XML-based Vector Graphics: Application for Web-based Design Automation

Julian H. Kang, Texas A&M University, U.S.A. (juliankang@tamu.edu)

Byeong-Cheol Lho, Sangji University, Korea (bclho@sangji.ac.kr)

Jeong-Hoon Kim, Kumoh University, Korea (jhk@kkk)

Young-No Kim, Texas A&M University, U.S.A. (young@tamu.edu)

Summary

Most retaining walls and box culverts built for arterial road construction are simple, and the design process of these structures is often repetitive and labor-intensive because they are so similar in structural configuration. Although some integrated design automation systems developed for retaining walls and box culverts have expedited the design process of these structures, the process of collecting and distributing the resultant engineering documents has not been fully integrated with the computer applications. We have been developing a Web-based design automation system to manage the resultant documents as well as to speed up the repetitive design process.

Manipulation of engineering drawings in the Web page is one of the critical functions needed for Web-based design automation. eXtensible Markup Language (XML) and XML-based vector graphics are expected to facilitate the representation of engineering drawings in the Web page. In this paper, we present how we used XML and Scalable Vector Graphics (SVG) to compose engineering drawings and represent them in the Web page. XML Data Island we designed to define drawing components turned out effective in manipulating the engineering drawings in the Web page.

1 Introduction

Every year, many simple structures such as retaining walls and box culverts are designed for infrastructure development. The design process for these structures is often repetitive because their shape and function are similar regardless of their location. Professionals in the Architecture, Engineering, and Construction (AEC) industry have tried to speed up repetitive design processes by using integrated computer applications with which engineers can implement all design tasks seamlessly, from the analysis of structural stability to the generation of CAD drawings. Usually these computer applications take design parameters such as the geometry of a structure and implement all the engineering calculations needed to generate engineering reports and drawings automatically.

Application of Information Technology (IT) has helped construction professionals realize that effective document management, especially in a central repository, is an essential step for sustaining infrastructure. The public sector in the construction industry is especially interested in creating project legacy data to use in making decisions about how to maintain structures. Korean government has started requesting design firms to submit electronic design documents for managing infrastructure in the future. However, electronic design documents collected may not be effectively stored in the central repository and distributed because of the following reasons. Firstly, the format of electronic documents keeps changing as software advances. Application of an electronic document management system to future infrastructure management may not be easy because advancement in software development often exceeds what the electronic document management system can handle. Secondly, most integrated design automation systems are developed for stand-alone computer systems. Information loss or

duplication of resultant engineering documents may hardly be avoided unless they are transported from individual design firms to a central repository of the electronic document management system automatically. Indeed, in many fields it has been notoriously difficult to create, access, and adapt project legacy data effectively. Unless the entire process from engineering design to document management is integrated with electronic document management system, creating project legacy data may remain problematic.

Web-based design automation and management has attracted professionals in the construction industry because it would facilitate the structural design and resultant document management over the Internet. One obstacle in integrating design automation systems with Web-based project management to create project legacy data is to generate CAD drawings and manipulate them in the Web page. Representation of vector graphics in the Web page, which is one of the critical requirements for manipulating engineering CAD drawings over the Internet, was not convenient until the recent emergence of Internet technologies such as Extensible Markup Language (XML), Vector Markup Language (VML), and Scaleable Vector Graphics (SVG). XML is a text format designed to manipulate large-scale electronic publishing and data exchange over the Internet. VML and SVG are applications of XML which define tags for illustrating vector graphics in the Web page.

The growth of compatible Internet design technologies has inspired us to develop a Web application to integrate the entire design process of simple structures from structure analysis to document management. This Web application is expected to provide such functions as: 1) Collection of design parameters, 2) Analysis of the structural stability, 3) Selection of optimum structural members, 4) Generation of engineering reports and CAD drawings, 5) Representation and manipulation of the vector drawings in the Web page, 6) Generation and representation of the bill of materials, and 7) Management of structure maintenance records. The engineering drawing employed in our development is composed of several drawing components, such as plans and sections. The scale and location of drawing components in the drawing can be modified by the users over the Internet. In this paper, we present how we used XML and SVG to compose engineering drawings and manipulate them in the Web page.

2 XML-based Vector Graphics

Bitmap graphics, such as raster images and photos, successfully represent graphical information on the Web page. Bitmap graphics are described by millions of pixels in a binary format such as GIF and JPEG, which require a special package for modification. As opposed to bitmap graphics, vector graphics are described by a series of points to be connected. Vector graphics are resizable to any proportion without sacrificing graphical resolution. They are flexible because they can simply be re-rendered at any point. In order to facilitate the representation of vector graphics on the Web, the World Wide Web Consortium recently released both Vector Markup Language (VML) and Scaleable Vector Graphics (SVG), which are XML-based formats for vector graphics.

2.1 Extensible Markup Language (XML)

XML is a simple and flexible text format derived from the ISO 8879 Standard Generalized Markup Language (SGML). It is called extensible because it is not a fixed format like Hyper Text Markup Language (HTML). Originally designed to meet the challenges of large-scale electronic publishing, XML is also playing an important role in the exchange of a wide variety of data on the Web. Teague et al. (2003) defined XML as “a set of formatting rules that define structured information in a software-neutral text file.” Since XML defines the structured information in a simple text file, it can be recognized by virtually any computer system. Thus XML is very appropriate for building software-neutral project legacy data. By utilizing XML

Stylesheet Language Transformations (XSLT), information defined in XML can then be transformed into a common HTML document represented on the Web page.

In 1999, the International Alliance for Interoperability (IAI) proposed aecXML, an XML-based language designed for exchanging information in the AEC industry. Harrod (2003) noted that "the main idea with aecXML is to not only establish some standard ways of structuring building data but also to do it so as to enable automated processing of that data as much as practicable." Zhua and Issab (2003) suggested that a well-developed XML schema to classify construction data is a critical key issue for successful information exchange. Using the XML-based information standard, Tserng and Lin (2003) developed an electronic data acquisition model for project scheduling. They pointed out two major obstacles in gaining efficient access to information about multi-contract projects: 1) the variety of data structures that project participants use, and 2) the lack of an automatic mechanism for data acquisition.

2.2 Vector Markup Language (VML) and Scalable Vector Graphics (SVG)

VML is an application of XML which defines a format for encoding vector information together with additional tags for editing and displaying the information. VML is written using the syntax of XML, such as <v:shape/> and <v:path/>, and is compatible with HTML. A Web page encoded in VML can be displayed on the Microsoft Internet Explorer without installing any plug-in software.

SVG is another application of XML, designed for describing two-dimensional graphics in the Web page. It allows for three types of graphic objects: vector graphic shapes, images, and text. SVG graphics are scalable, so the same SVG graphic object can be displayed at different sizes on the Web page without sacrificing graphic resolution. In recent years, several researchers have built on this new way of representing vector graphics. Baravalle et al. (2003) demonstrated the use of SVG to produce a pictogram representation of numerical data obtained from scientific computer programs. Gonzalez and Dalal (2003) presented a web service that allows end users to specify a database query and visualize the extracted data as charts or graphs using SVG.

3 Development of Web-based Design Automation System

3.1 System Configuration

Our group has been developing a prototype Web application to facilitate the design automation of box culverts and retaining walls using ASP.NET and XML-based vector graphics technology. Design parameters are collected via a series of Web pages and saved in the database located in the Web server. An input text file for the structure analysis is created from these design parameters. The server application then calls the commercial package and initiates the analysis of the structure. After the structure analysis is finished, the server application reads the output text file of the commercial package to extract the maximum bending moment and maximum shear force. The bending moment diagram (BMD) and shear force diagram (SFD) are then displayed graphically on the Web page using VML. Once the user provides additional design parameters, the server application selects the amount and type of reinforcing steel bars needed to ensure structural stability. The server application then generates engineering drawings and save them in the database using XML tags. Engineering drawings are converted into SVG and displayed on the Web page. Drawings displayed on the Web page can be printed on the plotter. Figure 1 illustrates this process.

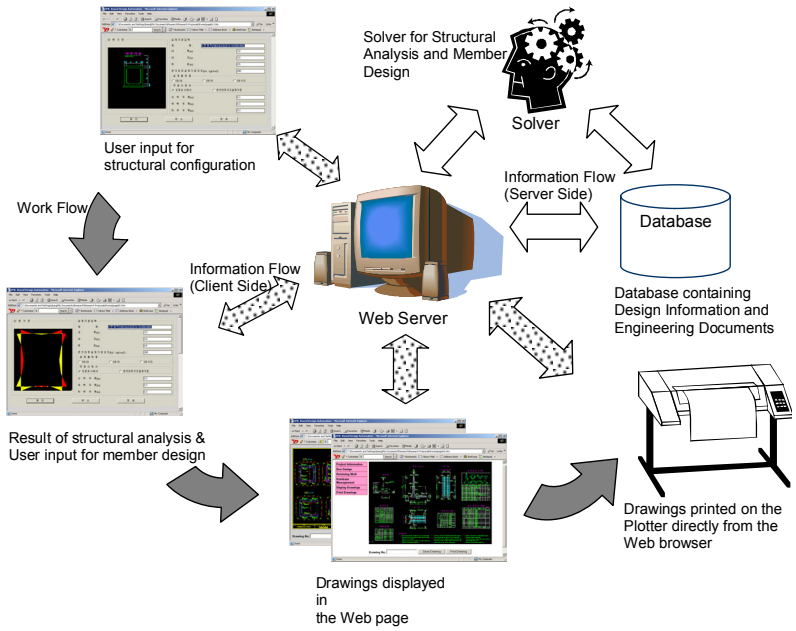

Figure 1: Process of Web-based design automation

3.2 Drawing Composition

An engineering drawing is composed of several components such as floor plans, sections, and detail views. One obstacle, in the process of generating the drawing automatically, is that some of these components may not be depicted properly in the drawing. A floor plan, for example, may be constructed too small or located at the awkward place in the drawing. Modification of the drawing generated by the server application may be an essential function to ensure the quality of the drawing. We decided to add a drawing modification function in a way that users can change the scale or location of the drawing component in the full drawing over the Internet. For this goal, we defined the full drawing as an object that is composed of multiple drawing components. The drawing component is also an object that is created independently with its own origin point. Similarly, the structure object can be defined as a composition of full drawing objects. The project object is composed of multiple structure objects. Figure 2 illustrates how we composed drawing objects.

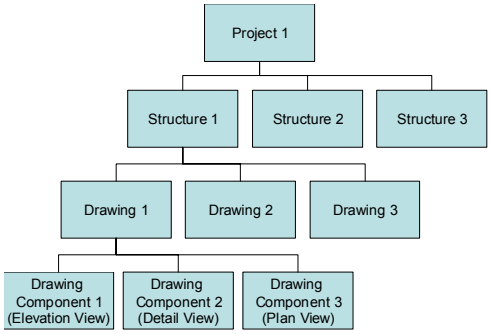

Figure 2: Structure of drawing objects

The local coordinate of the drawing component is converted into a global coordinate when the full drawing is composed. The location and size of the drawing object in the full drawing is first determined by the server application automatically and then adjusted by users. Users adjust the location of the drawing object by assigning an X and Y coordinate in the full drawing, to which the origin of the drawing component will move. The scale of the drawing object in the full

drawing can also be adjusted by the scale factor that users will identify. The diagram in Figure 3 illustrates how the full drawing is composed.

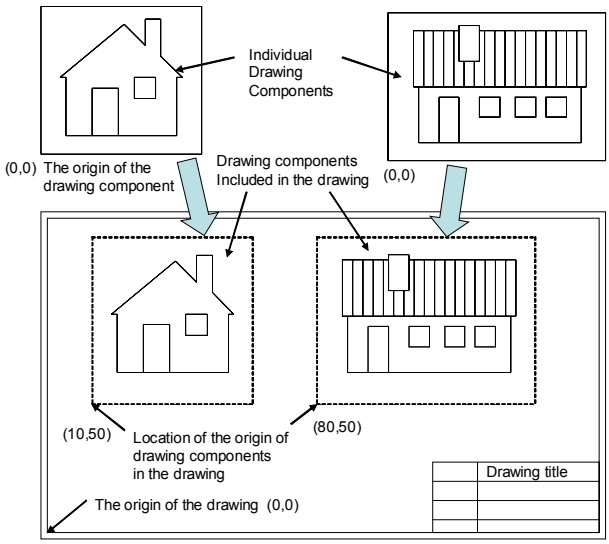

Figure 3: Composition of drawing

3.3 XML Data Island

We decided to use an XML Data Island, which defines the lines and texts using XML tags, to describe individual drawing components. This decision was made because we expected that drawing components defined in the XML Data Island would be easily converted into XML-based vector graphics such as SVG or VML. Figure 4 shows the XML Data Island we designed to define lines by the color, thickness, starting point, and end point. The first block in the XML Data Island in Figure 4, for example, defines a 3 point-thick blue line that connects (123,15) and (650,0). The drawing object is composed of a series of these lines.

```
<line>
      <s>blue</s>
      <sw>3</sw>
      <x1>123</x1><y1>15</y1>
      <x2>650</x2><y2>0</y2>
</line>

{Repeat for the number of lines… }

<line>
      <s>blue</s>
      <sw>3</sw>
      <x1>650</x1><y1>0</y1>
      <x2>240</x2><y2>150</y2>
</line>
```

Figure 4: XML Data Island that defines lines for drawing component

The full drawing is also defined in XML. In the process of composing the full drawing, multiple drawing components generated by the server application are combined and plugged in the designated space in the XML data shown in Figure 5.

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<drawing>

{The collection of XML data island to be plugged…}

</drawing>
```

Figure 5: XML data that defines a full drawing

3.4 Generation of SVG Drawings

Once an XML data for the full drawing is composed, it is temporarily saved in a text file. The sever application then convert this text file into an XML-based vector graphics using XML Stylesheet Language (XSLT). We decided to use SVG to display the full drawing in the Web page. Accordingly, the XSLT was developed to convert XML tags to SVG tags. Figure 6 shows a fraction of XSLT we designed. Figure 7 shows the SVG tags created by the XSLT.

```
<svg xmlns="http://www.w3.org/2000/svg"
            xmlns:xlink="http://www.w3.org/1999/xlink">
    <xsl:for-each select="drawing/drawingObject/line">
        <g style="stroke: {s}; fill: {f}; stroke-width: {sw}"
           transform="translate({moveX} {moveY})
                      translate({originX} {600 - originY})
                      scale({scale})
                      translate({-originX} {originY - 600})">
            <a xlink:href="step6_input.asp?doID={doID}">
                <line x1="{x1}" y1="{600-y1}" x2="{x2}" y2="{600-y2}" />
            </a>
        </g>
    </xsl:for-each>
</svg>
```

Figure 6: XSLT to convert XML line to SVG line

```
<?xml version="1.0" encoding="utf-8" ?>
<svg width="15in"
     height="15in"
     xmlns=http://www.w3.org/2000/svg
     xmlns:xlink="http://www.w3.org/1999/xlink">
<g style="stroke: blue;
          fill: none;
          stroke-width: 3"
          transform="translate(0 0)
          translate(0 460)
          scale(1)
          translate(0 -460)">
<a xlink:href="step6_input.asp?doID=317">
    <line x1="0" y1="460" x2="800" y2="460" />
</a>
</g>
</svg>
```

Figure 7: Sample SVG tag

3.5 Database Design

Normally, the XML data is stored in a software neutral text file. However, we decided to store the XML Data Island of the drawing component in the database. This decision was made because the XML data for the full drawing is composed of multiple XML Data Islands of drawing components, which can be dynamically changed based on how the user wants to locate in the full drawing. Accordingly, we designed tables in the database in such a way that the location information and scale factor of the drawing components are stored separately and utilized in the process of composing the XML data of the full drawing. We expect, by saving the drawing components in the database, that we should be able to allow the users to modify the full drawing after it is composed by the sever application automatically. Table 1 shows some of the fields we designed in the database.

The tblDrawingObject table contains the XML Data Island of the drawing component. This table stores 1) drawing component identification number, 2) drawing identification number to which this drawing component belongs, 3) the X and Y coordinates in the full drawing where this component will be located, and 4) the scale factor that will be used to determine the size of drawing component depicted in the full drawing. The tblDrawing table contains general information of drawings. It stores 1) drawing identification number, 2) structure identification number to which this drawing belongs, and 3) other general information of the full drawings. The tblStructure table contains design information of the structure to be designed. It also stores

1) structure identification number, and 2) project identification number to which this structure belongs. The tblProject table contains general project information. It may contain the project identification number and the name of project. Figure 8 shows the relationship between these tables. These relationships between tables make it possible to manipulate the drawing component as an object that belongs to the full drawing.

Table	Information to be managed
tblProject	- Project ID - General information of the project
tblStructure	- Structure ID - Project ID that the structure belongs to - Structure type - Design parameters - General information of the structure
tblDrawing	- Full drawing ID - Sturcture ID that the full drawing belongs to - General information of the full drawing
tblDrawingObject	- Drawing component ID - Full drawing ID that the drawing component belongs to - XML Data Island of the drawing component - Location of the drawing component in the full drawing - Sclae factor of the drawing component

Table 1: Database structure

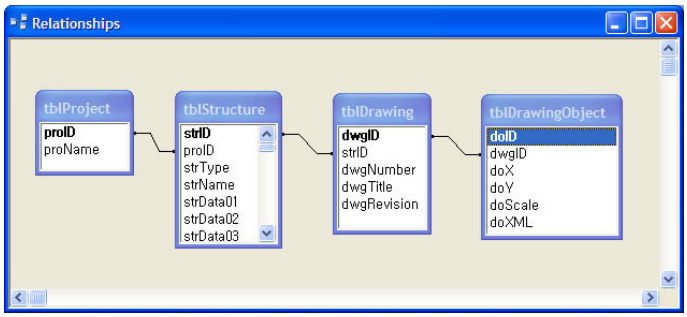

Figure 8: Relationships between tables in the database

4 Sample Test

In the process of composing the full drawing, the server application extracts the XML Data Island of drawing components from the database and positions them at a proper location in the drawing based upon the size and role of the drawing components. The server application then marks their location in the database and generates the XML data for the full drawing. As a specific component in the full drawing is selected, an editable Web page where the user can update the scale and location information of the component is delivered. If the user provides new location information or scale factor to the database in the server via the Web page as shown in Figure 9, the server application updates the full drawing accordingly. Figure 10 shows how a certain drawing object is updated. The front view, for example, is too small compare to the adjacent plan view and isometric view when the full drawing was created automatically. Although the user increases the size of the front view, it is not still located at the proper position yet. The user then adjusts the location of the front view to finish composing the full drawing.

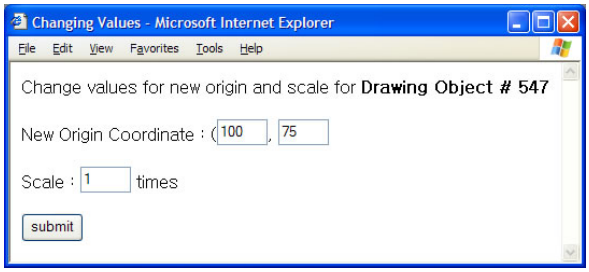

Figure 9: Web page for updating the drawing component

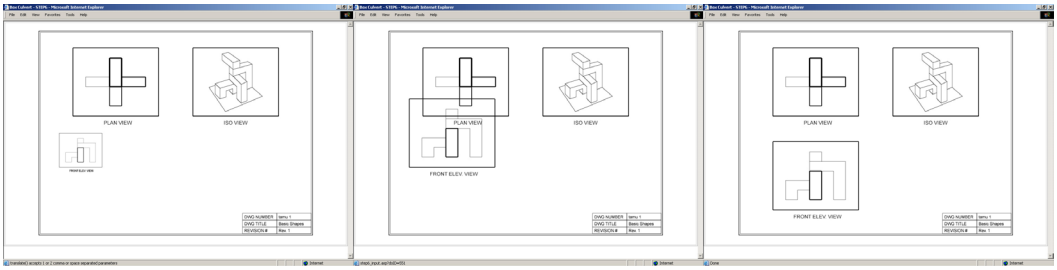

Figure 10: Update process of the full drawing

5 Conclusion

This paper demonstrates how XML and XML-based vector graphics can be utilized for representing or updating the drawings in the Web-based design automation system. The use of XML Data Island for defining the drawing component facilitates the composition and modification of the full drawing. Although it is not a usual approach to save the XML data in a relational database, the XML Data Island stored in the relational database works effectively in creating the full drawing and updating it over the Internet. Especially, the associated location information and scale factor stored separately in the database facilitates the process of updating the full drawing. Every time the user updates the location or scale of the drawing component, the server application composes a new full drawing by combining the XML Data Islands and their location information.

Manipulation of the drawing component in the sample full drawing demonstrates the potential of a Web-based CAD system in design automation and infrastructure management. Web-based CAD is expected to contribute significantly to the creation of the project legacy database in which resultant engineering documents are stored and used for the next project or future maintenance. The drawings stored in the central repository should be a great asset in cultivating collaboration among project participants. The use of Web-based project management has already demonstrated the usefulness of manipulating the project information in the central repository. Accordingly, the Web-based design automation system, which automates the repetitive process of designing the simple structure and save resultant engineering documents in the central repository, should speed up the design process and help the public sector in the construction industry sustain the built-in infrastructure. Indeed, our test to design a simple box culvert and generate the associated drawings took only about 15 minutes. This test indicates that the engineers, who specially design simple structures repetitively, will be benefited by the Web-based design automation system. Simple and repetitive structures can be designed in the Web page without running any structural analysis packages or CAD packages in the user's computer. The only application required to implement the design process is a Web browser. The public sector, such as the department of transportation, is also expected to improve the manipulation of engineering information for infrastructure management by easy access to the right data.

6 References

Baravalle A., Gribaudo M., Lanfranchi V. and Sandri T. (2003). *Using SVG and XSLT for graphic representation*. Proc., 2nd Annual Conference on Scalable Vector Graphics, Vancouver, Canada.

González G. and Dalal G. (2003). *Generating SVG Graphs and Charts from Database Queries*. Proc., 2nd Annual Conference on Scalable Vector Graphics, Vancouver, Canada.

Harrod G. (2003). *aecXML & IFC*. CADInfo.NET [WWW document] URL http:// www.cadinfo.net.

Teague T., Palmer M., and Jackson R. (2003). *XML for Capital Facilities*. Journal of Leadership and Management in Engineering, ASCE, 3 (2), 82-85.

Tserng H. and Lin W. (2003). *Developing an electronic acquisition model for project scheduling using XML-based information standard*. Automation in Construction, Elsevier B.V., 12 (1), 67-95.

Zhua Y. and Issab R. (2003). *Viewer controllable visualization for construction document processing*. Automation in Construction, Elsevier B.V., 12 (3), 255– 269.