. The generator can also illustrate the synonyms that will the used in the Synonym Thesaurus for semantics interpretation.

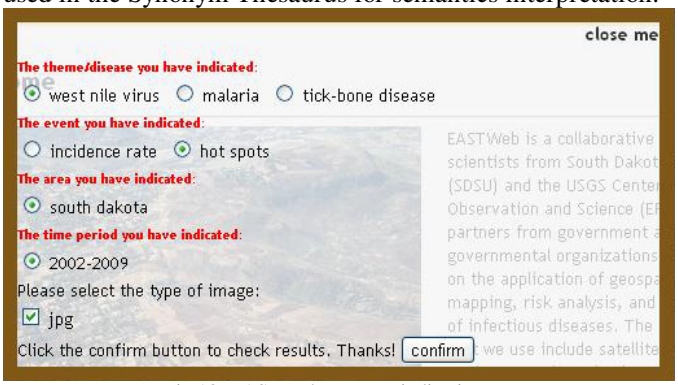


Fig.18. EASTWeb resource indicating UI


Fig.19. EASTWeb result UI


Fig.20. EASTWeb resource generation UI

# 7 Discussion

ISSS is designed and developed for searching and retrieving web image resources and it facilitates the construction of web based image search system websites. ISSS is designed with the aim for user friendliness, relevant search results and reusability of implementation. In order to address the user friendliness and result relevancy, we adopted the sematic web approach and designed a methodology for describing resources, extracting information, organizing resource properties and inferring resources to interpret both users and resources for retrieving target resources thoroughly. The architectural design of the system addressed the reusability.

Section 6 uses a case study to presents the reusability of the ISSS system. The achievement of the user-friendliness and result relevancy is evaluated based on the results from a focus group survey. Two Focus Groups with eight members in each participated in a survey containing questionnaires addressing the user friendly interfaces and search result relevancy. The first group was given a 35-minute presentation on ISSS and the second group was given a 5-minute instruction on how to use the semantic search function in a testing website. None of the participants has experience in developing a semantic search engine. A specific task was given to the participants for conducting a search. 94% of the participants think that ISSS is user friendly and the user interface is meaningful and straightforward to use. 100% of participants agree with that ISSS always provides relevant results without any duplicated or useless accompany. Based on these results, we believe ISSS successfully achieved the objectives of user friendly interface and result relevancy.

It can be concluded that ISSS is an image retrieving system, which is user- friendly, result relevant, and reusable.

# 8 Acknowledgment

This work is supported by NIH grant R01AI079411 "*An Integrated System for the Epidemiological Application of Earth Observation Technologies*".

# 9 References

[1] W3C Semantic Web Activity. http://www.w3.org/2001/sw/.

[2] T. Berners-Lee. "Weaving the Web". Harper, San Francisco, CA, 1999.

[3] D. Fensel, Wahlster, W., Lieberman, H., Hendler, J., eds.: "Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential". MIT Press, 2002.

[4] G. Jane. "Dublin Core: History, Key Concepts, and Evolving Context". DC-2010 International Conference on Dublin Core and Metadata Applications. October 20, 2010.

[5] P. Steve. "Expressing Dublin Core in Topic Maps". Lecture notes in Computer Science volume 4999, pp. 186-197. 2008.

[6] Dublin Core Metadata Initiative website. http://dublincore.org/documents/dces/.

[7] SPARQL. http://en.wikipedia.org/wiki/SPARQL.

[8] PHP: Hypertext Preprocessor. http://us.php.net/.

[9] J.J Garrett. "Ajax: A New Approach to Web Applications". http://www.adaptivepath.com/ideas/essays/archives/000385.php.

[10] Cascading Style Sheets. http://www.w3.org/Style/CSS/Overview.en.html.

[11] EASTWeb project. http://globalmonitoring.sdstate.edu/projects/eastweb/.