Modelling Web Navigation by Statechart

Karl R.P.H. Leung♠ Lucas C.K. Hui♡ S.M. Yiu♡ Ricky W.M. Tang♡
♠Department of Computing and Mathematics,
Hong Kong Institute of Vocational Education (Tsing Yi),
Hong Kong.
email: kleung@computer.org

♡Department of Computer Science and Information Systems,
The University of Hong Kong,
Hong Kong.
emails: {hui, smyiu, wmtang2}@csis.hku.hk

Abstract

There is a trend of increasing size of web sites and increasing complexity of web pages by dynamic content in recent years. Currently available web navigation modeling tools are unable to cope with the need of modeling these contemporary web sites, especially those with dynamic content. This need is analyzed in this paper and a web navigation model based on statechart is proposed to meet the requirements. The model will serve as a tool for the modeling and analyzing navigation of complex and dynamic web sites.

1 Introduction

The exponential growth of the World Wide Web and electronic commerce in recent years give rise to a need of modeling and analyzing the browsing semantics of web sites, which have increased in size and complexity. As in any other software development, a detailed model ease development and maintenance of the web site. In addition, the model can help in the analysis of navigation in the modeled web site. Undesirable situations can be identified and eliminated, and improvements can be made.

Modeling web navigation may not be as easy as it seems. Web pages has evolved from simple text, graphics and hyperlinks a few years ago to the contemporary dynamic web pages with varies extensions and scripts and client side programs embedded. Previous works on modeling web navigation [15, 11] treats the web as a kind of hyperdocument and can only support modeling of simple and static web pages. Complex, dynamic contemporary web pages, which are the most in need of modeling, cannot be modeled. In this paper, we will employ statechart [3] to solve this problem of modeling web navigation. As our aim is to model web navigation, presentation details of web, such as size, position and color of web page elements, although are also of importance to the human computer interaction, will be out of scope for this paper.

In Section 2, web navigation and modeling will be introduced, along with background information on Statechart. A detailed analysis on different kinds of web navigation will be presented in Section 3. Our model of web navigation will be discussed in Section 4, and a conclusion of this paper will be given in Section 5.

2 Background

2.1 Web Navigation

With the widespread acceptance of the World Wide Web, hypertext [13], which is the information structure used in WWW, has become the de facto standard of information storage and retrieval in the Internet. The WWW can be viewed as the collection of web servers that communicates with web browser using the connectionless HTTP protocol in request-reply pairs, and stream out the requested content written in the Hypertext Markup Language (HTML). HTML has evolved over time and with other Web technologies, and thus created some navigation features that are not available by hypertext alone.

In WWW, Web pages and links replaces information fragments and relationships in the hypertext model. Users navigate from a web page to the other by clicking on a hyperlink. In addition to this basic navigation method, various client side scripts and programs, such as Javascript,

VBScript, JAVA and ActiveX, provide other ways to control traveling between different web pages without clicking a link. Dynamic HTML, ActiveX components are also widely employed to change the content and hyperlinks of the currently displaying web page.

Concurrent display of web pages is possible by the use of frames and windows. Synchronization between the concurrently displayed web pages can be achieved with functions of the client side scripts and control programs. Detailed discussion on navigation of web pages is in Section 3.

2.2 Web Modeling

It is not hard to realize that the complications added by the above discussed web features makes the web page and links not necessarily a direct mapping from the information fragments and relationships in hypertext. So modeling techniques used in hypertext [13, 16] may not be directly applicable. The researches on web modeling are comparatively few to the hypertext model researches.

Conallen [2] discussed using the common behavior package in Unified Modeling Language (UML) [10] to model the business logic in web applications. Client pages, Server pages, Forms and Frames are defined as classes in UML, with their contents modeled as stereotyped attributes of the class. Links between web pages are modeled as associations between the linked classes. The modeling is viewed from the perspective of business logic in structure and functional view, so some user interface attributes affecting presentation and browsing semantics are intentionally left out from this model. The suggested modeling method can be used as a compliment to our model to supply information about business logic.

On the same track with us, HMBS [15] (Hypertext Model Based on Statecharts) uses statechart (see section 2.3) as a modeling tool. HMBS aims to support behavioral modeling in the design of hypermedia by the suggested model. An hypermedia authoring tool is also developed to achieve this goal. In the model, hypermedia pages are mapped to states in statechart by a mapping function M, which can be altered for different statecharts to make abstraction of physical pages possible. M is held static for a particular statechart, and cannot be changed during navigation. This freedom in definition of M may be feasible in hypermedia authoring as the designer can define M arbitrarily, and then all development thereafter will adhere to it. Problem arises if multiple parties wish to model an existing set of hypermedia pages, as it is not clear that how M can be defined in a consistent way by different modelers, there may be vast difference between resultant models of the same set of web pages. We will solve this problem by providing a consistent way to map web pages in our navigation model.

HMBS is more concerned with the hierarchical structuring of hypermedia pages than other navigation methods between pages. *Visibility level L* is defined to manage the visibility (state) of hierarchical pages during hierarchical navigation. All other navigation methods are generally described as events representing transition of states, with no details on different kinds of events given. Treating all kinds of events equally will introduce difficulties in modeling different kinds of hyperlinks in the web, such as between an ordinary hyperlink and a server side script which acts on parameters sent along with the hyperlink transition. We analyze and identify different ways to model different kinds of hyperlink in Section 3 and 4.

It is also not clear that how can the HMBS model support modeling of dynamic hypermedia pages, which are used extensively in contemporary web development. Even with the changeable mapping function M, it is impossible to model physical hypermedia pages that changes their content during navigation, or contains different content when different links are used (events) to enter the page. This is because M must be held static in a navigation session. This is especially true for those pages that changes their links (transitions) to other pages (states) during navigation, making a static mapping from physical pages and links to states and transitions impossible. In follow up works of HMBS, the modeling focus had shifted to timing and synchronization of multimedia content in hyperdocuments [11, 12], and the development of hyperdocument authoring software [14], so the above problems stills remain unresolved. In Section 4, we will discussed how to overcome all these difficulties by our navigation model.

There are also some researches on hypermedia modeling, which is the modeling of multimedia hyperlinked contents. HMData [6] is an object oriented modeling method for storage and retrieval of hypermedia. This modular approach discourages linkages between individual elements of different modules, and may not be suitable for the web where extensive, spaghetti linkages between web pages of different sites is the norm. EFSM [7] is an extended finite state machine to model temporal synchronization of temporal interactive web content. The analysis is mainly focused in synchronization, and the model may be blown up to unmanageable size if the quantity of media is large and linkage between media extensive, due to the nature of finite state machine.

Another approaching to model the navigation of the web is by using web page content in addition to their intra-linkages [1, 9]. The general idea is to use the content similarity between pages to structure them, providing a better understanding and sense of orientation for the user of the information space. These models give general shape of the information space, browsing the web pages in an "overview" mode, and is not suitable for analysis on browsing semantics between individual pages.

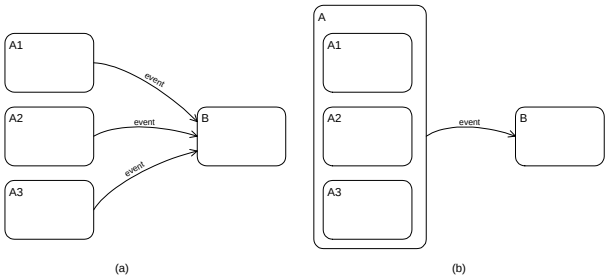

Figure 1. XOR grouping of states

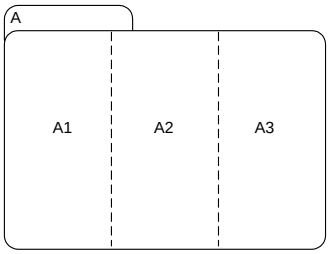

Figure 2. AND grouping of states

2.3 Statechart

Statechart is first proposed by Harel in 1987 [3, 5]. It is a visual formalism to extend state diagrams for modeling of complex systems that involves large number of concurrent states, synchronizations and action triggers. There are numerous extensions to statechart to support different modeling needs, we will introduce the basic and the relevant part of statechart we used in this paper here.

A statechart can be defined as the basic set of elements: *states, transitions, primitive events, primitive conditions and variables,* plus the extended set of *events, conditions, expressions* and *labels* and their inter-relations [5]. S is defined as the set of states. A *hierarchy function* $\rho : S \rightarrow 2^S$, is defined to map each state to its sub-states. $\rho(s) = \emptyset$ means s is a *basic* state. A *type function* $\psi : S \rightarrow \{AND, OR\}$ defines whether $s \in S$ is a *composed* AND / OR state or not. H is the set of *history symbols*. It is related to S by a function $\gamma : H \rightarrow S$. The *default function*, $\delta : S \rightarrow 2^{S \cup H}$ defines the initial states in S. V_p is defined as the set of variables and the set of *expressions*, V, is defined on V_p. The set of primitive conditions is defined as C_p with the set of conditions C defined on it. Similarly, E_p and E are defined as the set of primitive events and events respectively. A is defined as the set of *actions*. $L = E \times A$ is the set of *labels* on transitions. The set of transitions, T, is defined as $T \subset 2^S \times L \times 2^{S \cup H}$.

For clarity, state names will be shown in **bold** and event labels will be shown in sans in this paper.

Like a state diagram, a statechart model starts in an initial state. When an event happens, the system will transit to the next state by a transition triggered (labeled) by that event. An optional activity can be added to the transition label, indicating that the activity will take place when the transition happens. The triggered activity can in turn received by the system as another event to trigger other transitions.

State can be grouped together by *XOR* for efficient use of transition arrows. Figure 1 shows the savings from the expanded (a) to the more concise (b). The system can be exclusively in state **A** OR **B**. Concurrent states, like independent modules in a system, can be modeled by orthogonal *AND* of sub-states that exist concurrently (Figure 2). In the figure, being sub-states of the composed state **A**, **A1, A2** and **A3** maintains their own state.

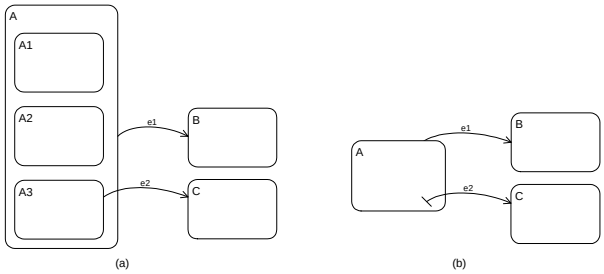

Figure 3. Abstraction of states

Abstraction of statechart can be done by hiding details of sub-states. Figure 3a is abstracted to Figure 3b by hiding the details of the *XOR* sub-states. The stubbed transition arrow of *e2* represents that the transition is available from some of the sub-states of **A** only. The *select* connective can be used to reduce the number of transition arrows. Events e in Figure 4b is defined as the disjunction of events e1, e2 in Figure 4a. e replaces e1, e2 by pointing to the *select* connective, meaning that **A** will transit to **B1** or **B2** depending on e1 or e2 happened.

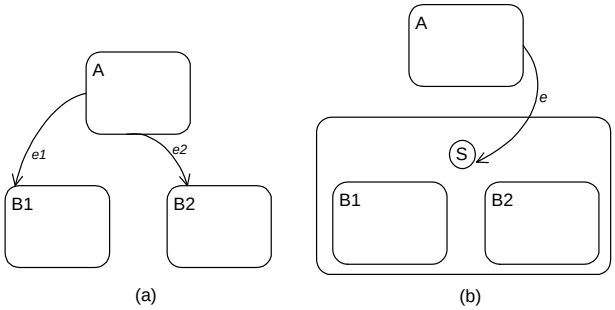

Figure 4. Select connective

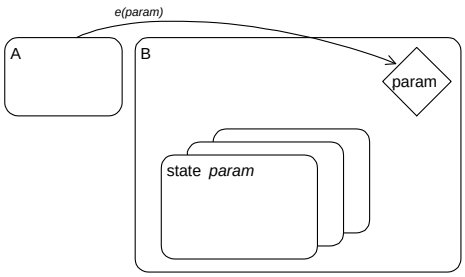

Figure 5. Parameterized OR

Events can affect states globally. This global effect of all occurred event is described as broadcasting of events to all parts of the system. All activities/events are considered instant (zero time), so synchronization can be supported by the triggering of some limiting events on all concurrent partitions.

We will also use the suggested *parameterized OR* extension to statechart [3] for situations where large quantity of similar states are encountered (Figure 5). Event e will happen with a parameter *param*. In **B**, *param* will be evaluated and the system will transit to the appropriate state, out of the group of states **state** *param*, based on this evaluation.

There are problems of undetermined state when events/activity are defined as zero time, as described in [3, 5, 8]. We will take the micro-step approach [8] to solved this problem.

3 Web Navigation Analysis

3.1 Web Page

A *web page* is defined as the sequence of HTML replied by the web server to a client making a request of a URL through HTTP. Other navigation related materials in the web page, such as scripts and client side programs, are considered *components* of the web page. Non-navigation related materials, such as graphics or audio clips contained in the web page, are not considered in this analysis.

A *static* web page is a web page that retains the same HTML for all the client requests of the same URL. It must also contain no reactive or executable components.

A *dynamic* web page is defined as a web page that returns different HTML for client requests of the same URL (server side dynamics), or contains reactive or executable components (client side dynamics).

3.2 Hyperlink

Navigation in the WWW is done by *activating* hyperlinks. Hyperlinks are directional links between a source and a target web page, which can be the same page, so linking to different sections of the same web page is possible. When a hyperlink is activated, the current web page in view will be replaced by the target web page. A hyperlink can be in form of text strings, graphics or video, activated explicitly by the users, using a mouse click for example. Hyperlinks can also be included in client side programs / scripts, to be invoked automatically by the browser on some predefined events. Examples of these events includes timeout, mouse movements and window focus. As in hypertext, the sequence of hyperlink activation, no matter by mouse click or other events, is the navigation path.

3.3 Web browser effects

In addition to the navigation provided by hyperlinks, web browsers can provide additional navigation functions that is out of control of the web pages, such as scrolling, back, forward buttons and a history list. We do not intend to discussion this form of navigation here, so web navigation by web browser functions are out of scope for this paper.

3.4 Intra-page Navigation

This form of web navigation involves only one web page. In a web page, hyperlinks can be defined to target different sections of the same web page. User can also scroll to different sections of the page if the page is displayed in a scrollbar enabled frame or window to obtain the same result.

3.5 Inter-page Navigation

This most common form of web navigation is by activating a hyperlink on a source web page, to jump to a target web page which the hyperlink links to.

3.6 Frame-based Navigation

Frames inside a browser window makes concurrent viewing of web pages possible (Figure 6). A browser window can be divided into frames, each containing a separate web page for viewing. The contained web pages can be another web page containing frames. Navigation within a frame can happen independently. Navigation out of a framed web page can also be done by activating a hyperlink that specifies the target is to replace the whole framed page. All frames in a window are created and destroyed

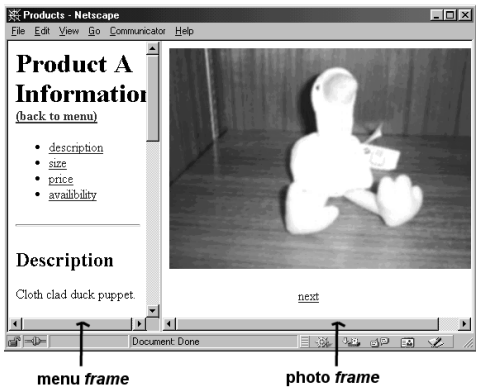

Figure 6. Screenshot: Frames

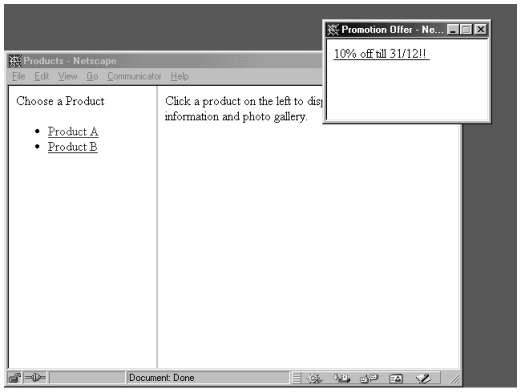

Figure 7. Screenshot: Multiple Windows

together. Frames can be set scrollable or unscrollable, and each frame maintains its own scroll position independently.

Web pages in different frames can affect each other by using client side scripts or programs. Changes that can be done includes presentation appearance and changing the web page displaying in another frame. Scripts and programs in different frames may also interact with each other to create complicated reactive behavior.

3.7 Navigation of Multiple Windows

Multiple browser windows can be used to support concurrent viewing of web pages in another way (Figure 7). Windows differs from frames in the sense that they can be created or destroyed independently at different time, and they are managed by the operating systems windows manager. Client side script and programs can affect multiple windows in the same way as in multiple frames.

3.8 Dynamic Content

The content and hyperlinks of a web page can be dynamic while viewing. There are two levels of dynamics, client side and server side.

3.8.1 Server Side

An example of this kind of dynamic web page is the search result page of a search engine. When different search strings are entered, different hyperlinks will appear in the results page depending on the keyword inputed in the search. This kind of dynamic web page generation is used extensively in the WWW today.

This is done by the web server giving out different web pages to requests of the same hyperlink target (same URL). Users activating the same hyperlink of the same target twice may not get the same returned HTML. Server side scripts and CGI programs can be used to generate these web pages. A template is used to define the structure of the document and content is filled in run time. This makes the returned web page have similar structure but with different content and hyperlinks. Note that every script or program can use more than one web page template and every template can be used by more than one script or program.

Server side dynamic page generation can also employ web forms, as in the above search engine example. The source web page will contain a form with data, when the form is submitted and the new page requested, the form data is sent along to the web server and a new web page is returned. This is equivalent to activating a hyperlink with run time generated parameters in the source web page, and then a proper target web page is constructed in the web server and returned based on the parameters.

3.8.2 Client Side

An example of client side dynamic web page is a Java program on a web page. The Java program may have a scrolling list of hyperlinks, those links scrolled out of view cannot be activated while those newly scrolled into view can be activated.

In these client side dynamic pages, the target of a hyperlink can be dynamically defined, enabled, or disabled while a web page is displaying, without contacting the web server. Based on user selections or other external factors such as current time, the target of a hyperlink can be different when it is activated. This kind of web navigation is supported in various client side scripts and programs, such as JavaScript and Java. These scripts and programs have their own state and reactive behavior, and can affect the content and navigation of the hosting web page. They cannot execute without a hosting web page.

A special property of these client side scripts and program is that their life time is the same as the web page hosting them. If the web page is replaced by another or window is closed, all the attached scripts and programs will be terminated immediately.

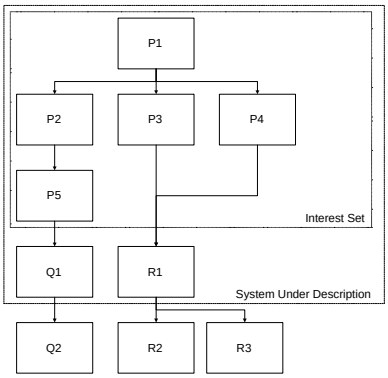

Figure 8. Defining System Under Description

4 Mapping from Web Navigation to Statechart Modeling

In this section, *web navigation* will be modeled by using Statechart. In general, web pages and their accompanied client programs and scripts are modeled as states in the Statechart, and the hyperlink between web pages are modeled as transitions, governed by the events that triggers the transition. Only the navigation related elements are modeled to suit our goal. Because of the complexity of the contemporary WWW, the modeling scope will first be discussed, then we will introduce modeling of web navigation that applies to all web pages, followed by web navigation model extensions for dynamic web pages.

4.1 Modeling Scope

In the web environment, it is not possible to model all web pages hyperlinked directly and indirectly to the pages we are interested in. This is because there are rarely any "dead end" of web pages that do not have any hyperlink out of it, so by simply following hyperlinks, one can navigate to some millions of web pages. We must therefore limit the scope of the system under description.

The set of web pages that are of interest is first identified, then the scope of the system under description is limited to any web page that belongs to the set or is *directly* hyperlinked by any web page belongs to the set. As shown in Figure 8, some web pages with their hyperlinks are represented by a simple directed graph. The interest set is first identified as web pages *P1* to *P5*. The immediately linked pages are *Q1* and *R1*. So the system under description includes web pages *P1, P2, P3, P4, P5, Q1, R1*.

The reason to include this extra "layer" of pages in the model is that navigation paths leading out of the interest set of pages is also important for navigation modeling, as they act as the "exit" of the system, and the destinations of these exit paths will also be of importance.

By the same principal, it is arguable that web pages that lead into the interest set should also be modeled. But by the directional nature of hyperlink, the target web page of a hyperlink cannot know what is the source web page. So it is impossible to identify which web pages, if any, leads into the interest set. Moreover, after arriving at the interest set, the "entrance page" is of no more value to navigation as hyperlink is not bi-directional and there is no way to navigate back to it. If, however, there is a link back to the "entrance page" in the interest set, that "entrance" will be modeled as an exit path.

Exit path destination pages are modeled as *basic* states. Note that these exit pages are not necessarily *terminal states*, unless every component of the system has entered these exit page states, the system can come back from these states. See *Client Side* in Section 4.2.5 for an example of re-entering to other states from an exit page state for part of the system.

4.2 Web Navigation Modeling

4.2.1 Intra-page

For intra-page web navigations, when scrolling is not possible (either limited by a short page length or by intentionally disabling the scroll bar) and there is no self targeted hyperlink, the web page is modeled by an *basic* state (Figure 9a). [1]

In other cases, sub-states are used to represent the different positions of the web page in view. We first define jp(*target, pos*) is the event to jump to position *pos* of a target web page *target* by a hyperlink. Both *target* and *pos* are defined as primitive variables in the statechart model. The event of activating a self targeted hyperlink is modeled by setting *target* to the current web page. The page is modeled as a composed state and every possible browsing position of the page is modeled as a possible sub-state of the page. When the page is in a particular sub-state, further intra-page hyperlink jumping events will cause it to leave the current sub-state to the new appropriate one by the *select* connective in Statechart. The proper new sub-state is selected by the value *pos* (the new display area) of the event to reflect the new browsing position (Figure 9b).

4.2.2 Inter-page

This is a more trivial kind of mapping from hyperlinks to state transitions. Abstraction is done here on individual pages in composed state, details of sub-states are hidden

[1] The web pages which Figures 9-16 modeled are located at *http://www.csis.hku.hk/∼swflow/www/* with their respective filename as shown in the Figures. Readers can refer to them to experience the navigation modeled.

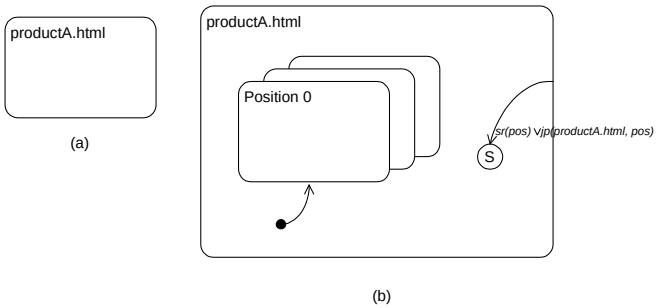

Figure 9. Intra-page Modeling

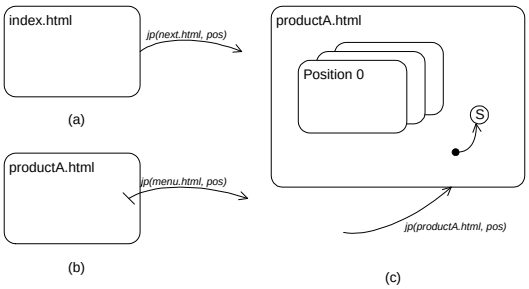

Figure 10. Inter-page Hyperlinks

for simplicity. Every hyperlink from one page to the other is represented by a state transition arrow, the hyperlink activation is the event that triggers the transition. If multiple hyperlinks on a web page point to the same target, they are considered equivalent and only one transition arrow is needed to represent them. The hyperlinks can be available from all page sub-states (Figure 10a) or only available from some of them (Figure 10b). Target of the hyperlink is always one of the sub-states of the target web page, so this can be represented by defaulting to the *select* connective (Figure 10c). The change of state represents the change of displaying current web page to the target web page of the hyperlink.

The *XOR* decomposition of Statechart plays an important role here to simplify the diagram. Web pages having hyperlinks of common target can be grouped. The model of a common tree-like menu structure of web pages is shown in Figure 11a to illustrate this. jp(*menu*, *pos*) of **productA.html** and **productB.html** are grouped together to be represented by one transition arrow.

The *select* connective can also be used to group all hyperlink targets of a page. In figure 11b, menu.jp event from state **menu.html** is defined as the disjunction of the two lower level events jp(*productA*, *pos*) and jp(*productB*, *pos*) in figure 11a. This definition should accompany the statechart so expansion back to Figure 11a is possible if desired.

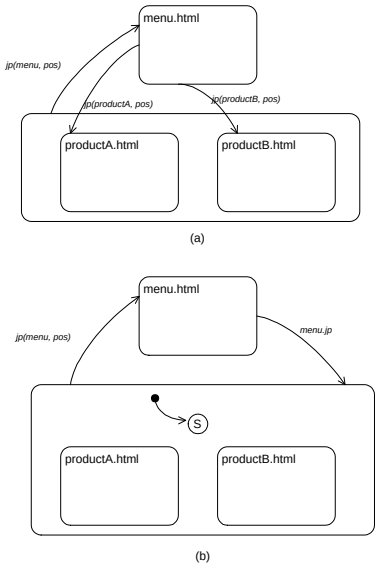

Figure 11. Inter-page Modeling

4.2.3 Frame-based

A web page with frames is modeled as the orthogonal *AND* of the state of its frames (Figure 12). An optional state name, such as **frame1**, **frame2** and **frame3** in the figure can be given to each frame. **frame2** and **frame3**, having two or more possible sub-states, has a default state to indicate the default web page loaded in that frame. This is indicated by the default arrow in the Statechart. The sub-states can be a simple web page or another framed page, Navigation from a framed web page to another page can be modeled as the

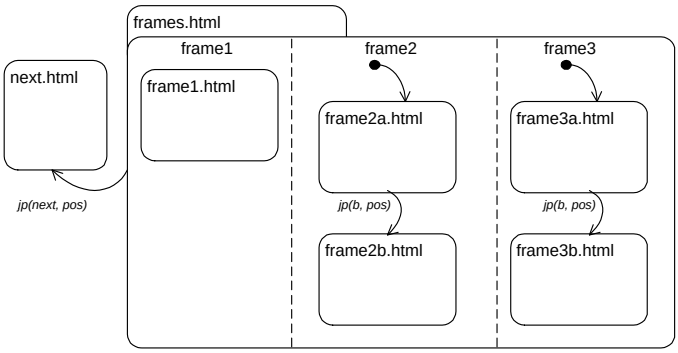

Figure 12. Modeling framed page

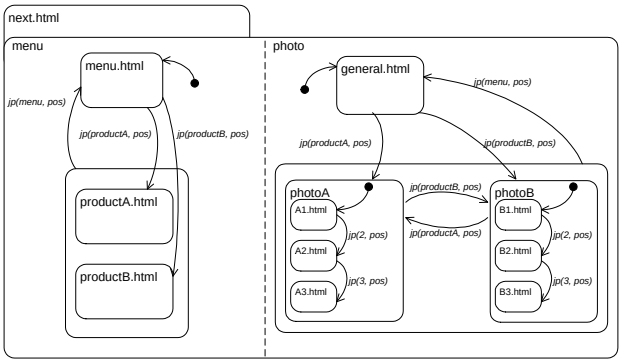

Figure 13. Synchronization in Frames

event jp(*next, pos*) in Figure 12.

Synchronization between frames can be done by events of client side scripts and programs. Most of the events generated have one intended receiver to synchronize with. This synchronization is modeled by event broadcast of Statechart. Every event is received by all frames to react upon, so that their states can be synchronized by events. The difference bring about by using broadcast in the statechart model instead of unicast events can be solved by adding the receiver's identifier to the event name when modeling, so that even if the event is broadcasted, only the intentional receiver will act on it. Other components can realize the event is not for them and discard it.

Figure 13 demonstrates this. A web page **next.html** needs to have its product information and photo gallery displayed in synchronization. A **menu** frame is used for displaying information of product A and product B. A **photo** frame is used for displaying the photo gallery of the currently displaying product. Synchronization between the displaying product and its photo gallery is by the jp(*target, pos*) events (Figure 14). Initially, the page is in the sub-states **menu.html** and **general.html**. When event jp(*productA, pos*) happens, frame **menu** transit to state **productA.html** and frame **photo** transit to state **photoA** (Figure 15). In **photoA**, the photo gallery of product A can run by the events jp(*2, pos*) and jp(*3, pos*). The two frames are synchronized. Events jp(*productB, pos*) and jp(*menu, pos*) performs similarly.

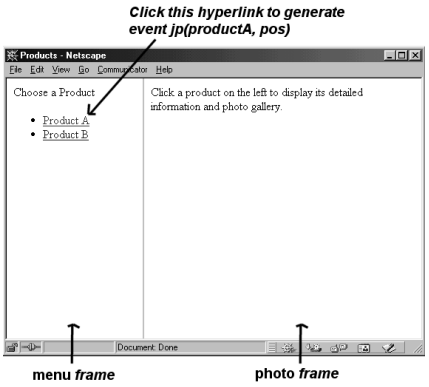

Figure 14. Initial State of Figure 13

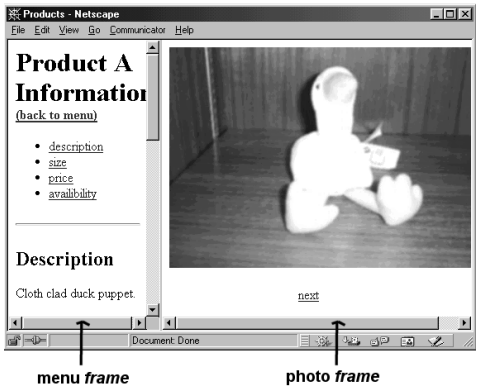

Figure 15. After event jp(*productA, pos*)

4.2.4 Multiple Windows

Concurrent viewing of web pages by multiple windows is modeled by using a separate Statechart for each window. This is modeled in Figure 16. The system first started with one window *mainWin*. When event jp(*next, pos*) happened, a new window is popped up containing **ads.html**. This is represented by specifying the name of the another statechart *adsWin* (which is of the newly created window) as the ac-

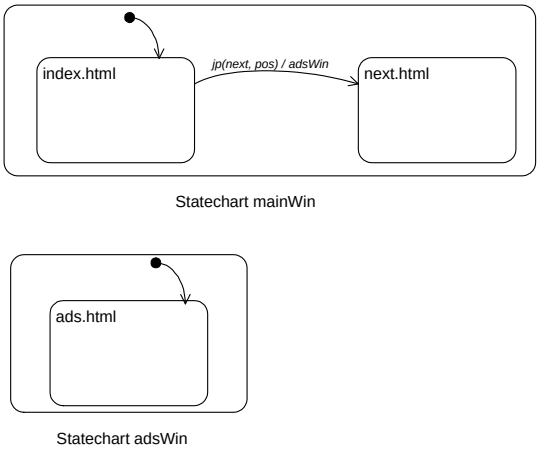

Figure 16. Modeling Multiple Windows

tion of the event. They are not modeled as orthogonal *AND* because not all of the windows will co-exist all of the time, either by design or by user manually creating or closing new windows, so there are times when the system is not in all of the sub-states simultaneously.

Synchronization between windows is still possible by the same mechanism as in frames. Events can be directed to a different window than the generating one. Modeling this synchronization is done by "broadening" the effect of broadcast of events to every Statechart in the model.

4.2.5 Dynamic Content

Dynamic web pages give rise to difficulties in web navigation modeling, as not all states are known in advance (as of *static* web pages) and the size of the set of possible state can be very large even if known. The set of possible navigation paths of these dynamic pages can also be dynamic. The followings discuss the modeling method of these dynamic relationships.

Server Side

To model server side dynamic content, it is impossible to list out every possible result page as a separate state in the model because of the sheer quantity. *parameterized-OR* is used here to solve the problem. All pages generated from the same template are considered to be a single parameterized state (**result.html** - Figure 17). A new event sm(*target*, *param*) is defined as the action of sending out a page request to *target* with parameters *param*. Both *target* and *param* are primitive variables in the statechart model. There will be a set of possible return pages and the one actually returned will be selected or generated using *param* by server side programs at run time.

Hyperlinks leading out of the parameterized state

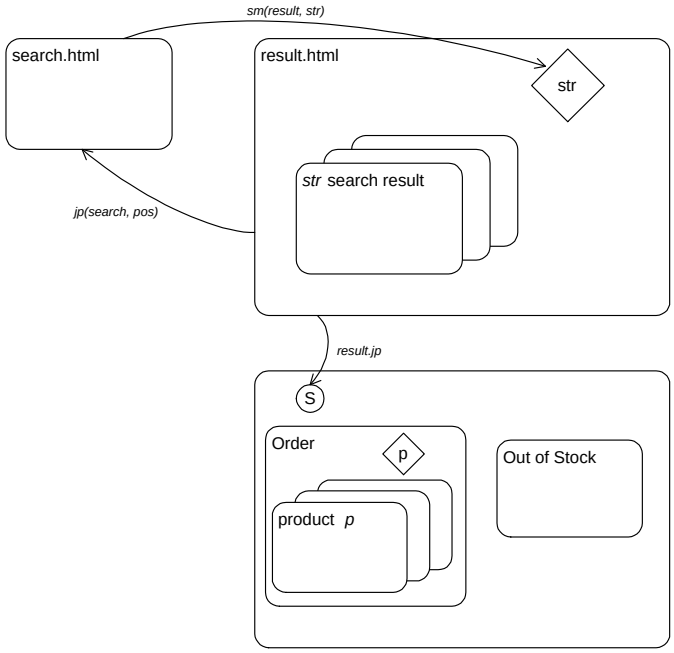

Figure 17. Server Side Dynamic Pages

have two types. For the hyperlinks that are present in all parameterized pages, the parameterized set of web page can be treated as a single state with the transition leading out normally as shown by the event jp(*search*, *pos*) from **result.html**. For all other hyperlinks, if their quantity is too large or cannot be predetermined, a disjunction of all these lower level events need to be defined to connect to a *select* connective. In Figure 17, the event result.jp from **result.html** is defined as the disjunction of events sm(*order, p*) and jp(*outOfStock*, *pos*), which in turn represents jumping to the order page of the selected product *p* and a *Out of Stock* page respectively.

Client Side

We are not concerned with the state change in client side scripts and programs that do not have any implication on navigation. So they are modeled as any other executable object that maintains their own state by Statechart [4]. Out web navigation model will not cover modeling of executable objects. The Statechart obtained from executable object modeling is orthogonal *AND*-ed with the web page state which hosts the script or program to form the new composed state representation of the page. Because the scripts and programs are initialized and destroyed with the hosting web page, the system will be in all of these sub-states concurrently. This is similar to the modeling of *frames*, but sub-states of a client side program are not web page states and transitions are not from hyperlink activation.

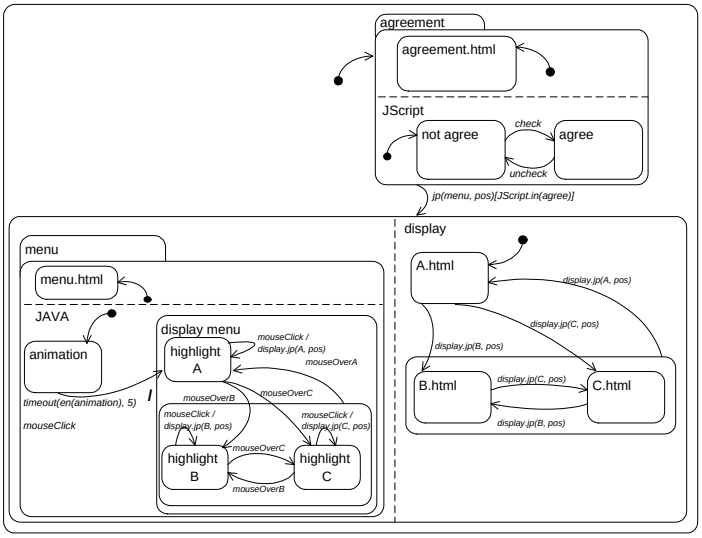

Figure 18. Client Side Dynamic Pages

They are program states and events instead.

On the navigation aspects, these scripts and programs can post constrains (enable/disable) on the hyperlinks, this is naturally modeled as constrains on the transition triggering events. Dynamic definition of hyperlink target is modeled by variables of transitions. *target* of a hyperlink is a variable in the model, which the scripts and programs can change the value at run time and achieve the effect of dynamically defining a hyperlink. Note that some scripts and programs can post constrains on hyperlinks of pages other than the hosting page.

Figure 18 illustrates this. The system starts with the state *agreement*. JScript is used with a checkbox to make the user agree to some terms and conditions before proceeding. This is modeled by the constrain on the event jp(*menu, pos*). The condition *JScript.in(agreed)* is true only if the sub-state JScript is in **agree**, and the transition is allowed only if this condition is true.

In **menu**, a JAVA menu, embedded in menu.html, is displayed in separate frames with the information pages A, B and C. The JAVA menu is modeled as an orthogonal substate of the **menu** state. It will first show an opening animation when initialized, and will proceed to display the menu items after the animation has ended, or when the user clicks a mouse button.

Actions in state **display menu** shows how the JAVA program control navigation on another frame. When a mouse click event is received by the JAVA program, it will replace the current content of the **display** frame with the page the currently highlighted item represents. This is modeled by the action display.jp(*target, pos*) of the event mouseClick. mouseClick in different states **highlight A, highlight B, highlight C** will give a different target to the action, and this action will be received by the **display** state as a event to trigger transition. The name *display* is inserted before the jp(*target, pos*) event to prevent ambiguity.

Regarding the scope of system under description, note that some, or all of *A.html*, *B.html* and *C.html* can be outside the interest set. Control actions from the JAVA program make it possible for the system to return from those states even if no hyperlink leading out of them is modeled. This shows that they are not *terminal states*.

5 Conclusion

We have pointed out previous hyperdocument modeling techniques of web navigation cannot model dynamic web navigation, which is used extensively in contemporary web pages. Most of these previous models only address hypermedia, without the support for dynamic content. We analyzed different kinds of web navigation methods, and a web navigation modeling method using statechart is proposed to provide a more comprehensive modeling of web navigation. This new model provides a simple way to model complex, dynamic web navigation. Future works will be carried out on refinement of the model and the analysis of web navigation using the model.

References

[1] Chaomei Chen. Structuring and visualising the www by generalised similarity analysis. In *Proceedings of the 8th ACM Conference on Hypertext (Hypertext'97)*, pages 177–186, Southamption, UK, 1997.

[2] Jim Conallen. Modeling web application architectures with UML. *Communications of the ACM*, 42, No.10:63–70, 1999.

[3] D. Harel. Statecharts: a visual formalism for computer system. *Science of Computer Programming*, 8, No.3:231–274, 1987.

[4] D. Harel and E. Gery. Executable object modeling with statecharts. In *Proceeding of the 18th Int. Conf. Soft. Eng.*, pages 246–257. IEEE Press, March 1996.

[5] D. Harel, A. Pnueli, J.P. Schmidt, and R. Sherman. On the formal semantics of statecharts. In *Proceedings of the 2nd IEEE Symp. on Logic in Computer Science*, pages 54–64. IEEE Press, 1987.

[6] Denis Helic, Hermann Maurer, and Nick Scherbakov. Introducing hypermedia composites to WWW. *Journal of Network and Computer Applications*, 22, No.1:19–32, 1999.

[7] Chung-Ming Huang and Ming-Yuhe Jang. Interactive temporal behavious and modelling for multimedia presentations in the WWW environment. *The Computer Journal*, 42, No.2:112–128, 1999.

[8] C. Huizing. *Semantics of Reactive Systems: Comparison and Full Abstraction*. PhD thesis, Eindhoven University of Technology, March 1991. Chap. 1.

[9] Sougata Mukherjea and Yoshinori Hara. Focus+context views of world-wide web nodes. In *Proceedings of the 8th ACM Conference on Hypertext (Hypertext'97)*, pages 187–196, Southamption, UK, 1997.

[10] OMG. *OMG Unified Modeling Language Specification*, 1.3 edition, June 1999.

[11] F. B. Paulo, P. C. Masiero, and M. C. F. Oliveira. Hypercharts: Extended statecharts to support hypermedia specification. *IEEE Transactions on Software Engineering*, 25(1):33–49, January/February 1999.

[12] F. B. Paulo, M. A. S. Turine, M. C. F. Oliveira, and P. C. Masiero. XHMBS: A formal model to support hypermedia specification. In *Proceedings of the 9th ACM Conference on Hypertext (Hypertext'98)*, pages 161–170, Pittsburgh, PA USA, 1998.

[13] P. David Stotts and Richard Furuta. Petri-net-based hypertext: Document structure with browsing semantics. *ACM Transactions on Information Systems*, 7, No.1:3–29, 1989.

[14] M. A. S. Turine and M. C. F. Oliveira. HySCharts: A statechart-based environment for hyperdocument authoring and browsing. *Multimedia Tools and Applications*, 8:309–324, 1999.

[15] M. A. S. Turine, M. C. F. Oliveira, and P. C. Masiero. A navigation-oriented hypertext model based on statecharts. In *Proceedings of the 8th ACM Conference on Hypertext (Hypertext'97)*, pages 102–111, Southamption, UK, 1997.

[16] Yi Zheng and Man-Chi Pong. Using statecharts to model hypertext. In *Proceedings of the ACM conference on Hypertext*, pages 242–250, 1992.