WebGD: A Framework for Web-Based GIS/Database Applications

Toshimi Minoura and Hiroshi Tashiro

School of Electrical Engineering and Computer Science
Oregon State University, Corvallis, Oregon, 97331-4602, minoura@cs.orst.edu

Abstract

A typical Internet map-server application allows only *retrieva*l of maps and map-related data. We have been developing *web-based GIS/database* (WebGD) applications that allow users to *insert*, *query*, *update*, and *delete geographical features* and the data associated with them from standard Web browsers. The code shared by these applications is organized as the WebGD framework. We can create the map interface of a new Web-based GIS/database application by providing *configuration files*. We have also built the WebGD application generator (WebGD-Gen) that *automatically* produces from the database schema a set of Web scripts that interact with the map interface and the database. This application generator greatly simplifies the process of creating a complex Web-based GIS/database application and significantly reduces the development time and maintenance cost. The WebGD framework and WebGD-GEN currently support such advanced features as *tight integration* of a Web-based map interface with a database, *automatic selection* of the *spatial reference* and *map layers* for the current region, and *automatic generation* of Web forms. The forms generated can be used to *insert*, *search*, *update*, and *delete* geographical features and the data associated with them.

1. Introduction

The Internet is emerging as the major venue for sharing information. Dynamic web-based mapping applications provide customized on-line geographical information systems (GISs), which allow users without desktop-based GIS software to perform *spatial queries* and produce maps with standard Web browsers.

However, a typical web-based GIS application allows only the retrieval of maps and map-related data. A web server provides information to the client, but the client cannot feeds information back to the server (Kingston, 1998). This *unidirectional* flow of information is a major problem with a current typical map-server application. Furthermore, map information and data stored in a database are not integrated well. The map coverage is limited to *one* region. Creating an interactive web application with a map interface is time-consuming. Commercial map servers and geographical database management systems are expensive. Analysis tools implemented are limited. This situation hinders use of Web-based GIS applications in data gathering, analysis, and decision-making.

We have been developing a set of tools that significantly reduces the cost of application development (Wangmutitakul et al., 2003, Wuttiwat et al., 2003; Sano et al., 2003, Wangmutitakul et al., 2004). The code shared by interactive Internet GIS applications is organized as the Web-based GIS/database (WebGD) framework. Most of the complex workings for delivering GIS functions over the Web are included in this framework. With the WebGD framework, we can create a map interface through which users can *insert*, *query*, and *delete* geographical features and the data associated with them only by providing *configurations files*.

An important feature of a WebGD application is that it tightly integrates spatial data and associated tabular data, enabling analyses involving location-based data (Sharma 2003, http://yukon.een.orst.edu/ms_apps/w6grin_cs549/analysis/analyze_collect.htm). These functions are available to users located at different geographical locations for economical and timely data management.

We developed several years ago a *web-form generator* that automatically generates code for traditional web-forms from the *schema* of a relational database (Eum et al., 2003). This functionality was recently extended as the WebGD application generator (WebGD-Gen). The new application generator can produce the code of an entire WebGD application from the schema of a relational database and the information on *geometry columns*. The code generated for the map interface can *insert*, *query*, and *delete* geographical features (i.e., points, line-strings, and polygons). When a relational schema and the GIS data for the map layers are available, a *non-customized* application can be quickly assembled with the WebGD framework and WebGD-Gen.

The automatic generation of the map interface and Web forms makes possible *incremental* and *iterative* development of complex web-based GIS applications. When a map layer is added or modified, we only need to create or update the configuration file for that layer and then regenerate the Web forms for that layer. When a database schema in modified, we can regenerate the entire set of Web scripts in several minutes. Therefore, even with an incomplete set of map layers and a database schema, we can generate a working prototype for an initial review. Furthermore, we can fix most of the software bugs by modifying only the shared code in the WebGD framework or WebGD-Gen and then by regenerating Web scripts for each application.

WebGD applications use the following *open-source* software components. PostgreSQL, an *object-relational* database, and PostGIS together manage geospatial data. PostGIS is an extension of PostgreSQL for GIS applications (Ramsey). MapServer (Univ. of Minnesota) generates maps to be displayed on a web browser by using geospatial data provided by PostGIS. Web pages, including the one that displays the maps, are generated by server-side scripts written in PHP. The PHP Mapscript module interacts with MapServe (McKenna, DM Solutions). When a request to insert or delete a map feature is received by a PHP script, the script directly accesses the PostgreSQL database, using the PostGIS extension. An application developed runs on a PC without any licensed software.

We explain the features supported by the WebGD framework in Section 2 and the capabilities of the WebGD-GEN application generator in Section 3. In Section 4, we describe a brief history of the development of the WebGD framework and WebGD-GEN. Section 5 concludes this paper.

2. WebGD Applications

The Web interface of one of the WebGD applications, Natural Heritage Information System (NHIS) for North Carolina, is shown in Figure 1. This application provides a map interface for a copy of the

Biotics 4.0 database maintained by the North Carolina Natural Heritage Program. Biotics 4.0 is a desktop GIS application built on the database developed by NatureServe. The key elements in this database are *element occurrences* (EOs), which are *areas* of land and/or water in which species are or were present (NatureServe, 2002). EO records have both *spatial* and *tabular* data, and the database contain approximately 700 relational tables (Fogelsong, 2002). The Biotics Mapper implemented with ArcView by NatureServe provides a map interface that allows EO representations and associated data to be created, updated, and deleted (NatureServe, 2003). In our implementation, we can perform these operations with standard Web browsers. Also, Web forms, approximately 3500 in total, are provided for all the tables in the database.

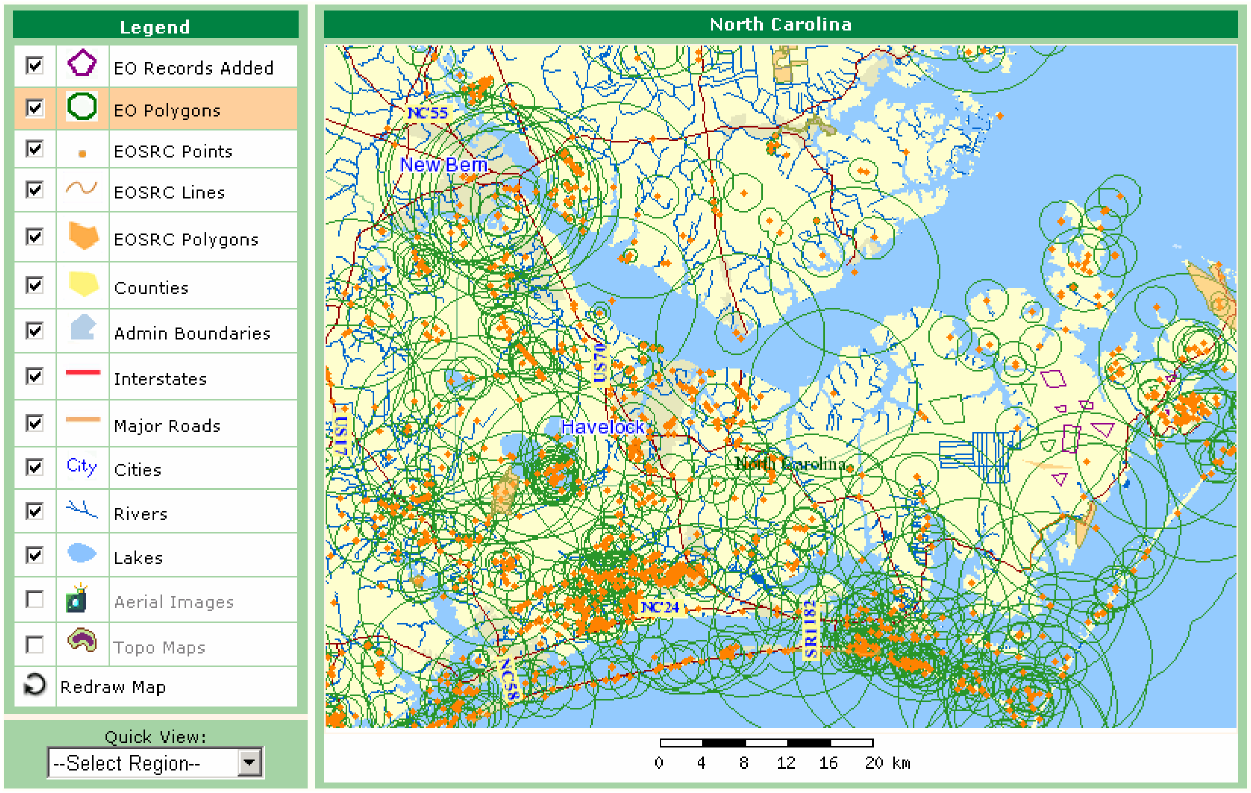


Figure 1: Interface of the NHIS Application for North Carolina.

The NHIS application enables *bi-directional* movement of *geospatial data* as well as ordinaly data. Scientists and others with proper authentication can *insert*, *query*, and *delete* geographical features such as EO polygons, lines, and points, as well as the data associated with them. Queries can be executed by spatially *selecting an area* on the map or by using a traditional web form. In addition, one-meter resolution *digital orthographic quadrangles* (DOQ), or aerial images, are included as a layer. When DOQ images are combined with other map layers such as highways, county boundaries, streams, and streets, locations can be easily pinpointed by taking advantage of features between map layers (Wuttiwat et al., 2003).

The major operations supported by the map interface of a WebGD application are as follows.

1. To retrieve information on the geographical features in the area of interest, the user can zoom in/out to that area by using the map navigation tools. If the user zoom-in enough, one-meter resolution aerial photos are displayed. The user can also go to a new area by selecting an entry in the **Quick View** menu.

2. To get information about a geographical feature, the user can select a layer in the legend and **Information** in the function menu, and then click the boundary of the feature.

3. Function **Insert** allows a geographical feature to be added with mouse clicks on the map. **Done** need be pressed after all points are entered.

4. Function **Search by Area** allows the user to retrieve the list of features that are within a *bounding box* specified on the map and that satisfy a search condition. The features that satisfy the search condition are *highlighted* on the map. Furthermore, the user can select features in the list by marking the checkboxes associated with them. Then, if the map is refreshed, the selected features are highlighted.

5. The data administration interface can be activated by clicking on the **Database** entry in the menu bar below the banner. A tree icon can be clicked to display a *treeview* for browsing. The treeview for **Higher Taxonomy** is the major one. To access the data of this application, a user must login with a password as some data on endangered species are confidential. Several WebGD applications produced with the WebGD framework and WebGD-Gen can be accessed at the following URLs:

```
http://yukon.een.orst.edu/ms_apps/nhis_n_carolina/gmap75_main.phtml
http://yukon.een.orst.edu/ms_apps/digir/gmap75_main.phtml
http://yukon.een.orst.edu/ms_apps/nrcs/gmap75_main.phtml
http://yukon.een.orst.edu/ms_apps/w6grin_cs549/gmap75_main.phtml
http://yukon.een.orst.edu/ms_apps/oregon_viewer/gmap75_main.phtml
```

The first application is Natural Heritage Information System. Although this application can cover the whole USA or the world, the data are currently available only for North Carolina. The second application provides a map interface for a local database containing DiGIR records harvested from DiGIR providers, DiGIR (Distributed Generic Information Retrieval) is an XML-based communication protocol for a *federation* of databases managed by natural history museums**.** The third application provides a map interface for an application that keeps track of conservation practices on land parcels.. The fourth one is a Web-based mapping application for a plant germplasm collection maintained at Western Regional Plant Introduction Station (USDA-ARS). The fifth one allows the soil information at the location where a mouse click occurs on the map interface to be retrieved.

One salient feature of the current WebGD framework is *dynamic switching of spatial references*. Typically, different geographic regions and localities have preferred *map projections* in order to avoid distortions in the maps created (Dana). The framework allows the whole world to be covered with multiple-levels of maps, e.g., the world map, continent maps, and regional maps. The map interface then automatically selects the most suitable projection for the region whose portion is displayed. For example, the world can use the *geographical coordinate system*, the United States the

Albers equal-area projection, and Oregon the *Lambert conformal conic projection*. Thus, spatial analysis can be performed with the most appropriate projection for a particular area. The *dynamic switching* of the *spatial reference*, the *map file*, the *legend*, and the *quick view menu* supported by the current WebGD framework allows any part of the world to be covered with its own scale and spatial reference, including regions with one-meter resolution aerial images. This is a very important feature, especially now that the cost of storing aerial images for the entire US has dropped to affordable levels (10 tera-bytes needed to store aerial images for the entire US now cost around $10,000). Furthermore, many states are putting aerial images in the public domain.

3. WebGD-GEN Application Generator

Several tools have been developed to augment the WebGD framework and simplify application development. The WebGD Web-site generator (WebGD-Gen) can create an entire WebGD application, including a web-based mapping interface. WebGD-Gen *automatically* generates a consistent set of Web scripts from *configuration files*, which are again automatically generated from a relational database schema. Since form generation is automatic, the cost of application development is greatly reduced. For a database such as Biotics that contains approximately 700 tables, programming all the required 3,500 (700 x 5) forms manually can be very costly, even infeasible.

WebGD-Gen is implemented as a collection of *templates*. Each template, combined with a corresponding *configuration file*, generates one of the following six types of Web scripts: *search, select, edit, information, action*, and *treeview* scripts. Templates and configuration files are written in PHP. The Web scripts generated by them are also in PHP. The generated scripts are executed on a Web sever by a PHP interpreter. Each script, except for an action script, creates a Web form that is displayed on a client computer by a Web browser. Figure 2 illustrates the interactions among the Web scripts and forms.

Furthermore, WebGD-Gen can automatically generate the statements for inserting, searching, and deleting *geographical features* if the following lines, e.g., are added to a configuration file:

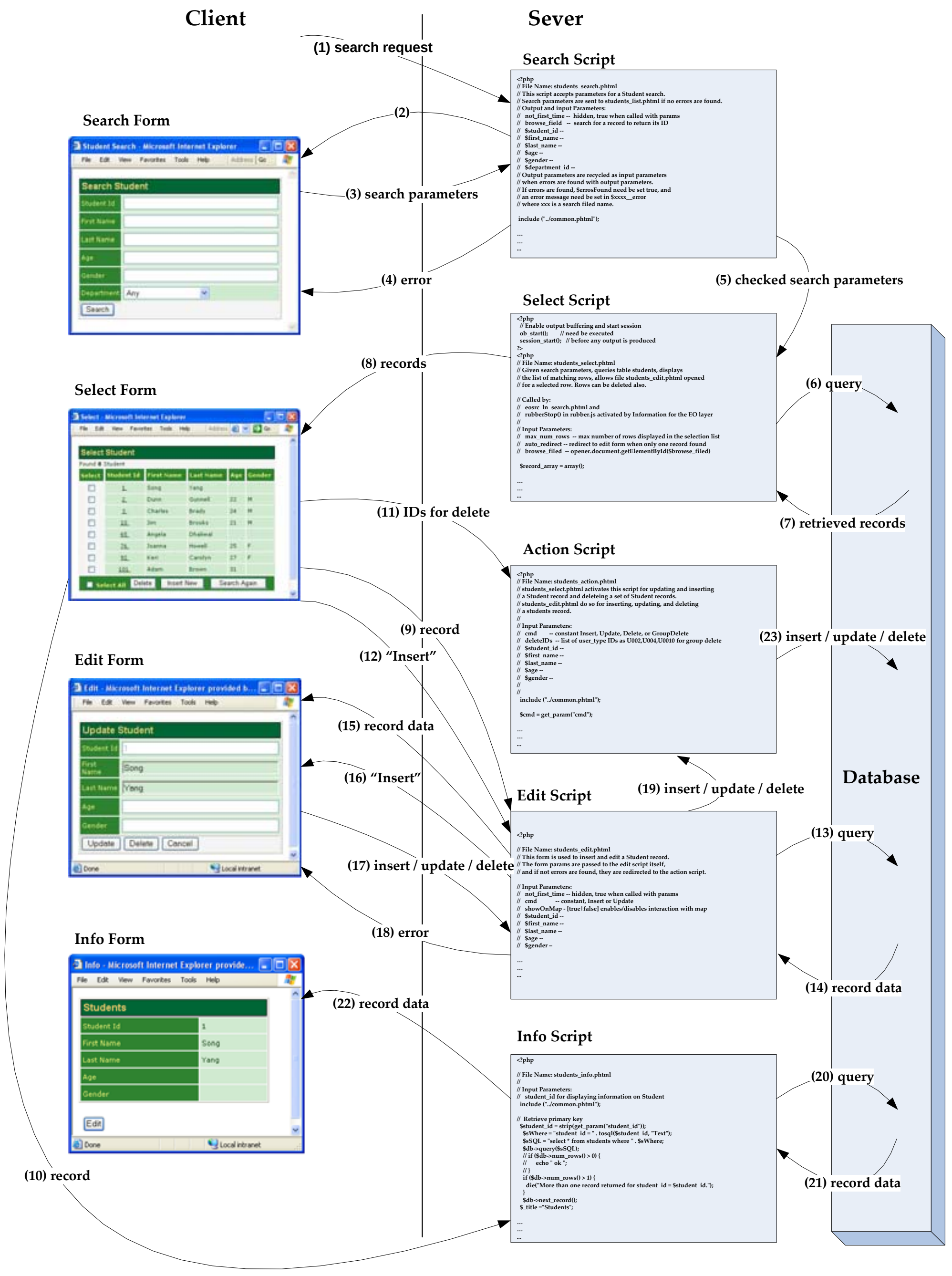
```
$web_gd = ‘MULTIPOLYGON’;        // type of geographical features
$layer_name = ’grp_eo_py’;       // layer group in legend
$geometry_column = ‘the_geom’; // geometry column containing shapes
$gid_column = ‘gid’;             // geographical feature IDs
$db_table_srid = 32119;          // epsg spatial reference
```

The forms generated for geographical features can perform the following additional functions compared to those for ordinary database tables

1. A search form can be activated from a map interface. In this case, the extent of a search box specified on the map is passed as additional search parameters.

2. A select form includes additional JavaScript code for highlighting geographical features retrieved or selected by the user.

3. An edit form can insert a record for a geographical feature, after transforming the coordinate values from the spatial reference used by the current map interface to the one used by the geometry column for the record.



Search Script:

```
<?php
// File Name: students_search.phtml
// This script accepts parameters for a Student search.
// Search parameters are sent to students_list.phtml if no errors are found.
// Output and input Parameters:
//   not_first_time --  hidden, true when called with params
//   browse_field   --  search for a record to return its ID
//   $student_id --
//   $first_name --
//   $last_name --
//   $age --
//   $gender --
//   $department_id --
// Output parameters are recycled as input parameters
// when errors are found with output parameters.
// If errors are found, $errosFound need be set true, and
// an error message need be set in $xxxx__error
// where xxx is a search filed name.

 include ("../common.phtml");

…
…
...
```

Select Script:

```
<?php
  // Enable output buffering and start session
  ob_start();       // need be executed
  session_start();   // before any output is produced
?>
<?php
// File Name: students_select.phtml
// Given search parameters, queries table students, displays
// the list of matching rows, allows file students_edit.phtml opened
// for a selected row. Rows can be deleted also.

// Called by:
//   eosrc_ln_search.phtml and
//   rubberStop() in rubber.js activated by Information for the EO layer
//
// Input Parameters:
//   max_num_rows  -- max number of rows displayed in the selection list
//   auto_redirect -- redirect to edit form when only one record found
//   browse_filed  -- opener.document.getElementById($browse_filed)

  $record_array = array();

…
…
...
```

Select Form:

Found 8 Student

Select	Student Id	First Name	Last Name	Age	Gender
☐	1.	Song	Yang		
☐	2.	Dunn	Gunnell	22	M
☐	3.	Charles	Brady	24	M
☐	10.	Jim	Brooks	21	M
☐	65.	Angela	Dhaliwal		
☐	76.	Joanna	Howell	25	F
☐	91.	Kari	Carolyn	27	F
☐	101.	Adam	Brown	31	

Action Script:

```
<?php
// File Name: students_action.phtml
// students_select.phtml activates this script for updating and inserting
// a Student record and deleteing a set of Student records.
// students_edit.phtml do so for inserting, updating, and deleting
// a students record.
//
// Input Parameters:
//   cmd        -- constant Insert, Update, Delete, or GroupDelete
//   deleteIDs  -- list of user_type IDs as U002,U004,U0010 for group delete
//   $student_id --
//   $first_name --
//   $last_name --
//   $age --
//   $gender --
//
//
  include ("../common.phtml");

  $cmd = get_param("cmd");

…
…
...
```

Edit Script:

```
<?php

// File Name: students_edit.phtml
// This form is used to insert and edit a Student record.
// The form params are passed to the edit script itself,
// and if not errors are found, they are redirected to the action script.

// Input Parameters:
//   not_first_time -- hidden, true when called with params
//   cmd            -- constant, Insert or Update
//   showOnMap - [true|false] enables/disables interaction with map
//   $student_id --
//   $first_name --
//   $last_name --
//   $age --
//   $gender –

…
…
...
```

Info Form:

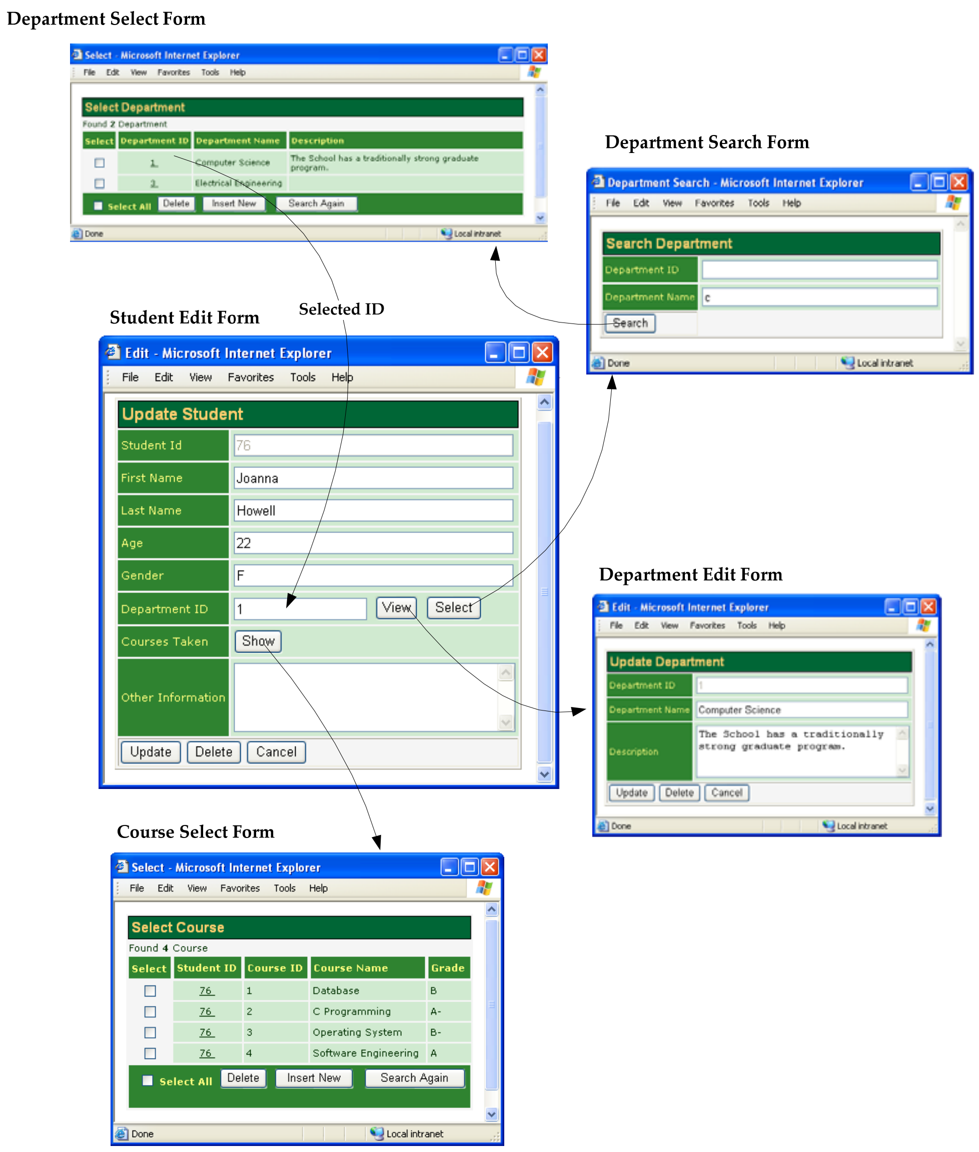
Students	
Student Id	1
First Name	Song
Last Name	Yang
Age	
Gender	

Info Script:

```
<?php

// File Name: students_info.phtml
//
// Input Parameters:
//   student_id for displaying information on Student
  include ("../common.phtml");

//  Retrieve primary key
  $student_id = strip(get_param("student_id"));
    $sWhere = "student_id = " . tosql($student_id, "Text");
    $sSQL = "select * from students where " . $sWhere;
    $db->query($sSQL);
    // if ($db->num_rows() > 0) {
    //      echo " ok ";
    // }
    if ($db->num_rows() > 1) {
      die("More than one record returned for student_id = $student_id.");
    }
    $db->next_record();
  $_title ="Students";

…
…
...
```

Figure 2: Interactions among Web Scripts and Forms.

The forms related are automatically linked each other. Figure 3 shows, as an example, the *edit* form for a Student table. From this edit form, the user can open the forms for the department and courses related to the student. The information needed to create the links are extracted from the *primary-key/foreign-key relationships* among the tables in the database.



Department Select Form
Department Search Form
Student Edit Form
Selected ID
Department Edit Form
Course Select Form
Select - Microsoft Internet Explorer
Department Search - Microsoft Internet Explorer
Edit - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Done
Local intranet

Select Department

Found 2 Department

Select	Department ID	Department Name	Description
☐	1.	Computer Science	The School has a traditionally strong graduate program.
☐	2.	Electrical Engineering	

☐ Select All | Delete | Insert New | Search Again

Search Department

Department ID	
Department Name	c

Search

Update Student

Student Id	76
First Name	Joanna
Last Name	Howell
Age	22
Gender	F
Department ID	1 View Select
Courses Taken	Show
Other Information	

Update | Delete | Cancel

Update Department

Department ID	1
Department Name	Computer Science
Description	The School has a traditionally strong graduate program.

Update | Delete | Cancel

Select Course

Found 4 Course

Select	Student ID	Course ID	Course Name	Grade
☐	76	1	Database	B
☐	76	2	C Programming	A-
☐	76	3	Operating System	B-
☐	76	4	Software Engineering	A

☐ Select All | Delete | Insert New | Search Again

Figure 3: Student Edit Form.

In order to generate the Student edit form shown in Figure 3, the following entry is provided in the configuration file.

```
$edit_fields=array(
  array("column"=>"student_id", "label"=>"Student Id",
         "type"=>"numeric", "maxlen"=>"40", "size"=>"40"),
   array("column"=>"first_name", "label"=>"First Name",
        "type"=>"text", "maxlen"=>"40", "size"=>"40"),
   array("column"=>"last_name", "label"=>"Last Name",
        "type"=>"text", "maxlen"=>"40", "size"=>"40"),
   array("column"=>"age", "label"=>"Age",
        "type"=>"text", "maxlen"=>"40", "size"=>"40"),
   array("column"=>"gender", "label"=>"Gender",
        "type"=>"text", "maxlen"=>"40", "size"=>"40"),
   array("column"=>"department_id", "label"=>"Department ID",
        "type"=>"to_one", "linked_table"=>"departments",
        "maxlen"=>"40", "size"=>"40"),
   array("label"=>"Courses Taken", "type"=>"to_many",
        "linked_table"=>"courses", "maxlen"=>"40",
        "size"=>"40"),
   array("column"=>"other_info", "label"=>"Other Information",
         "type"=>"textarea", "rows"=>"4", "cols"=>"32"),
);
```

Each element in array `$edit_fields` represents a field in an edit form, with options `column`, `label`, `table`, `maxlen`, `size`, and `type`. Option `type` can be `numeric`, `text`, `time`, `date`, `email`, `phone`, `textarea`, `to_one`, or `to_many`:

- `textarea`: The input is a string displayed in a text area.

- `to_one`: A field of type `to_one` allows a user to view or modify the record related to the current record via a *one-to-one* or *many-to-one* relationship type. In our example, column `department_id` in table `students` is specified as `to_one`. When the `View` buttion is clicked, the edit form showing the record of the student's department is displayed. Furthermore, a user can search and select a new department record to be linked. Option `linked_table` designates the name of the table storing the associated record. Option `child_column` indicates the column for linking in the associated table. If this option is omitted, the name of the column for linking is identical to that of the foreign key column in the current record.

- `to_many`: The purpose of this type is to list the records related to the current record via a *one-to-many* or *many-to-many* relationship type. In our example, the field labeled `Courses Taken` is specified to be `to_many`. When the `Show` button is clicked, all the courses taken by the student are displayed in the select form for courses. This type needs options `linked_table`, `parent_column`, and `child_column`. Option `linked_table` and option `child_column` are similar to those for type `to_one`. However, the `child_column` can be omitted, when the name of the foreign key column of the associated table is identical to that of the primary key column of the current record. Option `parent_column` indicates the *foreign key* column in the current table. If this option is omitted, the foreign key column is the primary key column of the current table.

Furthermore, the information needed to display each map layer and to activate the web scripts when insert and search requests are issued with the map interface can be defined in a *map-layer* *configuration file*. The following definition is for the map layer of element occurrence data to be collected by field researchers within North Carolina:

```
'grp_eo_record_added' => array(
     'geom_type' => 'polygon',
     'table' => 'eo_record_added',
     'layer_selectable' => true,
     'gid_column' => 'eo_record_added_id',
     'geom_col' => 'the_geom',
     'legend_label' => 'EO Records Added',
     'search_script' =>
'forms/data_submission/eo_record_added_search.phtml',
     'select_script' =>
'forms/data_submission/eo_record_added_select.phtml',
     'edit_script' =>
'forms/data_submission/eo_record_added_edit.phtml',
     'normal_layer' => 'eo_record_added',
     'searched_layer' => 'eo_record_added_searched',
     'checked_layer' => 'eo_record_added_checked',
     'selected_layer' => 'eo_record_added_selected',
     'img_src' => 'images/eo_record_added.png',
     'img_width' => 21,
     'img_height' => 20,
     'onclick' => 'activate_layer("grp_eo_record_added")',
     'data_srid' => 32119
  ),
```

4. WebGD Development History

The WebGD framwork and WebGD-GEN were developed *incrementally* and *iteratively* during the last four years. We first implemented in 2000 an application that allowed point features to be inserted on a map by using ASP with ArcIMS and ArcSDE. In 2001, we re-implemented this application with ASP.NET, as ASP.NET provides Web controls, which are better building blocks for Web pages. Based on this application, the first version of WebGD framework was created in 2002 in order to support multiple applications (Wangmutitakul et al., 2004).

In early 2003, we re-implemented an application called Motels Oregon with MapServer, PostGIS, and PostgreSQL (Sano et al., 2003). This version on Linux was more reliable and faster than the old one, as well as being built with free software. While implementing the next MapServer application, which was a germplasm resource management system (GEM-GIS), we created the first version of WebGD framework for MapServer. This framework was then enhanced so that it can handle polygon features as well as point features.

The two major enhancements made to the WebGD framework in 2004 were *dynamic switching* of *spatial references* for different regions and *automatic generation* of Web forms that can be used to insert, query, and delete geographical features. Compared to an application that simply displays geographical features as points on a map, the current WebGD framework is roughly 20 times more complex in terms of the time we spent implementing the required features.

5. Conclusions and Future Work

We have developed the WebGD framework and the WebGD-GEN application generator for rapid development of Web-based GIS/database applications.

1. Geographical features, such as the locations of habitats of plants and animals, road-work sites, and waterlines, can be inserted, queried, and deleted from the map interface displayed on a standard Web browser.

2. An application can be created without any programming. The look and behavior of the map interface can be customized with configuration files, and Web scripts for data access can be automatically generated from the description of a database schema. This feature not only reduces significantly the development cost of a Web-based GIS application, but it also makes incremental and iterative development of the software easier.

3. Dynamic switching of spatial references allows an application to cover different regions with different map files, projections, map legends, and quick-view lists. This is an important feature needed for an application that covers the entire USA or the world.

The cost of running our applications is extremely low. We could put copies of such large databases as Biotics, SSURGO2 soil data, and a part of National Germplasm Resource Information System on a $800 PC. The software tools we use, such as the University of Minnesota MapServer, PostgreSQL DBMS, PostGIS, Apache, and PHP are all available for free. The GIS data used, such as those from USGS, TIGER/LINE, and Digital Chart of the World (DCW), are also in the public domain. Automatic code generation of a WebGD application will save a great deal of effort in the development of a spatial decision-support system. Although some manual customization is required, the time needed for customization can be lowered to weeks or months compared to the years required to build a *spatial decision-support system* from scratch.

References

- Dana, Peter H. *Map Projection Overview*, http://www.colorado.edu/geography/gcraft/notes/mapproj/mapproj_f.html.
- DM Solutions Goup Inc. *PHP MapScript*, http://www.maptools.org.
- Kingston, R. (October 1998). Web-based GIS for public participation decision making, In *Procs* of NCGIA PPGIS Meeting, Santa Barbara, California. Retrieved May 2003 from http://www.ncgia.ucsb.edu/varenius/ppgis/papers/kingston/kingston.html.
- Eum, D. and Minoura, T. (June 2003). Web-based database application generator. *IEICE Transactions on Information and Systems, Vol. E86-D, No. 6*.
- Fogelsong, C. (December 2002). Biotics 4.0 data model version 1.0. Retrieved January 5, 2004, from http://whiteoak.natureserve.org/hdms/HDMS-DataModel.shtml.

- McKenna, Jeff. *Mapserver PHP/MapScript Class Reference - Versions 3.6, 4.0 & 4.2*, DM Solutions Group Inc.
- NatureServe (February 2002). Element Occurrence Data Standard. Retrieved January 4, 2004, from http://whiteoak.natureserve.org/eodraft/all.pdf.
- NatureServe (December 2003). Biotics 4.0 Getting Started Guide. Retrieved January 5, 2004, from http://whiteoak.natureserve.org/hdms/biotics-learn-more.shtml (now obsolete).
- Sano, J., Wanalertlak, N., Maki, A., & Minoura, T. (July 2003). Benefits of web-based GIS/database applications. In *Prosc. of 2nd Annual Public Participation GIS Conference*, Portland, Oregon.
- USDA-ARS. Western Regional Plant Introduction Station, USDA - Agricultural Research Service, Pullman, Washington, http://www.ars-grin.gov/ars/PacWest/Pullman/.
- Ramsey, Paul. *PostGIS Manual*, Refractions Research Inc.
- University of Minnesota. *MapServer*, http://mapserver.gis.umn.edu, 2003.
- Sharma, A. (December 2003). Web-based analysis module for a germplasm collection. Master of Science report, School of Electrical Engineering and Computer Science, Oregon State University.
- Wangmutitakul, P., Li, L., and Minoura, T. User Participatory Web-Based GIS/Database Application. In. *Proc. of Geotec Event Conference*, March 2003.
- Wangmutitakul, Paphun, et al. WebGD: Framework for Web-based GIS/database Applications, *Journal of Object Technology 3*, 4, 209-225, 2004.
- Wuttiwat, T., Minoura, T., and Steiner, J. (May 2003). Using Digital Orthographic Aerial Images as User Interfaces. In *Proc. of ASPRS Annual Conference*, Anchorage, Alaska.