21

Diagram Drawer&Browser for visual information processing on the WWW

*Wu Yue** *Christophe Gnaho***
** Computer Science & Engineering College*
University of Electronic Science & Technology of China
610054 Chengdu, Sichuan, China
Tel: 86 28 3202680 Fax: 86 28 3334131
E-mail: ywu@postoff1.uestc.edu.cn

*** Centre de Recherche en Informatique*
90 rue de Tolbiac
75013 Paris, Franc
Tel: 33 0140774634 Fax: 33 0140771954
E-mail: gnaho@univ-paris1.fr

Abstract

Diagram techniques are widely used in visual information processing, hyperlink mechanism used successful in HTML can flexibly link related information, and Java applets can dynamically put diagram into Web pages that makes Web applications with more lively visual display. By combining these three techniques, we have designed the Diagram Markup Language (DML) for describing the structure of diagrams, and implemented a tool, called Diagram Drawer&Browser (DD&B), for supporting visual information processing on the WWW. This paper presents the implementation of DD&B, introduces the DML, and gives an application example.

Keywords

Diagram techniques, Applet, Visual information processing, WWW application

1 INTRODUCTION

Information Systems in the WWW Environment C. Rolland, Y. Chen & M. Fang (Eds.)

Data displayed in a diagram, relational context is often more effective, more understandable than just published in a simple table of data, where relationship and context must be inferred. Diagrams are widely used in visual information processing, and its importance has been recognised by many Web users. Diagrams associated to modelling techniques have been introduced, such as, Data-flow diagrams, Entity_Relationship diagrams, State-transition Diagrams, Petri nets, etc. They are widely used in a variety of application areas, such as, software engineering, databases, design of electrical circuits, networks, and so on. Drawing techniques, such as, scrolling, paging and popup subwindows have been developed (Wei Lai, Maurice Danaher,1997). However, there is no consensus on the diagrams that should be used nor real standardisation of drawing techniques (Shi-Kuo CHANG,1990). In very large diagrams, the main problem is how to layout them on the screen. The hyperlink mechanism used successfully in HTML can link information together in a meaningful way. Java applets can dynamically put diagram into Web pages that makes Web applications with more lively visual display. they are especially suitable for WWW applications. Our motivation is to develop a generic diagramming tool with applets which combines the diagrammatic techniques, the hyperlink mechanism, and Java techniques for supporting visual information processing on the WWW. The purpose of this paper is to introduce the implementation of Diagram Browser/Editor (DD&B), which combines the diagram techniques, the hyperlink mechanism, and Java techniques for supporting visual information processing on the WWW.

The remainder of this paper is organised as follows: Section 2 presents the architecture and the implementation of DD&B. Section 3 describes the Diagram Markup Language (DML), which provides a mechanism to define the attributes of the element and relation, and the hyperlinks in them. Section 4 gives an application example. Section 5 concludes about some features of DD&B.

2 DIAGRAM BROWSER/EDITOR (DD&B)

2.1 Application Environment

DD&B is implemented with Java applets and can be integrated into many applications. It can be used as a diagram drawer and a diagram browser in the environment of visual information processing on the WWW. It uses diagrams to abstract and express information, and uses the hyperlinks to link and integrate information. Its application environment is shown in Figure 1.

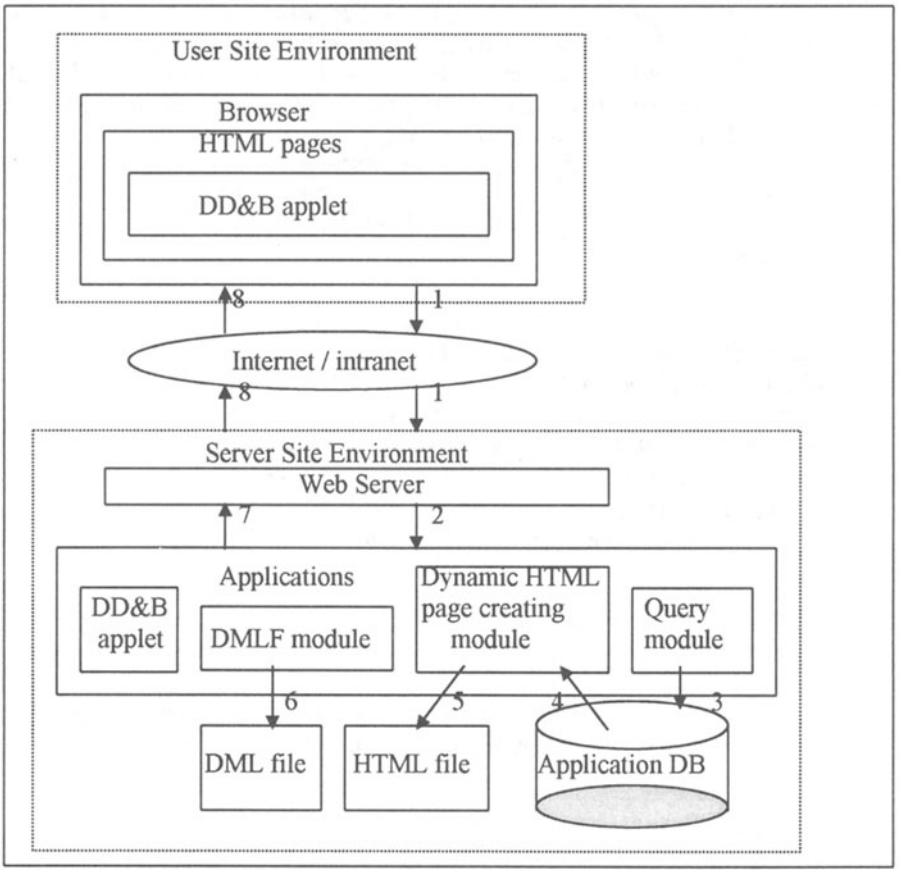

Figure 1 The DD&B Application Environment.

1Browser requests to get a HTML page
2 Web Server invokes application Query module
3 Query module gets data from Application DB
4 Application DB returns the results to Dynamic HTML page creating module
5 Dynamic HTML page creating module generates a HTML page embedding an <applet> tag

6 DMLF generates a DML file
7 Web Server returns the HTML page
8 The DD&B applet downloaded from server site runs on user site, and reads the DML file

The DD&B application environment is one typical Web client/server environment, which is divided into user site environment and server site environment connected by the Internet or intranet. In user site environment, users utilise browser to get HTML pages and run applets. Generally, the server site environment consists of the Web server, applications, and application DB. For using the DD&B applet, two things are needed to be done while dynamically creating a HTML page: one is to create dynamically the Diagram Markup Language file, which is the input data file of DD&B; another is to add the <applet> tag with the parameter item into the HTML page, the HTML page should include following <applet> tag:

```
<applet codeBase=http://paris1CRI/java/DiagramDrawBrowser
code=DiagramDrawBrowse.class width=600 height=500>
<param name=DMLFName value="DMLFName.txt">
</applet>
```

The codeBase item describes from where the DDB will be downloaded. Here the host is paris1CRI, the path is java/DiagramDrawBrowser. The code describes the name of applet. Here the DD&B applet is called DiagramDrawBrowse.class. the DMLFName.txt is the name of Diagram Markup Language File.
One way to integrate DD&B into application is to add a submodule (called as DMLF module in Figure 1) into applications for performing the tasks of generating the DMLF and adding the <applet> tag into the HTML. The DML file is dynamically created by using the data gathered from application DB.

2.2 Architecture

The DD&B manages the user operating interface, sends the message to appropriate methods, and transforms the different representations of a diagram. Its architecture is shown in Figure 2.

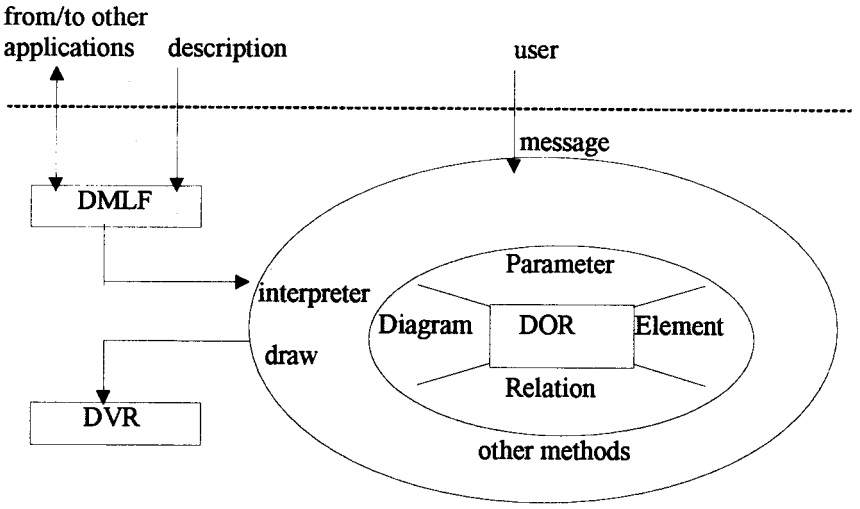

Figure 2 the Architecture of DD&B.

In this architecture, one diagram has three different representations:

Diagram Markup Language File (DMLF)
DMLF is a text file which contains the tags of DML. It provides a mechanism to describe the attributes of diagram elements, the topology of diagram, and the points and the destinations of hyperlinks. It is also a data interface with other applications, such as Web applications which can dynamically create the DMLF file by using the data in application DB, then interface DD&B for drawing diagrams. The DMLF is interpreted by interpreter() method to create the structure, features, and objects of the diagram, i.e. the Diagram Object Representation.

Diagram Object Representation (DOR)
DOR is the diagram's internal representation using objects. It contains sets of objects and the geometry of each diagram element. Using the draw() method, the objects which are in current visible area are mapped on screen, so the Diagram Visualising Representation is created.

Diagram Visualising Representation (DVR)
DVR is what users view on screen. It is created by laying out and visualizing diagram objects in DOR. It has many visual properties, such as, different colors or types can be used to represent the special nodes. And some navigating mechanisms are also provided on DVR.

2.3 Functions

Navigating diagram and laying out diagram are two powerful functions provided in DD&B. We have provided navigating mechanisms as follows:

Overview mechanism
When a diagram has many nodes or edges, frequently the screen is not enough to display every details clearly. So two view models are provide : one outline model for viewing the diagram structure using node names ; another detail model with the complete context. By clicking the Overview button, we can switch between these two view models.

Hyperlink mechanism
Two kinds of hyperlinks are also provided: One changes the navigation path, and jumps to the other diagram ; Another does not change the navigation path, it connects only with a document.

Browse upward/downward mechanisms
Using BrowseUpward button can reach at the previous level of the diagram, clicking at any diagram node can make it as the current root and browse downward.

Focus mechanism
We can see the details of a diagram element by focusing on it.

Diagram layout is a very complex problem. For dealing with this issue, we provide both automatically layout mechanism and interactive editing layout mechanism in DD&B.

Automatically layout mechanism
The diagram layout is created automatically according to the dimension of the graphic area and the visual levels. From the current root, the set of current visual diagram elements is created, and the location of current visual nodes and their links are computed.

Interactive editing layout mechanism
Designers can use the Editing layout function to modify, or adjust the layout for getting more interesting-looking diagram by using *drag-and-drop* mechanism to move diagram elements to appropriate location. After editing, all parameters of this final diagram, which include the description of the diagram, and the location of each diagram element can be saved as a final DML file. The description of the diagram is the same as original DML file, but each diagram element has a exact location.

3 DIAGRAM MARKUP LANGUAGE (DML)

Notation

In DML, we use the notation as follows:
The bold words are key words to be interpreted. The italic words are names of set variable, their elements can be used or redefined. The underlined words are the default values. The <> and </> are used as a pair to define a tag, <> presents the start , and </> is the end of this tag. The [ ] expresses that the contents is optional. The {item1/item2 /...} presents that one and only one item can be selected. The // is used as a comment line.
DML, which is much similar with Hypertext Markup Language (HTML) describes how a diagram should be constructed. In DML, there are four tags: **<Parameter>, <Diagram>, <Element>** and **<Relation>**. All tags can be omitted, or repeated several times.

Parameter tag

The Parameter tag acts as a changeable interface, which describes five types of set variables.
<Parameter> // defining the system parameters for a diagram
[***element_type_set***=(**NoType,Rectangle, Circle, 3DRectangle, ...**)] // graphic elements
[***relation_type_set***=(**Adjoin, Separate, Choice, Decomposition, ...**)] // relation types
[***color_set***=(**Yellow,White,Gray,Blue,Green,Red, ...**)] // color names in **java.awt.color**
[***linestyle_set***=(**Times New Roman, Arial, ...**)] // font names in **java.object.font**
[***narrow_type_set***=(**No, Single, Double, ...**)] // narrow types
</Parameter>
When this tag is omitted, the built-in parameters are used. By changing the parameters in the Parameter tag, this DML can be easily tuned to different applications.

Diagram tag

The Diagram tag decomposes the diagram or subdiagram into constructive blocks.
< **Diagram** >// defining the components of a diagram or subdiagram
Name=String // the name of this diagram or subdiagram
[**Dimension**=(width,height)]
// a pair of two integers denotes the width and the height of this diagram or subdiagram area
[**Location**={(x,y)/**Centre/East/South/West/North**}]
// the location of this diagram area in the window or the location of this subdiagram in its parents area. We can use a pair of two integers (x,y) as the up_left co-

ordination of this diagram area, or use some kind of relative co-ordination, such as, **Centre**, **East**, **South**, **West** and **North**. The default value is (0,0).
[Children=(Name[,Name]...)]
// defines a including relation. These Names used here are names of subdiagrams or elements that will be defined late. They are at the same level and are the children of this diagram or subdiagram being defined.
</ **Diagram** >

Element tag

The Element is one basic block of a diagram or subdiagram. Each Element tag defines the attributes of one element.
<**Element**> // defining the attributes of each element
[**Name**=[{Hyperlink}]String]
// the name of this element. The default value will be E0,E1... We can use {} here to contain a hyperlink, which is an URL address or a file name of a diagram. We can connect it from this element.
[**Type**=Element of ***element_type_set***] // the type of this element
[**Color**=Element of ***color_set***] // the color of this element
[**LineStyle**=Element of ***linestyle_set***] // the lineStyle of this element
[**Annotation**=[{Hyperlink}]String]...
//one line annotation of this element. We can repeat it for multilines. We can use {} here to contain a hyperlink, which is an URL address or a file name of a document.
</**Element**> // finishing the definition of one element

Relation tag

The Relation tag describes the structure of blocks that expresses how this diagram is constructed by these constructive and/or basic blocks. Each Relation tag defines one connecting relation between two elements, or between two subdiagrams. When this tag is omitted, it means that the elements in this diagram has no relation.
<**Relation**> // defining the attributes of each Relation
From=Name **To**=Name
// **From** and **To** are two Names to which this Relation connects.
[**RelationType**=Element of ***relation_type_set***] // the relation type of this relation
[**Color**= Element of ***color_set***] // the color of this relation
[**LineStyle**= Element of ***linestyle_set***] // the line style of this relation
[**NarrowType**=Element of ***narrow_type_set***] // the narrow type of this Relation
[**Annotation**=[{Hyperlink}]String]...
// one line annotation of this relation. We can repeat it for multilines. The default value is no annotation. Especially we can use {} here to contain a hyperlink which is an URL address or a file name of a document.
</**Relation**>

4 EXAMPLE – THE EKD ELECTRONIC GUIDE BOOK

This section shows an example running in DD&B, which tries to demonstrate some of the functionalities already described in the previous sections.
EKD Electronic Guide Book allows any user to gain knowledge on the use of the EKD concepts through the Internet (Loucopoulos, P., Kavakli, V., Prekas, N., Rolland, C., Grosz, G. and Nurcan, S., 1997). Its application environment based on the Web intrinsic client/server environment which is similar with Figure 1. It consists of two inter-related modules: the User Guidelines Browser which runs on the client end, and the Guidelines Search Engine which runs on the server end. The Method Knowledge Base implemented with Microsoft Access RDBMS is used as the application DB, the WebSite is used for the Web Server, and the applications consist of a set of Common Gateway Interface (CGI) applications using Visual Basic and Java applets. As the Method Knowledge Base in EKD Electronic Guide Book may be very large containing many hundreds of methods and relations, the DD&B is used as a diagram browser for drawing the graphical representation of the method chunks proposed by the methodology and the different ways to navigate from one chunk to another.
Assuming the user has selected the context (S. Si_Said, C. Rolland, G. Grosz, 1996): SITUATION is Goal, INTENTION is Reduce Goal, a query is sent to the server asking for the execution of the context, and the DD&B is downloaded to draw the graphic. The textual and graphical definitions of context are displayed on the left and the right frame respectively as shown in Figure 3.

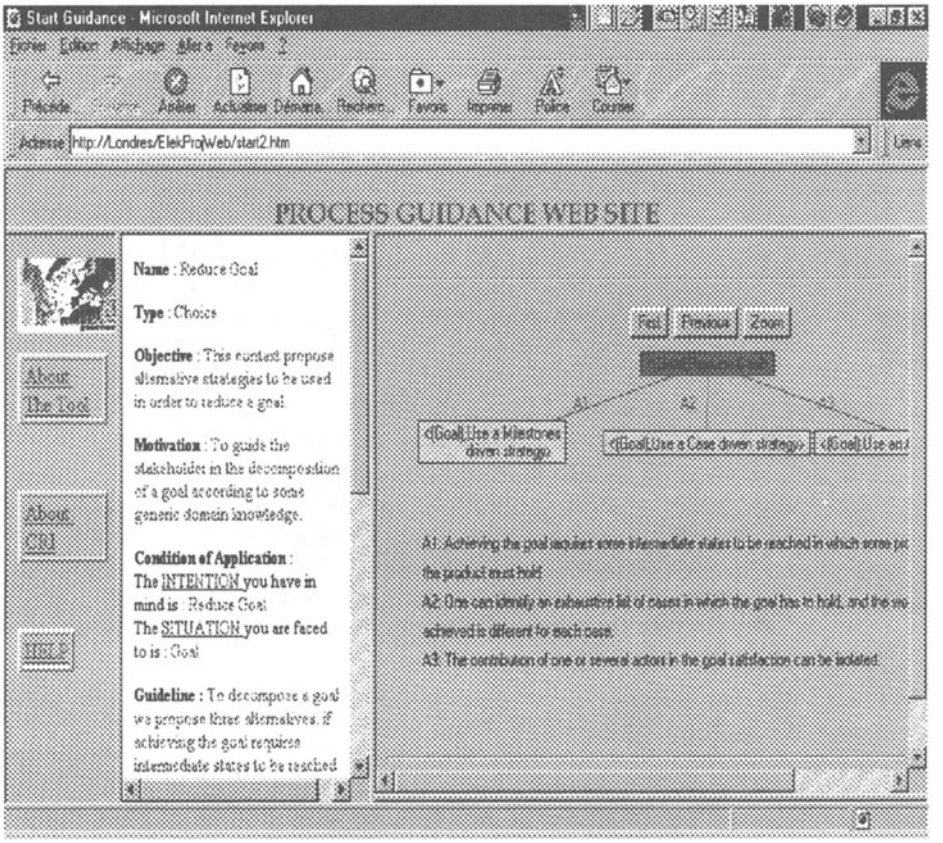

Figure 3 An Example of DD&B.

In the right graphical frame, user can click Previous button to browse upward, or click any node to make it as the current root and to browse downward. Zoom button can be used for enlarging or shrinking the graphic.

In this example, the graphic can be designed as two subgraphic: one is a tree with four nodes; another has only five line annotations. Using DML, the graphic can be represented as follows:

<Diagram> // describe the subdiagram of tree
Name=Subdiagram1
Location=North
Children=(ID-0100, ID-0110, ID-0120, ID-0130)
</Diagram>
<Diagram> // describe the subdiagram of annotation
Name=Subdiagram2
Location=South
Children=(line1, line2, line3, line4, line5)
</Diagram>

<Element> // describe the node of ID-0100
Name=ID-0100
Type=Rectangle
Annotation=<(Goal),Reduce Goal>
</Element>
// the definitions of other nodes are similar with node ID-0100
<Element> // describe the annotation line as a node with the type of NoType
Name=A1
Type=NoType
Annotation=A1: Achieving the goal requires some intermediate states to be reached in which some properties about
Annotation=the product must be hold.
</Element>
// the definitions of other annotation lines are similar with A1
<Relation> // describe the edge between two nodes
From=ID-0100 To=ID-0110
RelationType=Choice
Annotation=A1
</Relation>
// the definitions of other edges are similar with above description

5 CONCLUSION

DML is a simple, powerful and extensible markup language. Using default values, the description for a diagram is quite simple. Using the Children item in the Diagram tag, any complex diagram can be decomposed with the hierarchical structure. Using the Relation tag, not only any complex relation can be described, but also some discrete and different diagrams can be integrated into one diagram. And using hyperlinks in the name of element, or in the annotation of element or relation, some related diagrams or documents can be easily linked together. The Parameter tag provides a changeable interface. We can change the elements in some set variables, or add new elements into some set variables for different applications.

The DD&B implemented with Visual J++ version 1.0 provides rich navigational mechanisms around diagrams. It manages the transformation among the different representations of a diagram, and directly edits on diagram elements using visual mechanism. The features of DD&B are the results of mixing various techniques. First, it is a Java applet, so it has most characteristics of Java, such as platform independence, and of the object-oriented paradigm, such as objects, classes, and inheritance. Second, it merges hyperlink mechanism with diagram techniques, so it has both easy understandable and rich navigational capabilities. Third, it is powerful for representation and treatment. Based on the simple and powerful DML, it can easily represent and draw various famous diagrams, such as, Data-flow diagrams, E-R diagrams, State-transition diagrams, Petri nets, etc. And by using focus,

distortion-oriented, hyperlink, scrolling and paging mechanisms, it provides a powerful approach to navigate the details or outline of a diagram, or related information. It has provided many visual properties, such as, different colors or types can be used to represent the special nodes.

6 REFERENCES

(Loucopoulos, P., Kavakli, V., Prekas, N., Rolland, C., Grosz, G. and Nurcan, S., 1997) Using the EKD Approach: The Modelling Component, UMIST, Research Report (ELEKTRA project), ELEKTRA/WP2/T2.1/UMIST/3, March 1997.
(Shi-Kuo CHANG,1990) Principles of Visual Programming Systems (ed. Shi-Kuo CHANG) Prentice-Hall, Inc.
(S. Si_Said, C. Rolland, G. Grosz, 1996) Mentor : A Computer Aided Requirements Engineering Environment. in Proceedings of the 8th CAiSE Conference Challenges In Modern Information Systems, May 1996, Heraklion, Crete, Greece.
(Wei Lai, Maurice Danaher,1997) Information Browsing via Diagram Displays. Association for Information Systems Proceedings of the Americas Conference on Information Systems, August 15-17 1997, pp62-64, Indianapolis, Indiana USA.

7 BIOGRAPHY

Yue Wu received the B.S. and M.S. degrees from Harbin Institute of Technology in 1982 and 1984, respectively, both in computer science and Engineering. Since August 1984, he has been on the faculty of the Department of Computer Science and Engineering at University of Electronic Science and Technology of China, Chengdu, P.R. China. From 1991 to 1993, he was awarded the National Research Fellowship and was a visiting scholar at University of Geneva. From 1995 to 1997, he worked as an expert of 863 National High Technology CIMS in Networking DataBase Group. He is currently a Vice-Dean of the Computer Science and Engineering College at University of Electronic Science and Technology of China and a visiting associate professor at University of Paris 1, France. He has co-authored several journal papers and conference publications. His current areas of research interests include distributed object computing, Java and mobile agent technologies.

Christophe GNAHO graduated in Computer Science and Telecommunication Network Science, in 1992, from "Ecole supérieure d'Ingénieurs en Informatique et Génie des Télécommunications, France, and received a DEA (Diplôme d'Etudes Approfondies) in electronic, from Universite d' Orsay (Paris IX), in 1993. He is currently a PhD student at CRI/Universite Paris 1. His research topics concerns information system development methods modelling and CASE tools.