在全球資訊網上發展知識發掘之圖像化推理環境

An Embedded Visual Programming Interface for Intelligent Information Retrieval on the Web

李宗南
Chungnan Lee

國立中山大學 資訊工程研究所
Institute of Computer and Information Engineering
National Sun Yat-Sen University
Kaohsiung, Taiwan, ROC

陳耀宗
Yao-tsung Chen

國立中山大學 資訊管理研究所
Department of Management Information System
National Sun Yat-Sen University
Kaohsiung, Taiwan, ROC

摘要

在本論文中，我們將描述在全球資訊網路上發展的一套分散式知識發掘之圖像化推理系統。此系統整合了圖像化推理、全球資訊網瀏覽器、爪哇程式片段、推理引擎、資料庫伺服器來提供使用者圖像化的推理介面。
關鍵字: 圖像化推理、分散式、全球資訊網

Abstract

This paper describes an embedded visual programming interface for intelligent information retrieval built on the top of the Web.
Keywords:Visual programming, Distributed system, WWW

1. Introduction

As the amount of information in databases has rapidly grown in the past few years. It is a practical need to extract the high-level information, such as the trend of stock and commodity, diagnosis of finance, market planning, etc., from low-level data, such as stock index, finance report [1-4], etc.

The demand for user-friendly programming systems for nonprogrammers and professionals alike is also increasing tremendously. One of successful approaches to this demand is to use visual information in the man-machine interface [5-7]. A number of commercial packages, such Visual Basic, Visual C++, Delphi, etc. are the typical utilization of the visual programming concept. Visual programming is the process of reasoning and making inferences, based upon visual clue or presentation. Typically, the human draws a picture, a structured diagram, or a visual example, and the computer interprets the visual expression to understand the user's intention.

With the popularity of the WWW it is natural to use the WWW to access the database server. Some of such systems can be found in [8-10]. However, to extract the knowledge from the low level data over the WWW is just a beginning of focus. To access a Web database server, a user can issue a query through a Web browser to HTTP server. Upon receiving the user request, the HTTP server activates the Common Gate Interface program to handle the user requests and then returns the query results through the HTTP server to the Web browser. This is just a simple database search with a static HTML-based interface. As the Web gives a convenient, uniform user interface, most of the queries are still through the text mode. Since the visual expression is an effective communication between human and machine. To enhance the functionality of the Web browser, it is definitely worthwhile to incorporate the visual programming into the browser. In this paper we present a visual programming interface for information retrieval system on the top of the Web.

2. System Design Concepts

In this section we will give the basic design concepts, building blocks, and the basic theory for the system.

2.1. Design Principles

An embedded module

It is important not to change the current Web clients and database servers. Hence, the system is developed as an

embedded module to serve as a gateway between the Web client and servers. Meanwhile, it can provide the required functions to let users extract the required knowledge.

Visual programming
One of key idea behind the system is to provide a visual programming ability to the existing Web client. As most of the Web clients still send queries through text or template inputs to the Web server or database servers. A visual programming interface gives users another dimension to explore the information on the network through some pictorial and symbol icons.

Distributed communication.
Though using the Java technology can prevent the bottleneck imposed by Common Gate Interface on the HTTP server [15] through the sockets connection. When a large number of the Web clients connect to database servers through a single gateway, the bottleneck can occur. To prevent communication traffic to increase speed and reliability, the system uses a distributed communication model.

2.2. Building Blocks

Active Icons
The principal of visual component in the visual programming interface is an active icon. An active icon has image, information, behavior, constraint, knowledge base, and reasoning mechanism. For the information retrieval, the characteristics of active icons are classified into several types including reasoning, decision, computation, basic concept data, etc. Icons of same types have a similar set of properties.

Visual Grammar and Parser
The visual grammar for information retrieval is a context-free grammar that is a four-tuple, $G=\{N, \Sigma, P, S\}$, where, N is the non-terminal symbols, Σ is the terminal symbols, P represents the production rules, S is the start symbol. Each grammar symbol is associated with an icon. The product rule P is the relationship of grammar symbols and is represented as function of non-terminal and terminal symbols, rel (N, Σ). The form of production rules can be written as, $Sen \rightarrow rel\ (N, \Sigma)$, where *Sen* can be terminal, non-terminal, start symbol or composite of them. The relationships can be a connection, a sequence of computations, a sequence of decisions, a parallel processing of computations, a parallel processing of decisions, etc.

One of most important characteristic of active icons is the ability to validate the connection between two active icons. However, the whole visual sentence cannot be validated by the neighboring icons only. Hence, a parser, which is based on context-free grammar mentioned above, is used to parse and analyze the relationships of visual components in the visual sentences. The components in the parser include visual program, icon dictionary, non-terminal icon dictionary, and iconic inference rules.

3. System Overview

The overall working system as shown in Figure 1 contains a number of Web clients, Web server, message handling agents, inference agents, and database servers that can distributely locate over the Internet. The system is intended to be used as an embedded mediator between the Web browser and database servers to support information retrieval. As long as the database server supports the standard SQL and ODBC or related protocols, the system can be tightly coupled to database servers. Hence, the system can be added to serve its purpose without changing the Web client and the existing database servers.

The Web clients can be located anywhere in the network. It uses a standard HTTP to load the HTML, Java classes, image files, and related functions from the Web server agent. Once the Java applets migrate to the Web client, the connection between the Web clients and Web server will be no longer existed. The rest of connections are among the Web clients, database servers and rule base servers. This kind of connection is a three-tier architecture [14], in which a standalone server is used as a gateway, passing the request and response messages between the applet and the remote server. It communicates with the Java applet at one end by protocols defined by the system, while accessing database servers at the other end by original client/server protocols.

3.1. Visual Programming Interface Agent

The visual programming interface agent is actually an agent for users to interactively make inference and information retrieval. The visual programming interface receives messages from users' inputs such as drag and draw, mouse clicking, and text editing, then activates corresponding messages and active icons. It may ask other agents to provide service on behalf of users' requests.

To add the visual programming interface agent,to the Web client, a user can send a quest to the Web server agent to load all related functions wrapped in Java applets. The system provides a visual component editor to allow users to input parameters.

The visual sentence is parsed by a parser embedded in the visual programming interface agent, then is sent the parsing results to the message handling agent. The message handling takes the requests from the visual programming interface agent through the Internet, passes the requests to the inference agent to do inference or to the database server to retrieve the facts. The requests include parameters, data or execution commands that are accepted by the inference agent.

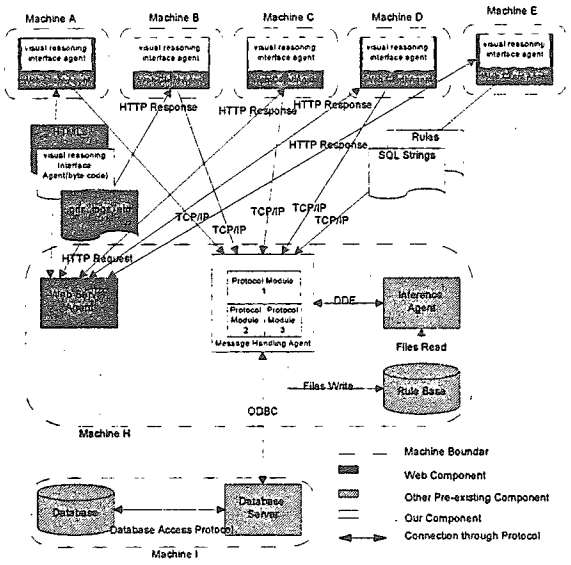
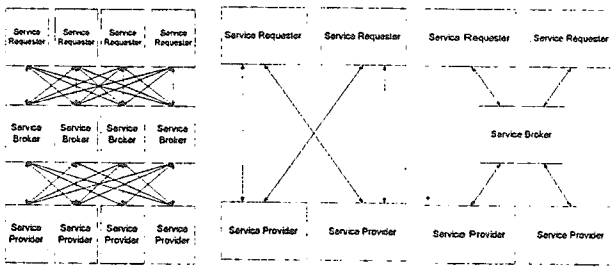


Figure 1 The overall architecture of the proposed system.

3.2. Distributed Communication Model

The information retrieval process involves communication among those agents and database servers, reasoning process on the inference agent and computation task on the MHA(Message Handling Agent). To avoid the communication traffic in the system, we use a CORBA(Common Object Request Broker Architecture)-like architecture as the communication model as shown in Figure 2 (a). The model consisting of 3 kinds of components – service providers, service brokers, and service requester gives a better communication speed and more reliable distributed connection compared to the models in Figures 2 (b) and (c). In the proposed model, service brokers act as the ORB(Object Request Broker) in CORBA, inserted into the client/server system to hide the server dependent protocols and to increase speed and reliability of the communication.

The reason to use such model is made clearly as follows. In Figure 2 (b) is a common client server model with service requesters and service providers. In case the communication protocols for the service providers are different, and then each service requester has to keep all protocols and address of various service providers in order to make a connection to request each service. To add a new service provider that needs a new protocol would be difficult. To solve the problem we add a service broker to handle address and protocols as shown in Figure 2 (c). If a new service provider with a new protocol is added to the system, only the new protocol module needs to be added into the service broker. Thus the service requesters are not affected by the new service provider. To further increase a reliable, fault tolerant and distributed connection, more service brokers have to be added as shown in Figure 2 (a).



(a) (b) (c)

Figure 2 Broker-like distributed communication model.

For the security reason Java applet can only establish a socket connection to the address where the applet comes from. If we only put one copy of the Visual Reasoning Interface Bytecode on the Web Server, then the running applets will always connect to one service broker. In the implementation, the service broker is implemented as an MHA(Message Handling Agent). To realize the communication model mentioned above, we setup the system as the following steps. We duplicate each service provider such as inference agent and database server and then setup the connection between each pair of the MHA and the service provider to avoid load imbalancing in one kind of service providers. Then, users can send an HTTP request to browse the Web page, The Web server will send a Web page with a script to randomly assign the URL of an MHA from which the visual reasoning interface agent bytecode is retrieved. The random assignment is based on the built-in routing table in the script to prevent load imbalancing among MHAs. Finally, the visual reasoning interface agent running in the Web client sets up the connection to the MHA based on the assigned URL. Since the random assignment mechanism is used, the running applets will

randomly connect to the different MHA.

3.3. Message Handling Agent

The message handling agent (MHA) composing of many protocol modules is also a mediator for the communication among agents. The MHA accepts messages from the client and routes them to the destination. In the current system, the MHA contains DDE for communication with the inference agent, ODBC for communication with database server, and TCP/IP module for communication with the visual programming agent.

3.4. Distributed Data Storage Management

The data used for the system are distributed over the network. Hence, a special care for the data management is needed. The Web pages and images are stored in the web server. Java classes can be replicated in many workstations which message handling agent runs to serve a vast number of clients. The database server only provides the raw data and can be replicated in many workstations to serve a vast number of clients, too. These data cannot be changed by users and are maintained by the system manager.

It is important to allow users to save and load their inference rules on the Web client. Unfortunately, a Java applet running on the Web client is not allowed to access any files on the local file system due to the security reason. To solve the problem we first store the file of inference rules on the message handling agent as the centralized management and allow users to transfer the file back to the Web client through the URL. The centralized files management provides a possible corporation of the clients under the circumstance that Java applets run on different clients cannot communicate directly. However, the cooperation can be achieved by accessing the files on the Web server.

3.5. Knowledge Representation & Reasoning Mechanism

We use the decision tree to represent the inference structure because it can be transformed to if-then rule format directly. Similarly, the decision table is used to describe the decision strategy for each node in the decision tree. In the proposed system, we use a commercial inference engine product called M.4 for the inference agent. The reasoning mechanism in M.4 is the backward chaining, and supports uncertain inference. The communication between the inference agent and the MHA is Dynamic Data Exchange (DDE). It is a method of interprocess communication (IPC) on Microsoft Windows platform. DDE uses shared memory to exchange data between applications and a protocol to synchronize the passing data.

4. A Technical Analysis Application

To illustrate the system we use a technical analysis example. First we give a background of the application. Then we focus on the design of active icons, inference rules, concrete of context-free grammar, and fuzzy functions. Finally we show the results.

4.1. The domain knowledge of the application

Technical analysis [11] is a term with complicated-sounding name for a very basic approach to investing. Generally speaking, technical analysis is the study of prices, with charts being the primary tool. It is an important knowledge for the investors. The Dow Theory, developed around 1900 by Charles Dow is the root of modern-day technical analysis. This root includes many principles such as the trending nature of prices, prices discounting all known information, confirmation and divergence, volume mirroring changes in price, and support/resistance.

The Relative Strength Index (RSI) is one of popular technical analysis index. It was first introduced by Welles Wilder in June 1978. As stated in [19], "The RSI is a price-following oscillator that ranges between 0 and 100. A popular method of analyzing the RSI is to look for divergence in which the security is making a new high, but the RSI is failing to surpass its previous high. This divergent is an indication of an impending reversal. When the RSI then turns down and falls below its most recent trough, it is said to have completed a failure swing. The failure swing is considered a confirmation of the impending reversal."

We use the failure swing of RSI as an example to illustrate our system. We assume RSI is a field of our database. The inference structure is shown in Figure 3. The definitions for active icons used in the financial diagnosis can be used in this application again. Only the parameters have to be changed. The RSI zone shows that yesterday's RSI tops above 70 or bottoms below 30 or lies between tops and bottoms. The trend direction shows that RSI turns down or turns up. The stock operation suggests that investor should buy or sell the stock or neither.

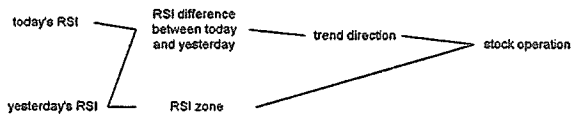


Figure 3 Inference structure for technical analysis

Active Icons Creation.
First, we create an active icon class inherited by other active icons, and then we create five types of active icon classes as follows.

1. Basic concept icon class: represents the basic concept, like RSI, MACD etc. The fact retrieving algorithm embedded in the icon is a couple of the SQL statements defined by users, we create instances to reuse each accounting concept.

2. Computation icon class: computes the basic concepts that are linked to it based on the operation defined by users.

3. Fuzzy icon class: divides the basic concept or computation result into several intervals given by users.

4. Decision icon class: the intermediate goal through the inference process, and user can define the decision table.

5. Goal icon class: similar to the decision icon class, just the final goal.

6. The images of five types of icons are shown in Figure The properties of the icons are added through the icon editor.

Figure 4 The images of basic concept, Computation, fuzzy, decision, and goal icons.

4.2. Interaction with the System

4.2.1. Designing the visual inference structure

User can design her/his own application in the Visual Programming Area by dragging the visual component from the Visual Component Box and instances of BasicConceptIcon, and arranging and connecting them to create the inference structure of an application. The steps to create such application are shown in Figures 5-7. First, we arrange active icons to form an inference structure as shown in Figure 3. Second, we customize the active icon by filling in the parameters. In the RSI zone icons, we fill in the parameter value as 30 and 70 in the if side, and the parameter values as top, internal and bottom in the then side as shown in Figure 5. In the stock operation icons, as shown in Figure 6, we fill in the parameter value as buy in the then side, if RSI zone is bottom and trend direction is up; or as sell, if RSI zone is top and trend direction is down; or as wait-to-see, if RSI zone and trend direction are other combinations. As shown in Figure 7, the suggest operation in this inference is to buy with 81% confidence based on the RSI data of an arbitrary company from Taiwan Stock Exchange Market on Dec 14, 1996.

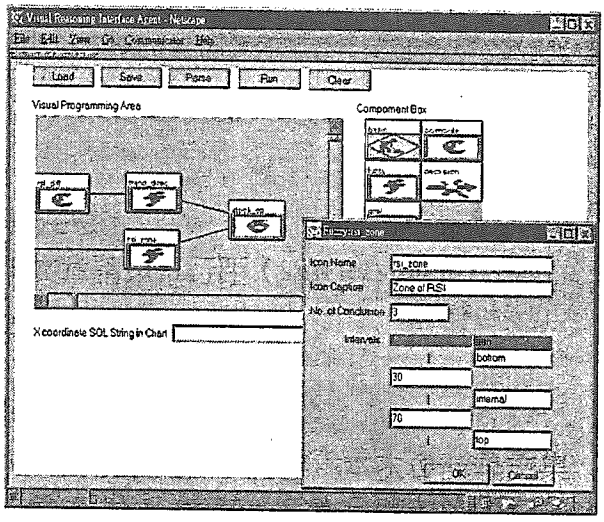


Figure 5 The parameters for the RSI zone icons.

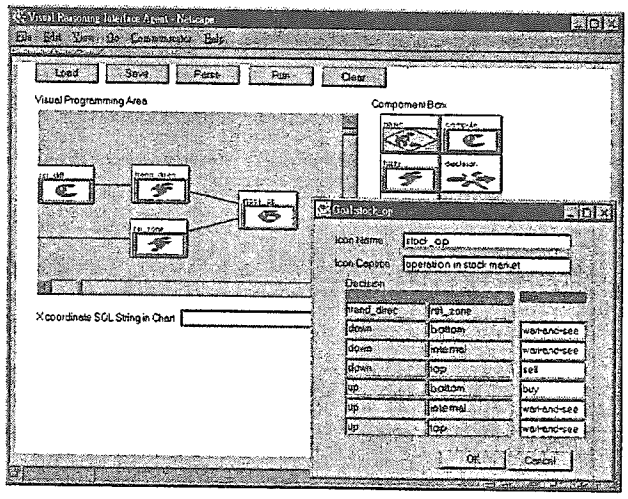


Figure 6 The stock operation icons and its parameters.

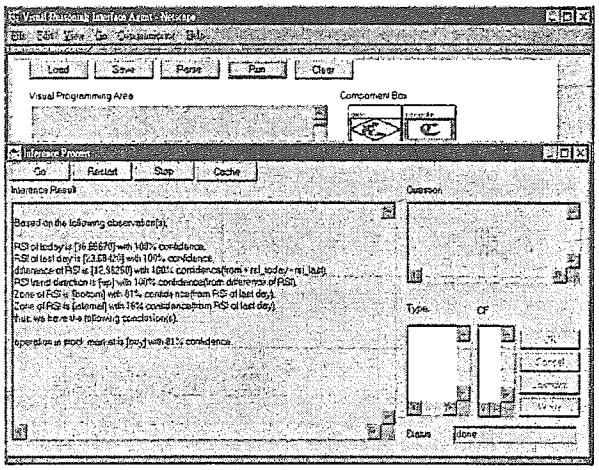


Figure 7 The inference results and text report generated from the system. The suggestion is to buy the stock with 81% confidence.

5. Conclusions

We have developed a distributed visual programming system for information retrieval. The system acts as a mediator between the existing Web client and database servers. It adds a visual programming functionality on the top of Web browser to provide a user another dimension of programming in addition to the traditional textual and template inputs. The system adopts a distributed communication model to increase the flexibility, reliability, and load balancing. Based on the domain of applications the user can design his own active icons, inference rules, fuzzy function, and computation. To maintain the flexibility of applications users have to fill in certain parameters. From the result of inferences the system performs very well over the Web.

References

[1] T. Anand, "Opportunity explorer: navigating large databases using knowledge discovery templates," *J. of Intelligent Information Systems,* Vol. 4, pp. 27-37, 1995.

[2] M. R. Klein & L. B. Methlie, *Knowledge-based Decision Support Systems With Applications in Business*, John Wiley & Sons, 1995.

[3] M. Frolick & N. K. Ramarapu, "Hypermedia: the future of EIS," *J. of Systems Management*, Vol. 44, pp. 32-36, 1993.

[4] R. T. Chi & E. Turban, "Distributed intelligent executive information systems," *Decision Support Systems*, Vol. 14, pp. 117-130, 1995.

[5] A. L. Ambler & M. M. Burnett, "Influence of visual technology on the evolution of language environments," *Computer*, Vol. 22, pp. 9-22, 1989.

[6] S. K. Chang, "A visual language compiler for information retrieval by visual reasoning," *IEEE Transactions on Software Engineering*, Vol. 16, pp. 1136-1149, 1990.

[7] N. C. Shu, *Visual Programming*, Van Nostrand Reinhold, 1989.

[8] S.E. Dossick & G.E. Kaiser, "WWW Access to Legacy Client/Server Applications," in: *Proc. 5th International World Wide Web Conference*, 1996.

[9] N. N. Duan, "Distributed database access in a corporate environment using Java," in: *Proc. 5th International World Wide Web Conference*, 1996.

[10] J. E. Pitkow & R. K. Jones, "Supporting the Web: A Distributed Hyperlink Database System," in: *Proc. 5th International World Wide Web Conference*, 1996.

[11] S. B. Achelis, *Technical Analysis from A to Z*, Probus Publishing, 1995.