Modeling Interactive Memex-like Applications Based on Self-Modifiable Petri Nets

Sheng-Uei Guan[1] and Wei Liu

Department of Electrical and Computer Engineering
National University of Singapore
10 Kent Ridge Crescent
Singapore 119260
Email: sg_1_1@yahoo.com\eleliuw@nus.edu.sg

Abstract This paper introduces an interactive Memex-like application using a self-modifiable Petri Net model – Self-modifiable Color Petri Net (SCPN). The Memex ("memory extender") device proposed by Vannevar Bush in 1945 focused on the problems of "locating relevant information in the published records and recording how that information is intellectually connected." The important features of Memex include associative indexing and retrieval. In this paper, the self-modifiable functions of SCPN are used to achieve trail recording and retrieval. A place in SCPN represents a website and an arc indicates the trail direction. Each time when a new website is visited, a place corresponding to this website will be added. After a trail is built, users can use it to retrieve the websites they have visited. Besides, useful user interactions are supported by SCPN to achieve Memex functions. The types of user interactions include: forward, backward, history, search, etc. A simulator has been built to demonstrate that the SCPN model can realize Memex functions. Petri net instances can be designed to model *trail record, back,* and *forward* operations using this simulator. Furthermore, a client-server based application system has been built. Using this system, a user can surf online and record his surfing history on the server according to different topics and share them with other users.

Keywords Memex, Petri net, color token, trail, history retrieval

[1] Corresponding Author: TEL: 65-68745153; FAX: 65-67791103

1. Introduction

As early as 1945, Vannevar Bush proposed a desktop personal information machine called the Memex (memory extender) [2]. Memex focused on the problems of "locating relevant information in the published records and recording how that information is intellectually connected." Memex later became an influential idea and by the 1980s it was hailed as the inspiration for hypertext and new ways to organize and retrieve information. Memex would archive, analyze, and index our browsing or retrieval experience, covering books, movies, conversation, etc.

With the development of the Internet, more and more computers are network based. A great deal of information is on-line. The sites related to your interest will often be visited by you. Maybe we will hope that the browser can record our visit trails and help us quickly find sites visited recently. Although bookmarks can be used to record frequented websites, browsers discard most history and trail information. The explosion of information needs a more effective mechanism. Memex has been considered in this domain. Assisted by Memex, a Web surfer can retrieve the URL trails that a user visited several months ago. In this paper, we propose a mechanism Self-modifiable Color Petri Net - SCPN to simulate the Memex functions in a Web browser. In this mechanism, an SCPN instance is used to record a trail of a topic, a place in an SCPN instance represents a website. To record trails according to different taxonomy, users are allowed to choose a website to be archived or a side-trail to be built.

The paper is organized as follows. Section 2 gives an introduction of the related work. Section 3 presents the definitions of SCPN. Section 4 gives solid examples on how SCPN is used to realize Memex functions. Section 5 describes the implementation of an SCPN simulator. Section 6 introduces the client-server based application system. Section 7 discusses some design issues, while Section 8 concludes this paper.

2. Related Work

2.1 Petri Net Related

Petri Net

Petri net is a graphical notation for the formal description of systems whose dynamics are characterized by concurrency, synchronization, mutual exclusion, and other conflict, which are typical in distributed environment. A formal definition of Petri nets is a four-tuple (P, T, I, O)[1] where P is a set of places that are the state variables of a system; T is a set of transitions, which are state changing operators. I and O are the pre- and post-conditions of a transition. A simple example of a Petri net model is shown in Figure 1. The dynamic performance of a Petri net is controlled by the firing rule. A transition (e.g. t_1 as shown in Figure 1) will fire if all its input places contain at least one token.

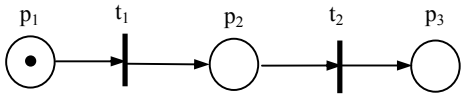


Figure 1. A Petri Net Example

Several extended Petri net models have been proposed to extend its application domains. In the following, we describe some of these models.

Object Composition Petri Net (OCPN)

The OCPN [9] model proposed by Little and Ghafoor is an enhanced version of Timed Petri nets. It introduces the value of time as duration to the conventional Petri nets. OCPN can model temporal relations between media data in multimedia presentation. As a drawback, this model cannot handle user interrupt and distributed environment.

Enhanced Prioritized Petri Net (EP-net)

Proposed by S.U. Guan and S.S. Lim, Enhanced Prioritized Petri Net (EP-net) [4,8] is an enhanced version of P-net [10]. EP-net uses a mechanism known as Premature/Late Arriving Token Handler to handle late and/or premature tokens (locked tokens forced to unlock). Moreover, EP-net introduces dynamic arcs associated with sets of program statements to handle user interactions, this improves the flexibility of designing interactive systems. Yet when handling user interaction, this model needs to replicate the Petri net being modeled every time an interaction (e.g. reverse) occurs, this makes the model size large and difficult to analyze when the network size increases.

2.2 Web Surfing Correlation

Memex

Proposed by Vannevar Bush, Memex [2] is a recording device that has some functions of a computer, microfiche, database and an information retrieval system. The essential feature of Memex is its capacity for retrieval and annotation. Another important feature of Memex is the function of associative indexing that presents the feature of hyperlinks. In addition to these links, Bush also wanted Memex to support the building of trails through the material in the form of a set of links that would combine information of relevance for a specific topic. Using Memex to support Web surfing can be more effective and powerful.

A Personal Web Map

SeiJi Yamada and Norikatsu Nagino proposed a database named Personal Web Map (PWM) [7] and developed the Anytime-Control algorithm to let users control their own web map construction of their favorite websites. PWM divides web pages in layered sets and provides two different user modes. The building mode allows a user to construct a PWM to record the URL information according to the keyword that he is interested in. The utilizing mode lets the user browse the database to retrieve

information. PWM can help users to gather relevant information in the WWW to a small database for convenient retrieval. It is more like a powerful bookmark but it does not record users' browsing trails.

Bookmark Organizer and PowerBookmarks

Bookmark Organizer [5] is a useful client-side bookmark organization. It achieves automatic classification by using clustering analysis to organize documents based on their conceptual similarity and at the same time allows the user to select when and where to apply it. Although this approach is useful for personal organization, it does not record user surfing context details, which are essential in reconstructing trail history.

PowerBookmarks [6] provides personalized organization and management of bookmarks by combining database with Web technologies. PowerBookmarks provides the control functions to navigation and bookmarking through a CGI form interface. It can achieve advanced query, classification and navigation functions and classify bookmarks. Although PowerBookmarks can store specific information for different users, it couldn't record surfing trails like Memex.

Using Memex as a Browser Assistant

Soumen and Sandeep et al. presented a beginning work of Memex for the Web [3]. In their work, Memex is used as a browser assistant. Data can be indexed not only by keywords but also according to the user's view of topics. This mechanism has a client-server architecture. The server side performs various archiving and mining functions and the client side provides personal folders to record the trails in different topics and allows the user to choose to archive and share trails.

Related work so far all uses ad hoc approaches constructing Memex-like applications, in contrast, our approach offers an underlying model with which a systematic approach to constructing Memex-like applications can be adopted. Most browsers implement incomplete history trails by stacking URLs visited, which means trails can be lost once popped off the stack. In contrast, our approach enables

history & trail retrieval as many times as you wish as it is based on the Self-modifiable Color Petri net model that also facilitates many types of user interactions. So the URLs recorded will not be lost unless the user deletes them. Therefore it is more powerful.

CZWeb

CZWeb [11] consists of a hierarchically organized network (or graph). A map-like visualization tool is provided to represent the intricate interconnection among pages. Rectangle is used to represent node in the network. There are two types of nodes are supported, page node and cluster node, which represent a particular web page and a group of web pages separately. CZWeb supports non-metric relationship through a hierarchical network representation of web pages. Although CZWeb provides a useful interconnection among pages, no surfing history information is involved.

Memoir

Memoir [12] is an open framework, it is extensible and adaptable to an organization's infrastructure and applications and provides its interface via standard Web browser. Trails are used to open hypermedia link services and a set of software agents to assist users in accessing and navigating vast amounts of information in Intranet environments. The trails in Memoir are mainly used to record actions on documents that users have visited. In our Memex application, trails are mainly used to record and retrieve surfing history information.

3. Self-modifiable Color Petri Net (SCPN)

In this section, we introduce the general concepts and definitions of Self-modifiable Color Petri Nets (SCPN).

There are two types of tokens in SCPN: color tokens and resource tokens. And resource tokens are divided into two sub-types: forward tokens that move in the same direction with arcs and reverse tokens that move in the opposite direction with arcs.

Definition 1: SCPN

SCPN is a seven-tuple S=(P, T, C, A, D, M, U), where P∩T=Ø.

P= $\{p_1, p_2, p_3, \dots, p_m\}$ is a finite set of places, where m>0.

T= $\{t_1, t_2, t_3, \dots, t_n\}$ is a finite set of transitions, where n>0.

C= $\{c_1, c_2, c_3, \dots, c_k\}$ is a finite set of commands (as defined in table 1), where k>0 (e.g. c_1= "create an arc", c_2= "delete an arc").

A: {PXT} U {TXP} is a finite set of arcs representing the flow relation.

D: P $\rightarrow$ R^+ is a mapping from the sets of places to the set of non-negative real numbers, representing the presentation duration of the resources concerned.

M: P $\rightarrow$ $\{I_r^+, I_c^+\}$ (I_r—counts the number of resource tokens; I_c—counts the number of color tokens). $I_{c/r}^+$ ={0, 1, 2,...}. M is a mapping from the set of places to the set of integer numbers, representing a marking of a net.

U: is a finite set of colors, each color represents some types of user interaction, which may reconfigure the Petri net being modeled. This is done by associating commands (Table 1) with each color token. There are two approaches to represent commands using color tokens: one is to use one color token to represent each different type of (user) interrupt, thus a color token represents a combination of several basic commands. Another is to use one color token to represent one command, and the color of a token also indicates the priority of the command represented (i.e. which command will be executed first). The former approach is closer to the reality, we have adopted the former approach in this paper. Note that for the ease of presentation, no special color coding is prescribed for each command, colors in this paper are just used to indicate that there are different commands in the system. In practical use, a 'color code' is used to encode each color token (and user interrupt), when a user interaction occurs, the corresponding 'color code' will be generated and the program related with this code will be executed to finish the particular user interaction. In the

figures that follow, we use different shapes to represent color tokens to make it easier to recognize when printed in black and white.

The list of commands shown in Table 1 are sufficient for any types of change to a Petri net because the basic modifications to Petri nets have been included in our list (the commands labeled with ∗).

Table 1 List of commands

Mechanism	Command	Action
Arc	Create arc∗	To create an arc
	Delete arc∗	To delete an arc
Place	Create place∗	To create a place
	Delete place∗	To delete a place
Transition	Enable transition	A transition is able to fire when the conditions to fire are met
	Disable transition	A transition is not able to fire regardless whether the conditions to fire the transition are fulfilled
	Create transition∗	To create a transition
	Delete transition∗	To delete a transition
Token	Lock token	To lock a token
	Unlock token	To unlock a token
	Reverse	Change the direction of the resource token
	Create token∗	To create a token in the indicated place
	Delete token∗	To delete a token

By using these commands, it is easy to change a Petri net model to fit with different user requests. These changes are simple such as adding new places, transitions and arcs. To finish a user interaction, generally we will not need to increase the model's size much so as to increase adversely the time and space complexities. In comparison, if we use the OCPN or EP-net to handle a reverse operation, the 'forward' part of the model needs to be replicated in the reverse direction to finish the operation, this makes the model too large and complex. But using the SCPN model, we just need to insert the reverse color token, and the resource token will then move in the reverse direction, no model structure needs to be changed.

Definition 2: Firing rules of SCPN

The firing rules of Petri nets are: a transition is enabled when all its input places contain some number of tokens greater than or equal to the number of each respective place's arcs to transition. If the condition is met, the transition can fire and if the transition fires, token(s) are moved from their input places and created at each of its output places.

By introducing some novel mechanisms, SCPN can handle user interactions flexibly. For the new mechanisms to work, some new rules are defined to assist SCPN to complete its functions:

- A color token will be injected into each place that contains resource token(s) when a user interaction occurs.
- When a color token is injected, the execution of the model will be interrupted.
- When all the commands associated with a color token have been executed, this color token will be deleted. Then the playback of resource tokens will be resumed.

The commands associated with each color token can be designed according to the corresponding user interaction. In the following, we use some solid examples to demonstrate how color tokens are used to realize Memex functions in Web surfing.

4. Designing Memex Functions using SCPN

The characteristics of Petri net make it straightforward to record a trail. Given a fixed trail, Petri net can describe a trail visually by using places to represent nodes along the trail while using transitions and arcs to indicate the direction of trail. However, Petri net configuration once prescribed cannot be changed during run-time which makes it difficult to record a dynamic trail. By adding the self-reconfiguration function, SCPN can be used to record dynamic trails. So in this paper, we use SCPN to design and simulate Memex functions in Web applications. It allows users to create trail records and inquire about trail history.

4.1. Simulating Memex Trail Recording in Web Browsing

To simulate Memex in Web browsing, we assume that a place in SCPN represents a website. Each time when a website is opened, a color token including the following basic commands will be injected into the place p_{start} that includes a resource token as shown in Figure 2: lock the resource token in p_{start}, create a new place p_1 (this place will represent the newly opened website), create a new transition t_1, create an arc from current place p_{start} to the new transition t_1, create an arc from the new transition t_1 to the new place p_1, unlock the resource token in $p_{start.}$ Finally, the color token self-deletes, transition t_1 fires, the resource token moves to p_1 indicating that the website represented by this place is active now. While SCPN is recording the surfing trail, the corresponding website address will be recorded along with each place.

a. Create a new place and transition b. Transition t_1 fires

Figure 2. An Example - Using SCPN to Record the Surfing Trail

4.2. Main Trail and Side Trails

Almost all websites contain some related hyperlinks. A trail can bifurcate: when a hyperlink of one website is visited, a side-trail will be created to record it. As shown in Figure 3, the main trail that represents the main surfing history is composed by places with *m* as the first subscript, the side-trail that represents the hyperlink of a website is composed by places with names having *s* as the first subscript.

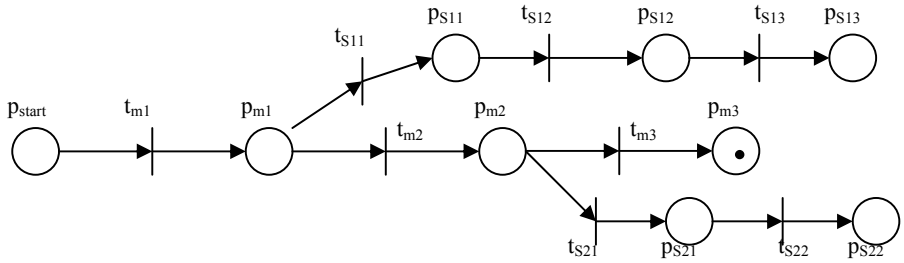


Figure 3. Trail Recording Using SCPN

If the hyperlink is opened in a new window or the user wants to record the hyperlink of a website as a new trail, a new starting place will be created as the first place in a new trail as shown in Figure 4. The arcs linking from p_{m1} to $p_{m'1}$ are represented by dot lines meaning that these arcs do not allow a reverse token moving along them.

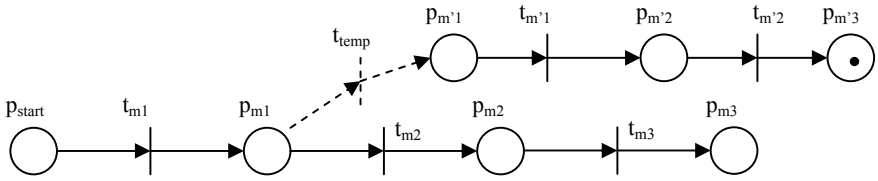
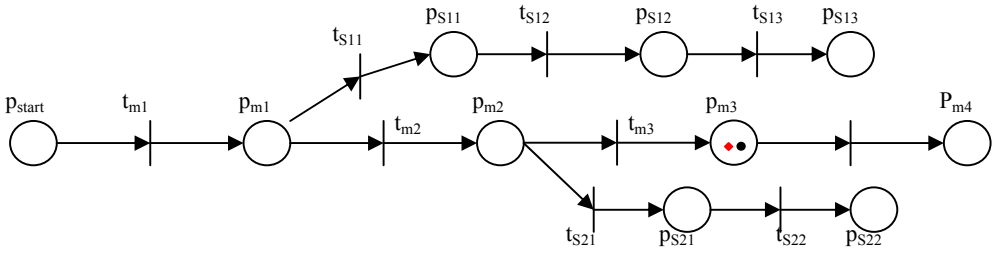


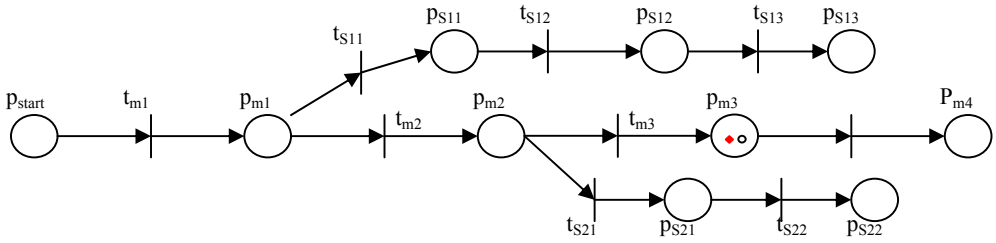
Figure 4. A New Trail is Created

4.3. Backward and Forward Operations

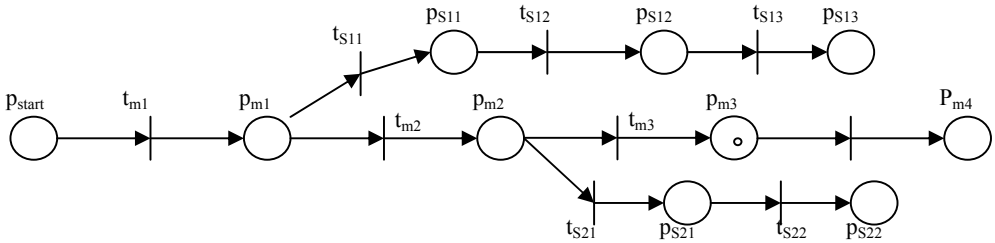
Using SCPN to record a browser trail, it can simulate the *backward* and *forward* operations of Web browsing. A resource token in a place indicates that the website corresponding to this place is active, the arcs indicate the sequence of websites being visited. When a user issues a *backward* command, a color token corresponding to this command will be injected into the place p_{m3} that includes a resource token as shown in Figure 5a. Then the commands associated with this color token executes, the resource token in p_{m4} is locked and changed to a reverse one as shown in Figure 5b. In Figure 5c, the reverse token is unlocked and the color token self-deletes. Finally, transition t_{m3} fires, the reverse token moves from p_{m3} to p_{m2} and changes back to forward resource token as shown in Figure 5d, the information of the website related to place p_{m2} will be retrieved. At the same time, p_{m3} is recorded as an exit point so that a future forward move will allow p_{m3} to be revisited.



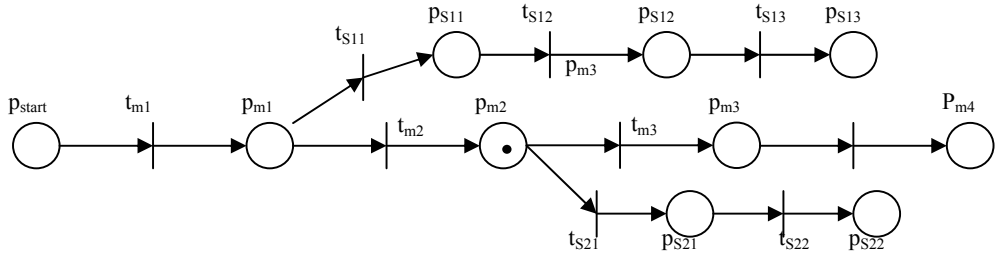
a. A color token corresponding to the backward operation is injected into p_{m3}.



b. The resource token is locked and changed to a reverse one.



c. Reverse token is unlocked and color token self-deletes.

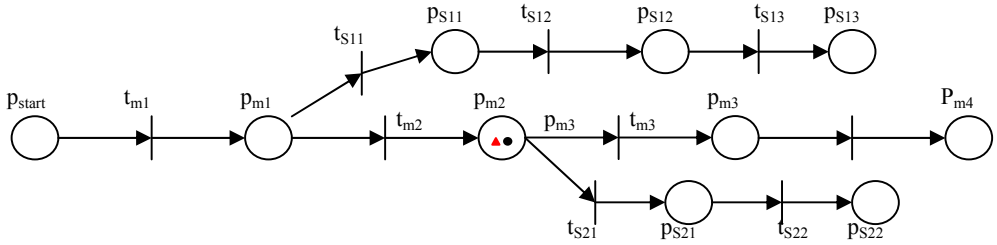


d. Execution of the backward operation is completed.

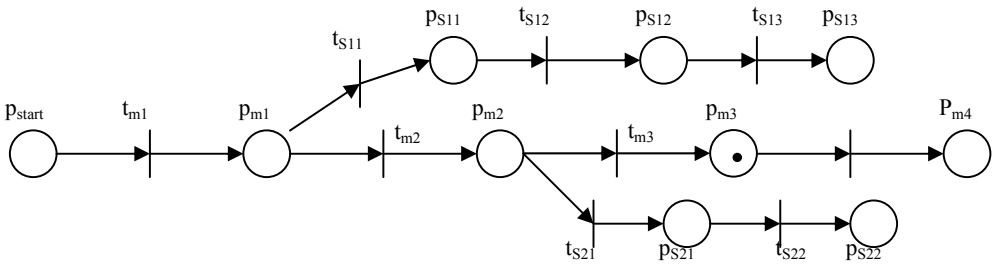
Figure 5. Implementation of the Backward Operation

After checking the content of this website, if the user decides to go back to the previous website again, a *forward* command can be issued. A color token (the red token in triangle form in Figure 6a just indicates that this is a token associated with the *forward* command) associated with the *forward* command will be injected into place p_{m2} that contains the resource token as shown in Figure 6a. Then the command executes to direct the resource token to fire. At this moment, we can see that one of the two transitions t_{m3} and t_{s21} can fire. In modeling Memex functions, SCPN is used to record the surfing history, the resource token is used to indicate the active website. There can be only one place that can

contain the resource token at a time. In such a *forward* operation, because the exit point of a previous *backward* operation has been recorded, t_{m3} will fire and the resource token will move to p_{m3} as shown in Figure 6b, at the same time, the record of the previous exit point will be replaced by p_{m2} for future use.



a. A color token associated with the forward operation is injected into p_{m2}.



b. The forward operation executes.

Figure 6. Implementation of the Forward Operation

5. Simulator

Using Visual C++, a simulator has been built. This simulator can model Memex functions such as trail recording and retrieval. In this simulator, we use places in SCPN to represent the websites visited. An SCPN instance represents a trail. To make the simulation more realistic, we use the Microsoft Active X® controller in our program to display a website visited at the same time when the SCPN place corresponding to the website is created or a resource token is injected into the place. A user can click the buttons as shown in Figure 7 to simulate the corresponding function.

To make the simulator more powerful, a basic Petri net design tool is provided. A Petri net instance can be designed simply by clicking and dragging the icons from the toolbar to the white area. The Petri net instance created can be saved as a .mex file for future use. Also a *RUN* button as shown in the menu in Figure 7 is provided to execute a Petri net instance. When the *RUN* button is clicked, the Petri net instance will be executed. If the instance is active, the token will move according to the firing direction. The *Back, Forward* and *History* buttons are used to simulate Memex functions in a Web Browser. The *Save* button is used to save the trail. If some trails have been built, *Search* function can help a user to find an item of interest in these trails. We give an example to show how this simulator works.

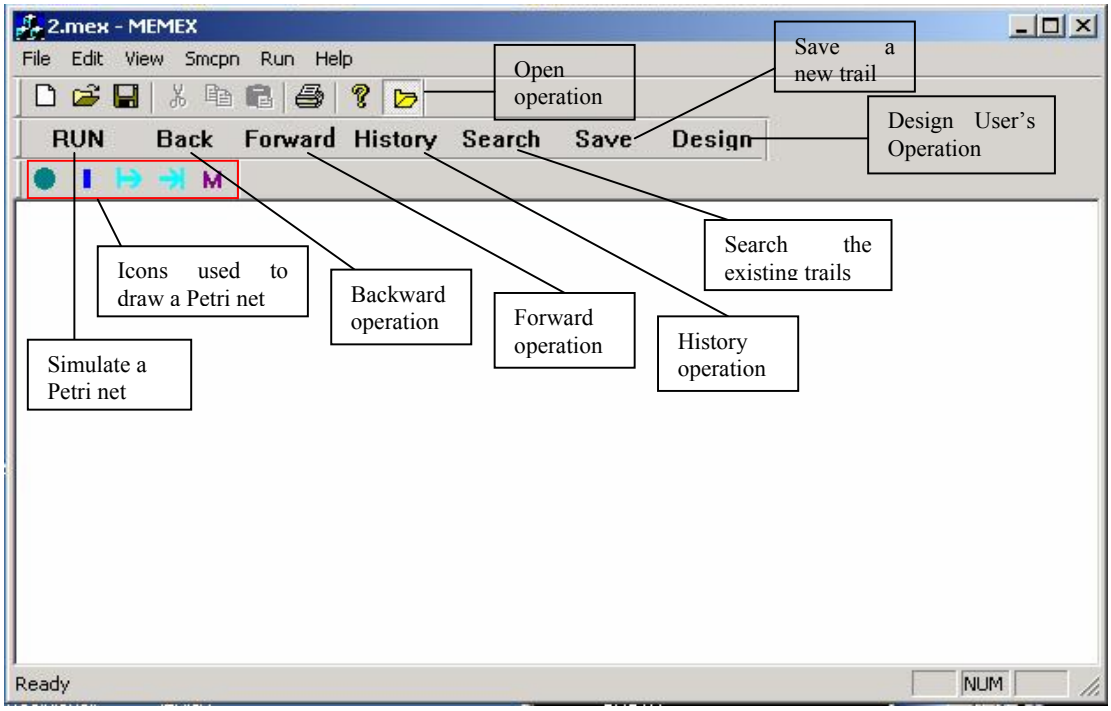


Figure 7. The User Interface of the Memex Simulator

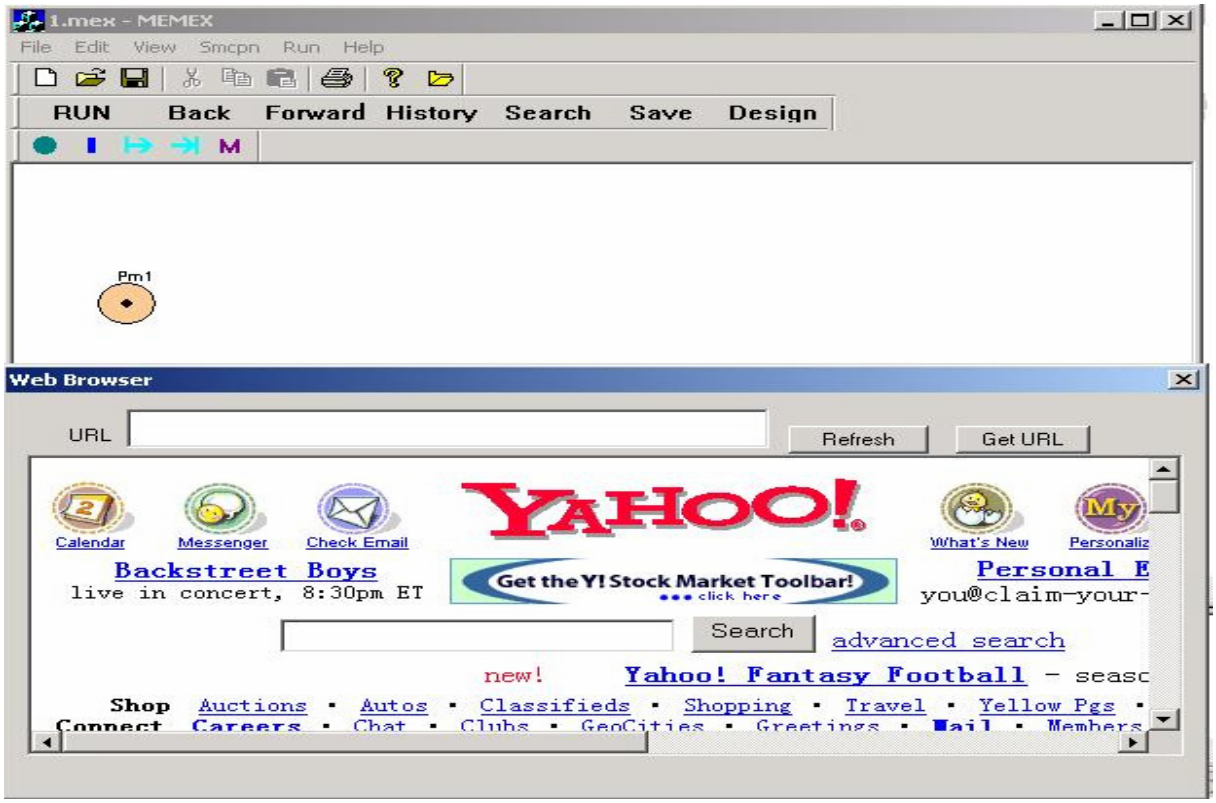


Figure 8. A Place Created to Record the Website Being Opened

As shown in Figure 8, when a website is opened (assume this is a new trail to be built), an event signal will be sent to the system indicating a new website is opened. With this event, a place will be created to record it. And a resource token will be created in the place at the same time to indicate that the website corresponding to this place is active. In order to let the user arrange trails according to his need, the simulator provides trail recording options. Each time when a website is opened, a dialog box will be popped up to ask the user if the website needs to be recorded as shown in Figure 9. If the user chooses not to archive this website, the place being created to record this website will be deleted after the website is closed. If the user puts down an existing trail name, the website will be added and recorded as the last place in this existing trail. If the user puts down a new trail name, a dialog box will be popped up to let the user choose how to record this website as shown in Figure 10. For example, if a hyperlink is followed after three websites have been visited, the user chooses to record it as a side trail

by clicking the *Yes* button (Figure 10). This website will then be recorded as a side trail as shown in Figure 11.

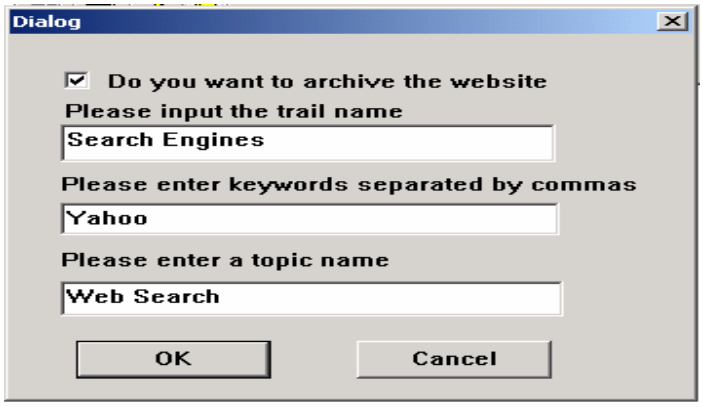


Figure 9. Archiving Choice Dialog Box

Figure 10. Trail Creation Choice Dialog Box

As shown in Figure 12, there are five places in the SCPN instance shown. From this we know that five websites have been visited. The active website is http://www.google.com associated with the fifth place. SCPN can show how many websites have been visited and which one is active now, but no detailed information of these websites is shown on the graph. If the user wants to see the details of the websites visited, the *History* button in the menu can accomplish this task.

Using SCPN to record trails, each place is associated with a website. It is easy to display history records. When the user issues a command '*History'*, this can be done by clicking on the *History* button, a dialog box will be opened to show the detailed trail information as shown in Figure 12.

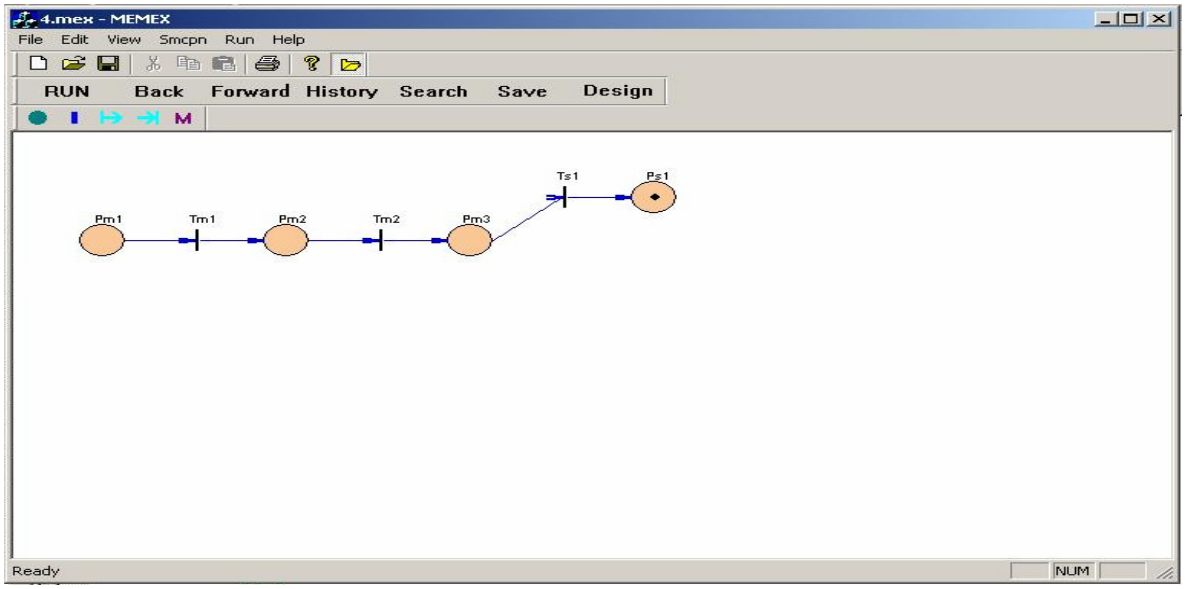


Figure 11. A Website Recorded as a Side Trail

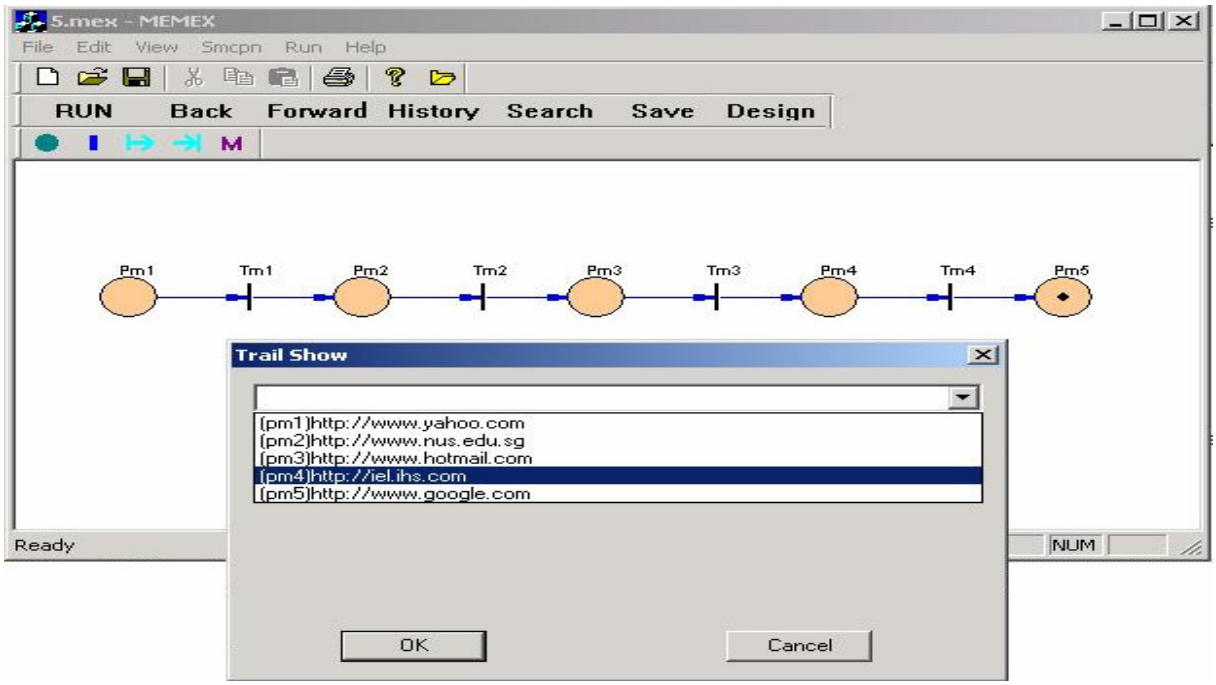


Figure 12. The History Information Displayed

With the trail shown, we can select any item to revisit. For example, if we want to visit the IEEE Xplore website, just select it from the list and click the *ok* button. The corresponding website will be retrieved and the resource token will move to p_{m4} as shown in Figure 13.

In addition to trail recording and retrieval, the Memex simulator can also achieve the *backward* and *forward* operations similar to those functions in web browsers. As shown in Figure 14, if the user wants to visit the previous website before the IEEE Xplore website, he only needs to click the *Back* button. The resource token will move to p_{m3} and at the same time the website associated with this place will be opened.

Following the above example, if the user wants to visit the next website again, he only needs to click the *Forward* icon, the resource token will move to p_{m4} and the corresponding website will be reopened at the same time.

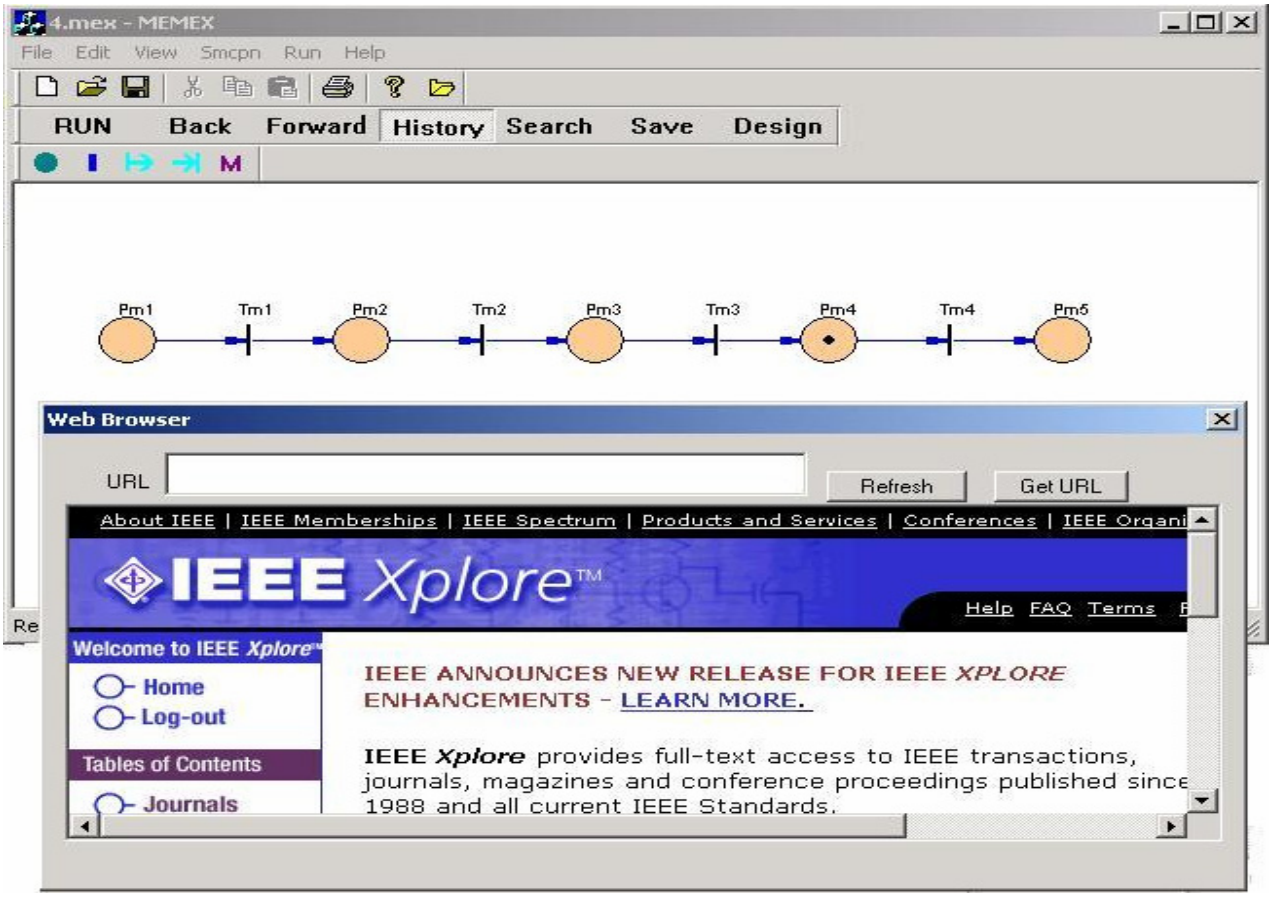


Figure 13. Website Represented by p_{m4} Retrieved

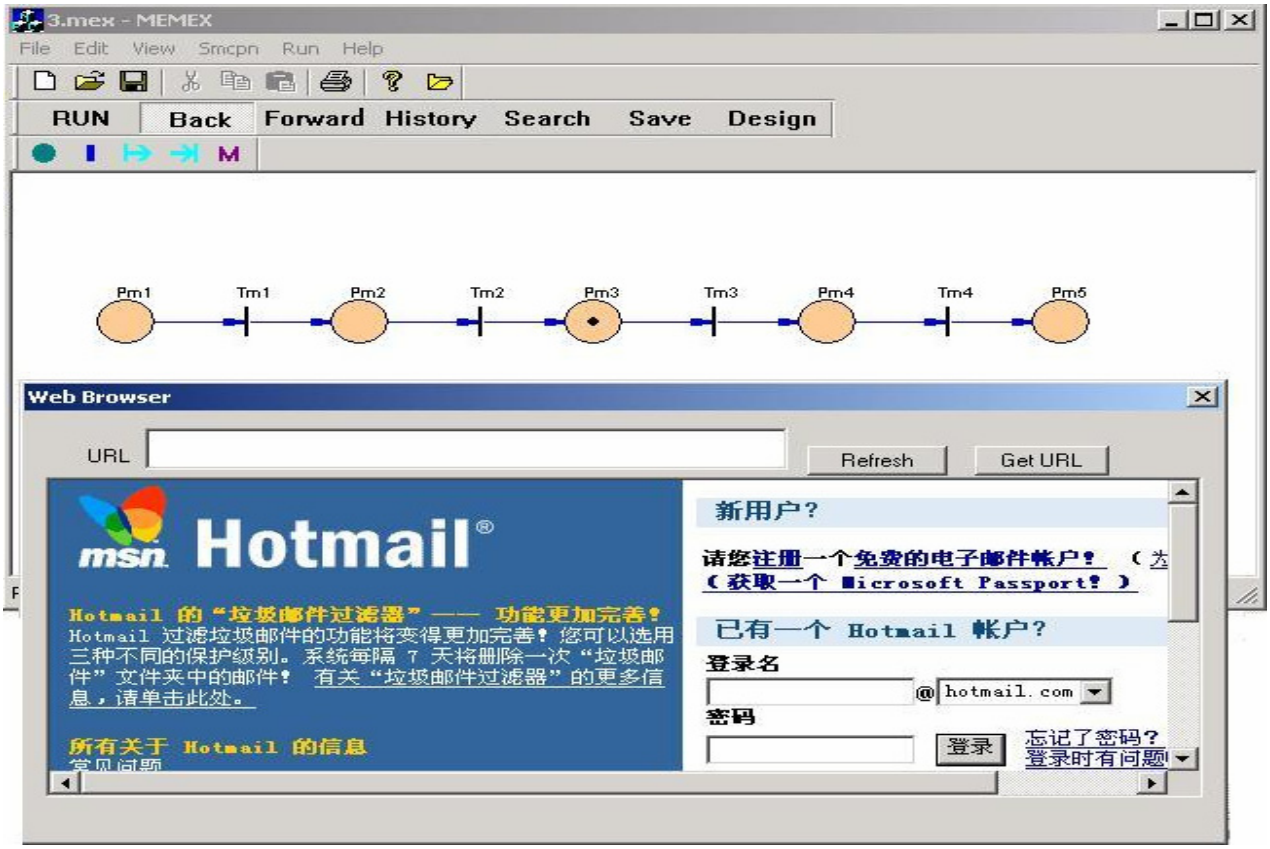


Figure 14. The Backward Operation Executed

Besides these web-browser-like operations, the most important Memex function is that when some trails have been built, a user can search for it according to name/topic/keyword. As shown in Figure 15, when a user clicks the *Search* button, a dialog box will be popped up to show the existing trails. Then the user can select from these trails the one that he is interested in to retrieve or input it in the search Edit-box. For example, if the user wants to find some information about Memex, he/she only needs to select the first item from the dialog box or input Memex into the search Edit-box and click the *ok* button. The Memex trail details will then be displayed in a dialog box. Then the user proceeds to choose a website he wants to visit from this trail. Assume that the first one is selected, the website will be opened and at the same time the trail represented by SCPN is displayed as shown in Figure 16. The resource token in p_{m1} indicates that the website associated with this place is active.

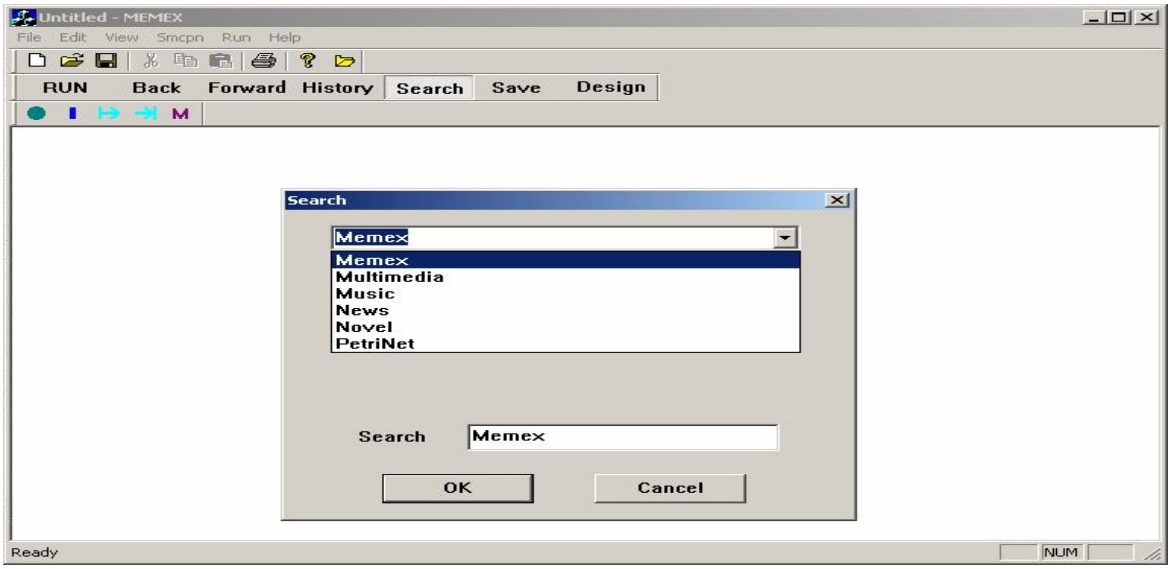


Figure 15. Search for Existing Trails by Name

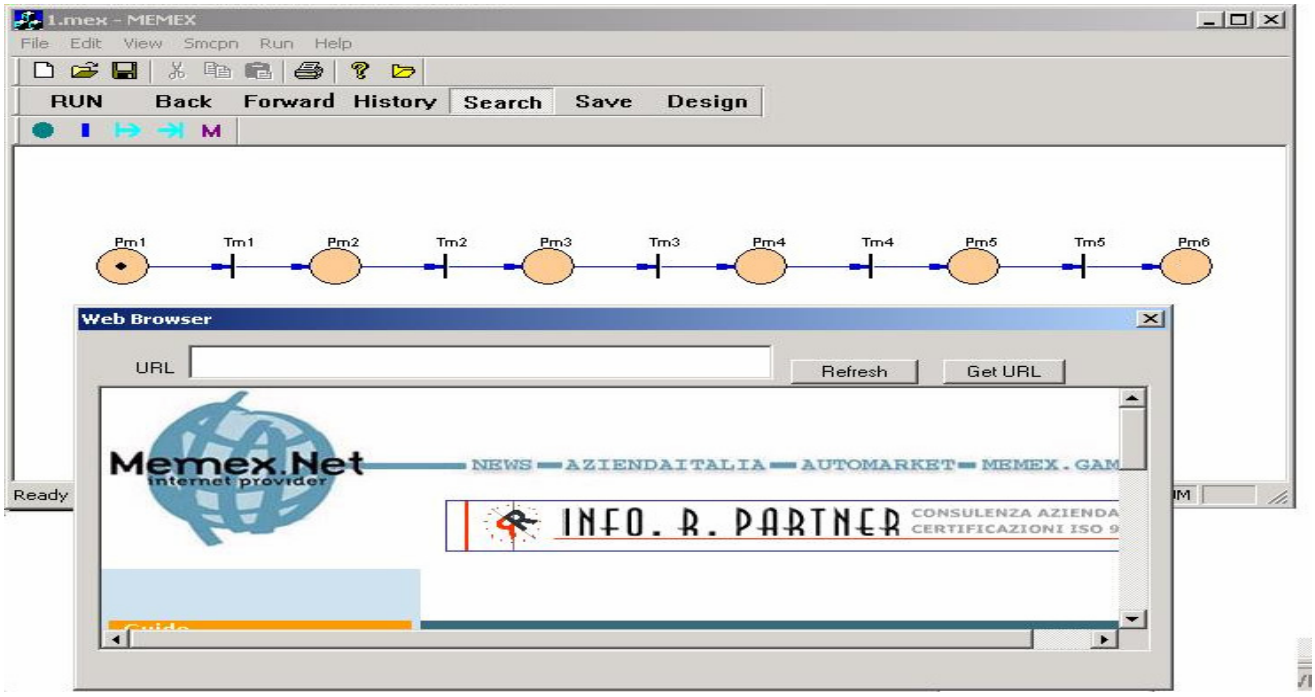


Figure 16. The First Website of the Memex Trail Retrieved

By introducing color tokens and a basic command set, SCPN allows users to design their own operations. Using *Trail Record* as an example, we show in the following how a user can design an operation. If a user wants to design his own operation, he only needs to click the *Design* button, a dialog

box will be popped up as shown in Figure 17. Basic commands have been listed in the left part of the dialog box to help the user finish his design easily. To achieve *Trail Record*, a new place and a link from the existing active place to this place will be created. The user then selects commands 2, 1, 4 and 3 to achieve this operation. As shown in Figure 17, the user puts down the corresponding numbers of these basic commands according to the execution sequence. He then clicks the *ok* button to finish the design of *Trail Record* operation.

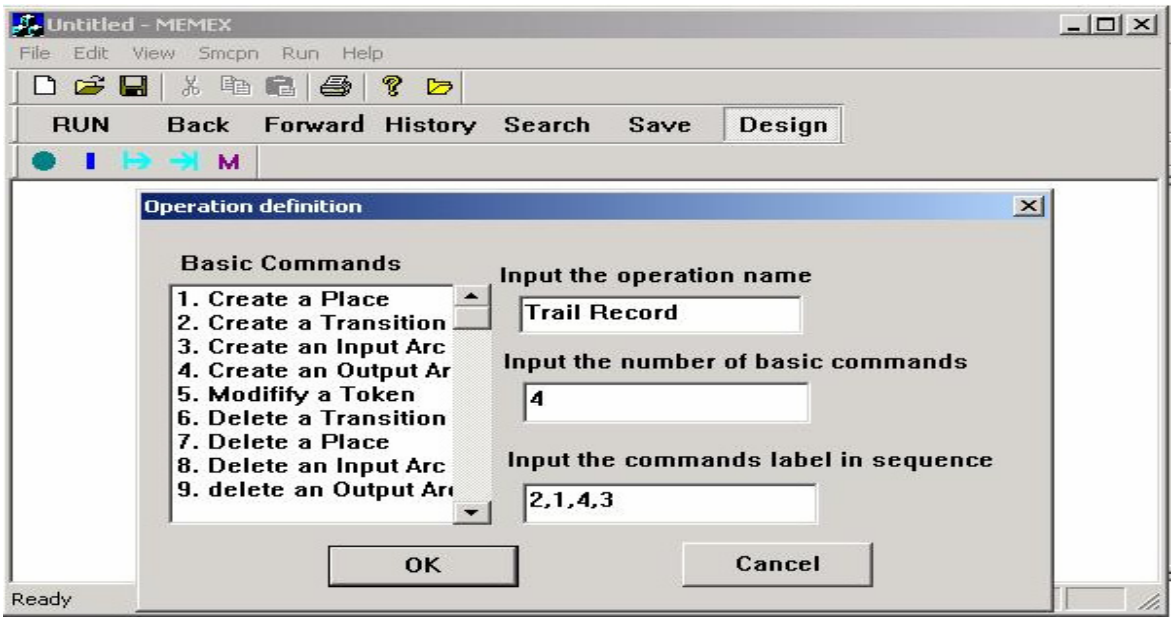


Figure 17. Using SCPN to Design an Operation

6. The Client-Server Based Application System

A Memex-like application system is built upon the architecture of client-server. Users can surf online using the embedded web-browser, designate history topics and request to store relevant surfing history on the server side. Additionally, the user can share the resources stored in the server with others, and add or delete items under the topics with most interest. The application utilizes the TCP/IP protocol within the Win32 (Windows 32-bit) environment to communicate with a multimedia resource server.

Based on the Windows 2000 platform, the application is running under the National University of Singapore's intranet.

6.1 Implementation of the Server

A Memex server is used to store trails, users can store and view their surfing trails in the server. Trails, which are meant to be shared by the other users, will be viewable by others. Before the server can be used by a client, some initialization needs to take place. Initialization of the server refers to the steps taken to establish connections with the client before data transfer. Figure 18 shows a Petri net for the server initialization. The server starts off in the listening mode, which enables the server to constantly listen for any attempts by a client connecting to the server. In the beginning, a locked token is in the place of p_{listen}. If a client requests a certain type of service from the server, the server will try to build up a connection with this client. Once the connection is built, the token in p_{listen} is unlocked and moves to $p_{connect}$, in the meanwhile, an acknowledgement signal is sent to the client. As shown in Figure 18, a token is created in $p_{connect}$ when a connection is built. Then the server will wait for the client's requests and handle them.

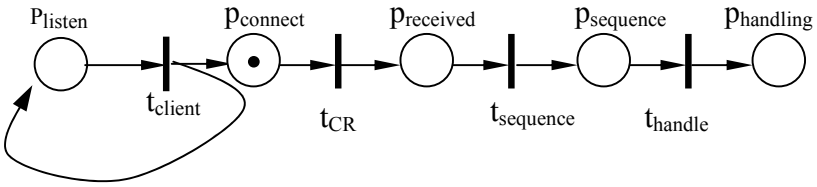


Figure 18. SCPN for Server Initialization

Figure 19 is the screen shot of a Memex server. All available trails in the server are listed under the "source list" with their topics as shown in Figure 19. The Send and Receive list boxes are used to show the messages received and sent. If the address and topic for a website are received (e.g. the webpage address of http://www.cs.brown.edu/memex/ and the topic of "Memex" are received), the server will add this website address to the end of the trail named as Memex. If a new topic is received (such as

Software), the server will open a new trail named as this topic and store the corresponding website address under this trail.

Figure 19. The Screenshot of the Memex Server

6.2 Trails Recording, Organizing and Retrieving in the Memex Server

Trail management is the most important part in designing the Memex server, issues need to be resolved such as how to store the trails and related information, and how to respond to and handle the clients' requests, etc. Selection of a proper database management system is crucial.

A users' surfing history will be stored in the server as trails under specified topics. The user can manage these trails by issuing corresponding commands to server. For example, if the user issues a trail store request (this request includes three parts: a web address, a topic and the user's information) to the server, the server will check the userID and password firstly. After verification, the topic will be taken out to see whether the same topic has already existed on the server. If it exists, the server will check the user's authorization to see whether he is allowed to change the trail. Although all users can access the Memex server, not all users can modify the trails stored in the server. Different users have different access rights to different trails. If the change is allowed, the server will extract the web-address and

append it to the end of the trail. Otherwise, the server will return a message to the client to inform that the topic exists and no change is allowed. If this is a new topic, the server will store the web-address under this new topic.

Any user can access shared trails on the server. However, only authorized users can change existing trails in the server.

To retrieve a trail, a client needs to send a trail retrieve request with the topic name to the server. The server will follow this request and search in its database for a match, then the matched list will be sent back to the client.

Note that our SCPN does allow an automated implementation which means no user intervention is required. The mechanism of accessing a certain user's surfing history based on verifying the user ID and password can be disabled if privacy is not a concern.

6.3 Implementation of the Memex Client

The Memex web-browser is a Microsoft Windows-based application and the screenshot is shown in Figure 20. The application was developed mainly in C++ under the Microsoft Visual C++ environment. The Microsoft Foundation Classes (MFC) were used extensively throughout the implementation as they provided much of the functionality demanded.

An ActiveX object is embedded in this tool to achieve the browser function.

The browser can be used without a connection with the server. If a user wants to get information from the server or store his own information in the server, a connection must be set up first. This can be done by clicking the connect button (button 1 as shown in Figure 20). To use the Memex browser, simply input the web-address in the URL edit box and click the refresh button (button 7 in Figure 20). If the address is correct, it will be loaded. We elaborate how the Memex functions are realized in the following section.

6.4 Trail Recording and Sharing Using the Memex Server

Almost all web browsers provide some 'favorite' functions, users can save their favorite web-addresses. Sometimes users may also want to share good websites with each other. With our Memex browser, users can store their surfing history on the server and share them with other users. If a user wants to store a web address, he only needs to click the favorite button (button 6 in Figure 20), a new dialog box will be popped up for the user to input the needed information such as store topic. Upon receiving such a request, the server searches the topic name in its database. If it exists and the user has the authority to change it, the received web address will be appended to the trail with the same topic; otherwise, an illegal operation message will be sent to the client. If it doesn't exist, a new trail will be generated to record this web-address for the user.

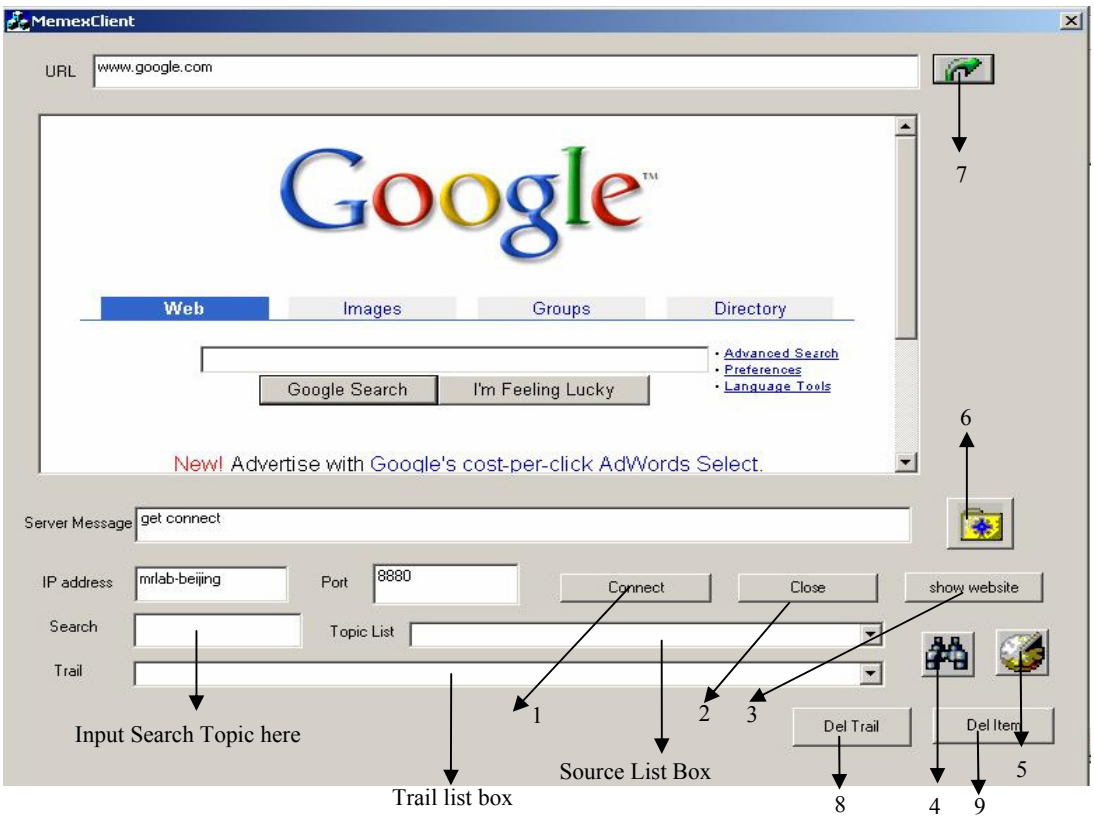


Figure 20. Screenshot the Memex Browser

At the same time, the user can decide if he wants to share this trail with other users. If the user has chosen to share this trail, the choice will be sent to the server to be handled.

If the user is not authorized to change a named trail, the system allows the user to store the web address as a side trail under the same topic. A dialog box as shown in Figure 21 will be popped up, allowing the user to create a side trail.

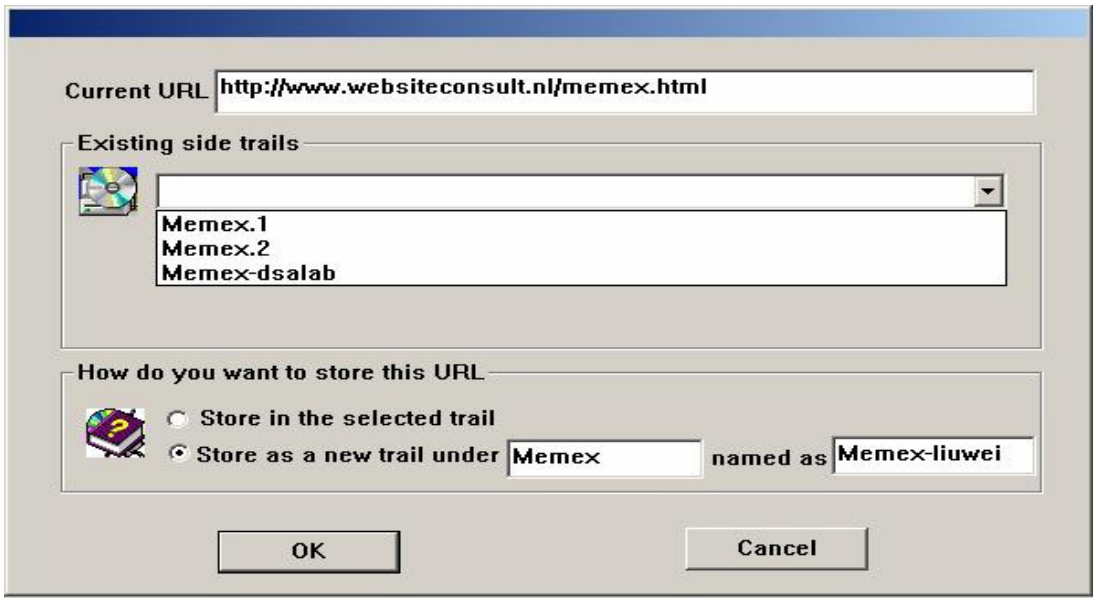


Figure 21. Dialog Box for the User to Create a Side Trail

6.5 Trail Retrieving from the Server

Search function is provided in the Memex server to help users to find their interested trails. To achieve this function, trail topics on the server are stored in a database. When the search function is requested, server will search in this database to look for the requested trail. A user can visit any shared trail stored on the server. Relevant trails can be retrieved from the server. Those trails labeled with '*' are shared.

With the trail list obtained from the server, the user can select a topic from the list to open . Exemplified in Figure 22, the topic of "Memex" is selected and this trail is opened as shown in Figure

22 by clicking the history button (button 5 in Figure 20). Web addresses belonging to this topic are shown in the underlying “Trail list box”.

Figure 22. Trail Display in a Memex Client

After a trail is retrieved, the user can visit it using the Memex browser. By simply clicking “show website” (button 3 in Figure 20) button, the selected web address will be opened.

6.6 Trail Removing and Updating

How to build and retrieve a trail using the Memex server-client system has been covered in the earlier sections. Besides these mentioned features, this application also allows users to delete unwanted trails or delete items from a trail. For example, if a user wants to delete a trail named as PetriNet, he only needs to select this item from the trail list and click the “Del Trail” button (button 8 in Figure 20). The ‘Del’ request will be sent to the Memex server, if the user has the right to do this, the trail named as PetriNet will be deleted.

Items in a trail can also be deleted. For example a user decides to delete the second item from the "Memex" trail, so he selects this item from the trail and click "Del Item" button (button 9 in Figure 20) to send this request to the Memex server. When the server receives this request, it will check the user's right firstly. If the delete operation is allowed, the corresponding trail will be opened and the items to be deleted will be extracted from the received packet. Then the items in this trail will be compared until the matched item is found and deleted.

7. Discussions

In the SCPN model, resource tokens are divided into two types: forward tokens and reverse tokens. To distinguish between these two types, a property parameter is attached to each resource token. For a reverse operation, when several segments need to be reversed, the color token will not move together with the reverse token. However, a number n indicating the number of segments (places) to reverse will be attached to the reverse token. Each time when a segment is displayed, n will be decremented by 1. When n reaches zero, the reverse token will be changed to a forward one.

When a reverse operation is requested for a trail including side trails, to avoid confusion, the user will be prompted to specify whether only the current side-trail is to be reversed or the reverse operation will include the main trail.

8. Conclusion

In this paper, we have given an introduction to a Self-modifiable Color Petri net model - SCPN. With the powerful reconfiguration function offered from this model, Memex functions can be achieved in Web browsing. Our approach offers an underlying model with which a systematic approach to constructing Memex-like applications can be adopted. A simulator with friendly user interface has been

built to show how this can be achieved. This simulator can also be used as a Petri net design tool to help users to design and implement their own Self-modifiable Color Petri net instances. A client-server based system has also been built to realize the basic Memex functions.

Acknowledgement

The authors are grateful to the valuable comments from the editors and referees.

REFERENCES

[1] James L. Peterson, "Petri Net Theory and the Modeling of Systems", Prentice-Hall, 1981

[2] Vannevar Bush, "As We May Think", Atlantic Monthly, 176, pp. 101-108, 1, July 1945. Online at http://www.theatlantic.com/unbound/flashbks/computer/bushf.htm

[3] Soumen Chakrabarti, Sandeep Srivastave et al., "Using Memex to Archive and Mine Community Web Browsing Experience", Computer Networks, Vol. 33. Iss. 1-6; pp. 669-684, Jun. 2000. Online at http://www9.org/w9cdrom/98/98.html

[4] Sheng-Uei Guan and Sok-Seng Lim, "An Enhanced Prioritized Petri Net Model for Authoring Interactive Multimedia Applications", Proceedings the Second International Conference on Information, Communications & Signal Processing (ICICS'99), Singapore, pp. 7-10, Dec. 1999

[5] Y.S. Maarek and I.Z. Ben Shaul, "Automatically Organizing Bookmarks per Content", in Proceedings Fifth International World Wide Web Conference, Paris, May 1996 http://www5conf.inria.fr/fich_html/papers/P37/Overview.html

[6] Wen-Syan Li, Quoc Vu, D. Agrawal, Y. Hara, and H. Takano, "PowerBookmarks: A System for Personalizable Web Information Organization, Sharing and Management", Computer Networks, 31, May 1999

[7] Seiji Yamada, Norikatsu Nagino, "Constructing a Personal Web Map with Anytime-Control of Web Robots", CoopIS'99 Proceedings, IFCIS International Conference on Cooperative Information Systems, pp. 140-147, 1999

[8] Sheng-Uei Guan and Sok-Seng Lim, "Modeling Multimedia with Enhanced Prioritized Petri Nets", Computer Communications, Vol. 25, Issue 8, pp. 812-824, May. 2002

[9] Thomas D. C. Little, A. Ghafoor, "Synchronization and Storage Models for Multimedia Objects", IEEE Journal on Selected Area in Communication Vol. 8, No. 3, pp. 413-427, Apr. 1990

[10] Sheng-Uei Guan, Hsiao-Yeh Yu, and Jen-Shun Yang, "A Prioritized Petri Net Model and Its Application in Distributed Multimedia Systems", IEEE Transactions on Computers, Vol. 47, No. 4, pp. 477-481, Apr. 1998

[11] Fisher B., G. Agelidis, J. Dill, P. Tan, G. Collaud and C. Jones. "CZWeb: Fish-Eye Views for Visualizing the World-Wide Web", Proc. Seventh Int. Conf. on Human-Computer Interaction (HCI International '97), pp 719-722, 1997

[12] D.Derource, W.Hall, S.Reich, et al. "Memoir: An Open Framework for Enhanced Navigation of Distributed Information" in Information Processing & Management, V37, pp53-74, 2001

[13] Kurt Jensen, "Coloured Petri Nets", Vol. 1, Springer-Verlag, 1997

[14] Yahya Y. Al-Salqan and Carl K. Chang, "Temporal Relations and Synchronization Agents", IEEE Multimedia, Vol. 3, pp. 30 - 39, 1996

[15] D.C.A Bulterman, "SMIL 2.0.2. Examples and Comparisons", IEEE Multimedia, Vol. 9, Iss. 1, pp. 74 -84, Jan-Mar 2002