FWA – A Framework for Developing Web-Atlas Applications

Mark Rop and Yi Liu
Dept of Electrical Engineering and Computer Science
South Dakota State University
Brookings, USA
mkrop@jacks.sdstate.edu , yi.liu@sdstate.edu

Michael C. Wimberly
GISc Center of Excellence
South Dakota State University
Brookings, USA
michael.wimberly@sdstate.edu

***Abstract*—The increasing availability of geospatial datasets, software, and analysis tools has resulted in the expanded use of maps in the field of public health. Web-based mapping has the potential to greatly enhance the accessibility of these maps and other public health data to health practitioners and the general public. Framework of Web Atlas (FWA) is a novel framework for building web atlas applications that support organizing, managing, and providing access to large amount of map dataset to the public health. This paper presents the architectural design of the FWA, and gives a case study to show how to use it.**

Keywords-software framework; web atlas; software architecture

I. INTRODUCTION

The increasing availability of geospatial datasets, software, and analysis tools has resulted in the expanded use of maps in the field of public health. For example, data from earth-observing satellites is increasingly used to map the environmental risk factors associated with both chronic and infectious diseases [13], and to provide early warning of future disease outbreaks [3]. Widespread access to the Internet has fostered the development of novel techniques for disseminating and visualizing the resulting digital map products. These approaches range from simple images of static maps, to dynamic web-enabled GIS applications, to geo-browser applications such as Google Earth [4]. Potential use cases for these emerging technologies include visualization of large volumes of geographic data, interoperability among multiple applications, and modeling of environmental phenomena [5]. Web-based mapping has the potential to greatly enhance the accessibility of disease maps and other public health data to health practitioners and the general public.

Considerable attention has been focused on the creation of specific web mapping applications for public health [8], but there has been less emphasis on developing broader systems for organizing, managing, and providing access to large amounts of data through a variety of web mapping applications.

From the user side there is a need for a well-organized framework that allows users to access spatial information in a variety of forms. The organization of the geospatial datasets will depend of the geospatial data formats available.

By observing existing web-atlas applications and discussing requirements with the customers, we found that the following three issues challenge the producer's side for developing and maintaining the web-atlas applications:

i. The web contents have a dynamic nature. The maps might be updated frequently on a monthly basis, a weekly basis or even in a daily basis.

ii. Producers who need to make frequent changes to the web contents might not have web design or programming skills.

iii. The change to the organization structure of the maps might involve tremendous effort to rearrange all the web pages that disseminate the maps.

Thus, from the producer side there is a need for an easy-to-use application that will allow health organizations to update and manage large collections of electronic map products. This will involve adding new contents, editing existing contents, or deleting old contents. The producers or the web authors will be responsible for these operations.

To respond to the needs from both web-atlas users and producers, we have developed FWA – a novel framework that facilitates the development of the web atlas applications to be used to disseminate geospatial datasets and authoring of the web contents.

The paper focuses on the construction of FWA. Section 2 illustrates the architectural design of the FWA. Section 3 sketches the implementation of FWA and section 4 uses two case studies to illustrate how to use FWA in developing real world web-atlas. Section 5 concludes the results.

II. ARCHITECTURAL DESIGN

Based on the requirements for a couple of different web-atlas proposed by map producers, we make the following design decisions, which are true for all the FWA applications [7]. These design decisions will be the basis of the architectural design.

A. *Design decisions*

i. Each map is organized into a collection/category/record hierarchy. For example, collections could be defined as

types of diseases (e.g.,West Nile virus or malaria). Categories could include static maps (e.g., pdf and image formats), and dynamic web-enabled GIS applications (e.g., KML/KMZ formats). A map for West Nile disease in the pdf format will be presented in *West Nile* (collection)/*Static maps* (category).

ii. A map and its associated information can be added/deleted/modified.

iii. The producer's view and the end user's view are separated as figure 1 shows.

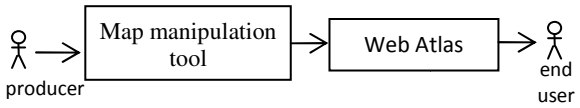

Figure 1. Separation of producer's view and end user's view

iv. The producer should be able to define and modify the categories and collections.

v. When the collection or category definitions are changed, the map records should be shipped to the updated collection/category upon the producer's request.

vi. The update to collections, categories, or maps should cause the changes to the web-based atlas.

B. *Design of FWA arichtecture*

A software framework is "a generic application that allows the creation of different applications from an application (sub)domain." [10] In object-oriented context, framework is "reusable design expressed as a set of abstract classes and the way their instances collaborate". [6] The difficulty part of framework design is the identification of the abstractions that can be tailored to specific applications within the framework domain.

In Schmid's [11] approach, within a certain framework domain, the variable and application specific aspects are abstracted as *hot spots* and the common aspects as *frozen spots.*

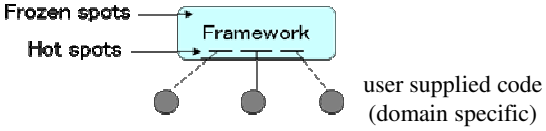

Figure 2. Hot spots and Frozen Spots

1) *Frozen spots system*

In the design of FWA, the common aspects are identified to construct the *frozen spots system.*

Based on the design decision (i), we use the hierarchy of collection/category/record to manipulate the maps. A collection can contain 0 to more categories. A category can contain 0 to more records, which holds the maps and their descriptive information.

Based on the design decision (iii), FWA is decomposed into *map manipulation* and *web atlas* for the first cut of the architecture. Further refinement is done on the map manipulation part by decomposing it into *Map storage*, *Map control* and *Web generation*. *Map Control* is designed for design decisions (ii) and (iv). It takes the user's inputs to add/modify collections, add/modify/delete categories, and add/modify/delete records. *Map Storage* stores the maps with the map hierarchy. *Map Control* feeds the data (maps) to *Map storage*. *Webpage generation* is designed for design decisions (v) and (vi). It retrieves the data from the *Map Storage* to make updated web pages for the web-based atlas. Figure 2 shows the top level FWA architecture with frozen spots added it.

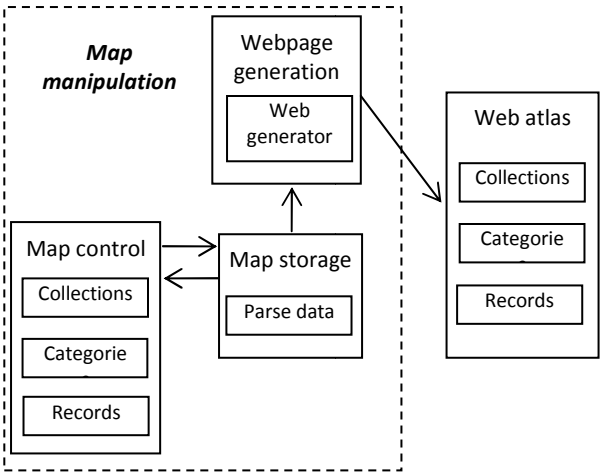

Figure 3. Top level FWA architecture with frozen spots

2) *Hot spots system*

Although the web atlas applications in the domain we focus on can share the modules shown in Figure 3, a web atlas application can also have unique features.

A web atlas application can have its own way of styling the web pages for hosting the web atlas, its individual location to store the maps in the *Map storage*, and its preference of the *Map control* user interface.

The variable aspects on webpage location and styling, map storage location, and map control user interface are addressed for constructing the *hot spots system.*

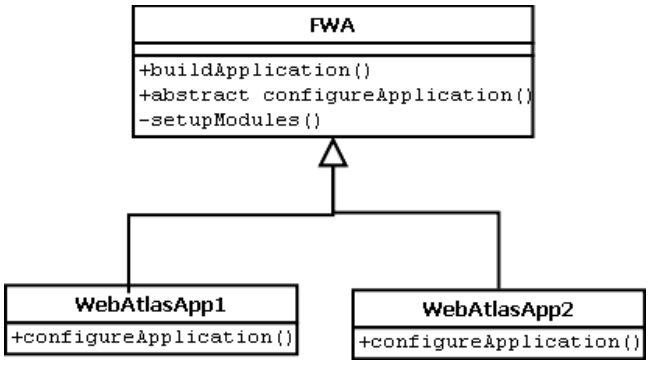

Figure 4. FWA's hot spots system

Template method pattern [2] is used in abstracting the three variable aspects. The *Abstract Template class* declares abstract primitive methods *configureApplication()*, which will be implemented by each concrete web atlas application to achieve its specificity. The template method *buildApplication()* invokes

the abstract method and set up the locations of *Map Controller*, *Map Storage*, and *Web Generator* and ship all the files with these three modules in frozen spots system to the concrete application.

3) FWA architecture

The FWA architecture (shown in Figure 5) is completed by connecting the frozen spots system and hot spots system.

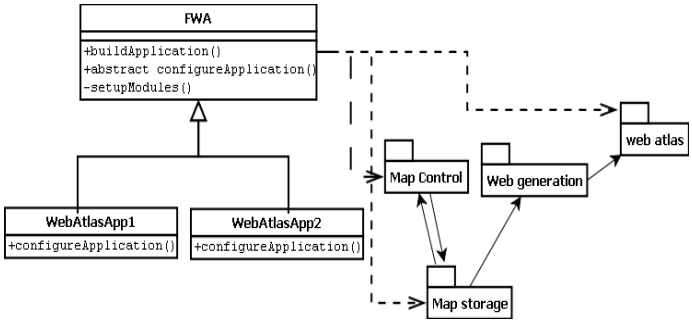

Figure 5. The architecture of FWA

III. FWA IMPLEMENTATION

The FWA has been developed mainly in PHP and AJAX. Three web atlases have been extended from FWA and currently are in use.

A. Implementation of FWA frozen spots system

1) Map storage

The maps are stored in the collection/category/record hierarchy. As mentioned in the Introduction section, one of the challenges for developing and maintaining web-atlas applications is that the change to the organization structure of the maps might result in rearranging all the web pages that disseminate the maps. For example, consider a situation where originally the producer makes all records held in collection directly. Later, there is a need to add a layer of categories (e.g., category1 and category2) to better organize the records. To respond to this challenge, we used XML due to its flexibility in presenting the information and it portability for transmitting the data. In the second phase, we plan to add search functions in FWA, and the usage of XML will also be beneficial for supporting the approaches of semantic search.

XML schema is used to describe the structure, define allowable document content and validate the correctness of data in the FWA XML document.

With the collection/category/record structure, a collection (e.g, *West Nile Virus*) contains 0 or more categories (e.g., *Historical West Nile Virus Data* and *Forecast 2011*), and a record contains a record title, a record description, a thumbnail of the record, and multiple maps (e.g., *Long-Term Hot Spots* record with 1 image map and 1 kmz file for download). Figure 6 shows a segment of the xml document.

```
<?xml version="1.0" encoding="UTF-8"?>
<EASTWeb
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FWA.xsd" >
  …
  <collection id="West Nile Virus">
    <category id="Default">
    <category id="Historical WNV Data">
      <record>
        <title>Long-Term Hot Spots, South Dakota</title>
        <description>Local indicators of spatial association (LISA) ...
        <thumbnail>thumb.png</thumbnail>
        <files filedescription=" Long-term hot spots, 2002-2007
              (.png)">lthp.png</files>
        <files filedescription="Long-term hot spots, 2002-2007
              (Google Maps)">lthp.kmz</files>
      </record>
    </category>
  </collection>
  …
</ EASTWeb>
```

Figure 6. Portion of the XML document for data storage

The *ReadData* class is developed to wrap the functions for accessing the XML map document. *get* and *set* methods are provided for accessing collection, category and record.

```
class ReadData
{
      /* class variables */
      private $_xml;
      public function ReadData($xml)
      {  $this->_xml=$xml;  }

      private function getCollections()
      {  $collection= Array();
         //Parse the xml and return all the collections
         return $collection;
      }
      private function getCategories($collection)
      {  $categories=Array();
         //Parse the XML and return the categories in the collection
         //Specified.
         return $categories;
      }
      private function getRecords($collection, $category)
      {
         $records= Array();
          if(isset($collection))
          {   //if collection is defined, get the records in the collections
              ……
          }
         else if(isset($category))
         {   //get records in the category specified
              …….
         }
         else
         {  //log the error
         }
          return $records;
      }
}
```

Figure 7. Accessing the map XML document

2) Map control

Map control is where a producer manipulates the maps through the collection/category/record hierarchy. Client side scripting and server side scripting are both involved. To

shorten the response time and avoid frequently reloading the web pages during interaction, AJAX is used to exchange the data between client and server asynchronously. Server side scripting is developed in PHP.

Collection, Category and Record have add, delete and edit functions. Interface *ItfControl* is introduced to declare these three functions to allow polymorphic access to Collection, Category and Record. Figure 8 shows the UML diagram of this design and Figure 9 shows the code of ItfControl and Figure 10 shows the partial code of how Collection implements the ItfControl interface.

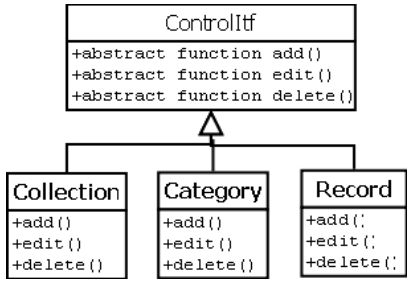

Figure 8. Collectio, Category, Record and the interface

```
abstract class ControlItf
{
  abstract public function add() { //define add operation}
  abstract public function edit(){//base edit function}
  abstract public function delete(){//base delete function }
}
```

Figure 9. .Map Control interface

```
class Collection extends ControlItf
{
   /* class variables */
   private $_collection;
   private $_xml;
   public function Collection($collection)
   { $this->collection=collection;          }
   public function add()
   {//override base class
    //implements the add functionality
   }
   public function edit()
   {//override base class edit
    //implements edit functionality
   }
   public function delete()
   {override base class delete
   //implements delete functionality
    }
}
```

Figure 10. Partial code of Collection class

3) Webpage generation

This module dynamically generates the web pages when a new collection or a new category is created. For example, the producer creates a collection named *West Nile Virus* and a category named *Historical West Nile Virus Data* under that, a web page for West Nile Virus and a web page for Historical West Nile Virus Data will be created on the end user's view and the new collection and new category will be added to the website navigation menu if it has one.

```
class WebGeneration
{
   public static function createCollection($collectionName)
   {
      //locate application collection template
      //Open and read file
      //dynamically write the destination collection file
   }
   public static function createCategory($categoryName)
   {
        //locate application specific category template
        //Open and read file
        //dynamically write the destination category file
   }
}
```

Figure 11. Web Generation

B. Implemenation of FWA hot spots system

The hot spots system contains an abstract class FWA (shown in Figure 12). This abstract class will be implemented in each concrete application.

```
abstract class FWA
{    protected $controllerPath;
     protected $storagePath;
     protected $generatorPath;
     protected $appname;

     public buildApplication()
     {          configApplication();
                // setup Map Controller
                setupModules($controllerPath, filePath::controller);
                // setup Map Storage
                setupModules($storgePath, filePath::storage);
                // setup Web Generator
                setupModules($generatorPath, filePath::generator);
      }

     abstract function configApplication();

     private function setupModules(&$appPath, $FWAPath)
     {       //path to unzipped FWA module
             $path = $FWAPath;
             $appname = str_replace(" ","",$appname);
             //path to destination application module
             $appPath="/".$appname."/".$path;
             if(!is_dir($appPath))
             {       mkdir($appPath);          }
             //bulding the module
             exec("cp $path $appPath");
     }
}
```

Figure 12. FWA abstract class

IV. USE OF FWA

It is very easy to use FWA to build a web atlas in the similar domain. All the developer needs to do is to (1) construct a website with the user's preferred layout and color

schema, and (2) implement the FWA abstract class. (1) could be done with any web editor, such as DreamWeaver. The following example presents the template code for implementing FWA for a concrete application.

EASTWeb [1] is a collaborative project involving scientists from South Dakota State University (SDSU) and the USGS Center for Earth Resources Observation and Science (EROS), along with partners from government agencies, and non-governmental organizations. Their research focuses on the application of geospatial technologies for mapping, risk analysis, and ecological forecasting of infectious diseases. The main purpose of EASTWeb is to provide public access to the various products that will be developed by the project, including geospatial datasets, maps, and forecasts.

The developer constructs a website for the end users to provide the information on background, updates on research activities, relevant web resources, and a spot for hosting the web atlas part. The website is shown in figure 13.

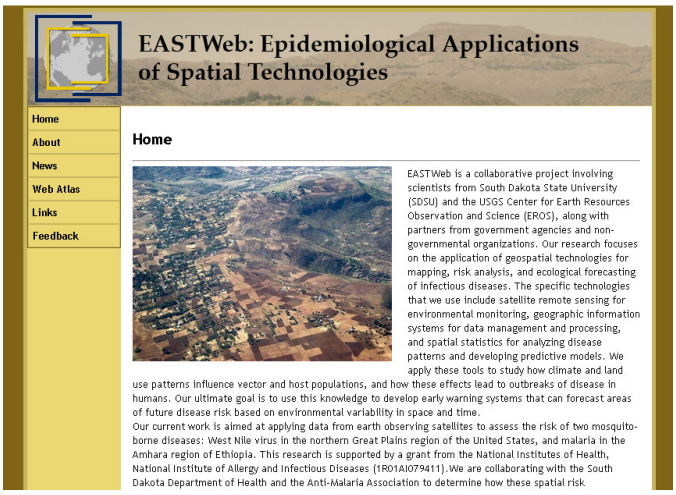

Figure 13. EASTWeb end user's view

The developer needs to allocate the locations (would be folders on the server) for holding the Map Controller and Data Storage. Once it is set, the developer implements the methods *configreApplication()* by given the application name to make the specific map manipulation tool for the producer and link the map data to the web atlas part of the website for the end users. Figure 14 shows the implementation. Figure 15 shows the producer's view of map manipulation tool and Figure 16 shows the end user's view of the web atlas.

```
class EASTWebApplication extends FWA
{
    public funtion EASTWebApplication();

    function configApplication()
    {       parent::appname= "EASTWeb";          }
}
```

Figure 14. Extending FWA for constructing EASTWeb

Figure 15. EASTWeb – producer's view

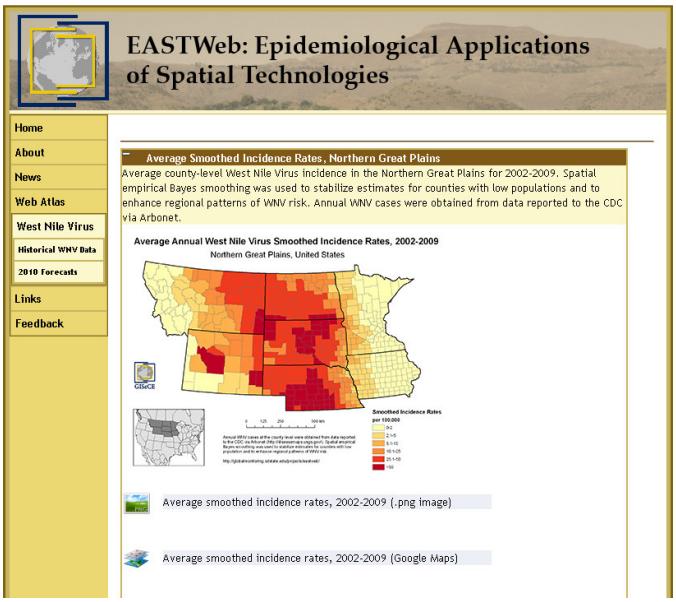

Figure 16. EASTWeb – end user's view

V. CONCLUSION

FWA is a framework that is designed and developed to support organizing, managing, and providing access to large amounts of geospatial data in the field of public health. The FWA can be easily customized to specific applications by being given category or collection definitions and the template of the web page layout.

FWA effectively facilitate the construction and maintenance of web-based atlas. Although the initial purpose of FWA is for the field of public health, the design of FWA architecture makes it flexible to be extended into building applications in other similar domains. We have already developed two more web atlases by using FWA in addition to the EASTWeb site described in section IV for forecasting

infectious disease. One of them is for providing geospatial information to support environmentally sustainable biomass production in the North Central Sun Grant Region [12]; and the other one [9] is for disseminating the geospatial data products resulted from the research on testing multiple working hypotheses about the environmental drivers of obesity.

In conclusion, we have found that the FWA greatly facilitates the dissemination of digital maps by allowing developers to quickly and easily implement a web atlas within a website, and by providing users with access to large amounts of geospatial information in a variety of formats. Given the experience of extending three web atlas applications from FWA, we believe FWA can be successfully and effectively used in constructing web atlases for multiple projects encompassing a variety of thematic areas.

ACKNOWLEDGMENT

This work is supported by NIH grant 1R01AI079411-01 "*An Integrated System for the Epidemiological Application of Earth Observation Technologies*".

REFERENCES

[1] EASTWeb project website.
http://globalmonitoring.sdstate.edu/projects/eastweb/. Last access date: 5/17/2011.

[2] Gamma, E., Helm, R., Johnson, R.,Vlissides J, Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, 1995.

[3] Grover-Kopec, E., M. Kawano, R. W. Klaver, B. Blumenthal, P. Ceccato, and S. J. Connor. An online operational rainfall-monitoring resource for epidemic malaria early warning systems in Africa. Malaria Journal 4:6, 2005.

[4] Mills, J. W., and A. Curtis. Geospatial approaches for disease risk communication in marginalized communities. Progress in Community Health Partnerships 2: 61-72, 2008.

[5] Goodchild, M. F. The use cases of digital earth. International Journal of Digital Earth 1:31-42, 2008.

[6] Johnson, R. Frameworks Home Page.
http://st-www.cs.uiuc.edu/users/johnson/frameworks.html. Last accessed: 5/17/ 2011.

[7] Liu, Y., Rop, M., and Wimberly, M. C. On the construction of framework of web-atlas (FWA). Proceedings of ACM South East, 2010.

[8] MacEachren, A. M., Crawford, S., Akella, M., and G. Lengerich. Design and implementation of a model, web-based GIS-Enabled cancer atlas. The Cartographic Journal 45: 246-260, 2008.

[9] Obesity web site.
http://globalmonitoring.sdstate.edu/projects/obesity/. Last accessed date: 5/17/2011.

[10] Schmid, Hans Albrecht. Systematic framework design by generalization. Communications of the ACM. 1997.

[11] Schmid, H. A. Framework design by systematic generalization, In Fayad, M. E. , Schmidt, D. C., and Johnson, R. E., editors, Building Application Frameworks, pp. 353-378, Wiley, 1999.

[12] Sun Grant website:
http://globalmonitoring.sdstate.edu/projects/sungrant. Last accessed date: 5/17/2011.

[13] Wimberly, M. C., A. B. Baer, and M. J. Yabsley. Enhanced spatial models for predicting the geographic distributions of tick-borne pathogens. International Journal of Health Geographics 7:15, 2008.