TELKOMNIKA Indonesian Journal of Electrical Engineering
Vol.12, No.3, March 2014, pp. 2356 ~ 2365
DOI: http://dx.doi.org/10.11591/telkomnika.v12i3.4522

Study on View-Oriented Navigation Modeling of Web Applications

Buye Lou
School of Information, Capital University of Economics and Business
121 Zhangjialukou, Huaxiang Fengtai District, Beijing 100070, China
e-mail: loubuye@163.com

Abstract

This paper presents view-oriented navigation modeling (VONM), a system analysis method and tool for web applications. Firstly, the view was discussed as a navigation node. The view is dynamic and hierarchical. The dynamic characteristic means that a view may render varied objects and state data of an object is changeable. The hierarchy of the view refers to the feature that a more abstract view may be refined to produce several more concrete views. Then, the paper analyzed the classification and the features of navigation. Non-action navigation is the navigation in the traditional sense. Action navigation is also navigation, but its primary purpose is to perform a specific data processing. In this work, the model is made up of view diagrams, navigation diagrams and specification cards for view or navigation. Finally, the paper introduced some navigation implement patterns for the action navigation.

Keywords: *view, navigation diagram, specification card, system analysis*

1. Introduction

The emergence of the World Wide Web has brought about a new generation of application software: web application. In view of the difference between web application and traditional application software [1], many development methods for web application have been proposed. There are some famous ones among them: HDM, RMM, OOHDM, WebML, WAE and UWE.

Web navigation occupies a significant position in web application. Compared with early websites, navigation of web application needs to reflect dynamic of the web page and functionality of the web application. Many Web application development methodologies specialized studies for modeling and design of navigation, contribute a number of important ideas. For example, OOHDM [2] proposed the conception of navigation context which can offer the contextual limitation to the navigation target objects. WebML [3] establishes the navigation model according to the content units. Navigation links are divided into contextual links and non-contextual links, and the former represent the navigation needs to carry some information. WAE expresses dynamic of the web page and functionality of the web application through the stereotyped class <<Server Page>>, <<Client Page>> and <<HTML Form>> [4]. In order to express the data processing involved in the navigation process better, UWE [5, 6] introduces a stereotyped class <<process class>> and a stereotyped association <<process link>> under the foundation of the stereotyped class <<navigation class>> and the stereotyped association <<navigation link>>.

Some literature [7, 8] discuss navigation modeling approaches based on the famous statechart [9]. The graphical representation of navigation can be simplified by the hierarchy and XOR decomposition of states. In the StartWebStarts [8], the following states can act as navigation nodes: static state, dynamic state and transient state. Among these states mentioned above, dynamic state reflects dynamic of the web page and transient state reflects data processing of web application.

The view-oriented navigation modeling approach mentioned in the paper aims at:

(a) The modeling methods can be used in system analysis of web application and the model diagrams are suitable as tools in the analysis phase. The elements in the model should be independent of the specific implementation technologies.

Received August 27, 2013; Revised October 1, 2013; Accepted October 24, 2013

(b) Navigation nodes are context of the navigation, and the navigation is closely related to navigation nodes. Through well-defined navigation node, the navigation itself can have some implicit attributes or behaviors, which simplify representation of the navigation. In this modeling approach, we introduce the concept of view, as the navigation node.
(c) Navigation and data processing should be fused and unified representation. In web applications, the navigation is the way to search information, and is also the way to carry out data processing. Data processing occurs in the navigation process in common, and the result of data processing may affect the navigation behavior as well.

The remainder of the paper is organized as follows. Section 2 gives the definition of the view, and describes how to express the abstract views and its navigation. In section 3, we propose a classification of the basic view, and present the view diagram to express the basic view. Section 4 analyzes the classification and the features of navigation, and discusses the representation, the semantics and the dynamic of navigation. In section 5, the paper introduces several navigation implement patterns which can improve user experience. Finally, in last section some conclusions of this paper are outlined.

2. View and Its Navigation
2.1. Definition of the View
A view is a representation of the states or partial states of a web application system. A view may consist of several sub views and each sub view is a representation of the partial states of its parent view.

From the perspective of web application development, developers can propose several high-level views for the whole web application at first, and then they will refine every high-level view to get some low-level views. Every view must be refined until it can be implemented by some web pages. The view which is implemented by a single web page is called a concrete view or a basic view, and the one which is implemented by several web pages is called an abstract view or a composite view.

A web page can be a web page file or it can be composed of a page template file and some page fragment files. A web page can include some CSS files as well.

A view can contain some navigation elements. With these elements, users are able to surf from one view to another or carry out specific data processing.

2.2. Representation of Abstract View and Its Navigation
The views and its navigation relationships of a web application can be represented with graphical notation, referred to as the navigation diagram. A view can be represented by a rectangle; and the navigation link among the views can be represented by arrowed lines or arcs. A navigation line or arc suggests that it is possible for a view or its sub view (called source view) to navigate to another view or its sub view (called target view).

In high-level navigation diagrams, the abstract views are mainly proposed according to application functions. The high-level view and its navigation relationships of a books e-commerce website are showed in Figure 1. Here, the “home page” view needs to introduce the website and provide the login and registration functions. The “picking books” view is responsible for choosing books. Eventually, these abstract views should be refined to produce the concrete views. Each concrete view should be able to render the appropriate information or carry out the corresponding function.

The navigation diagram is a graphic tool for navigation analysis. It is essential to provide the supplementary explanation for a navigation diagram. The explanation for the high level abstract view and its navigation is rough and abstract in general, for instance, which entities and objects should be involved in a view, which functions should be provided in a view, what is the purpose of every navigation link, and so forth.

The supplementary explanation for a navigation diagram is completed with the specification of the views and navigation links in their respective cards. Figure 2 illustrates the specification card for the abstract view “picking books”. Entry “outline” describes what the user can do within the view. Entry “scope” lists the entities and objects involved in the view. Entry “functions” lists the functions should be provided by the view. Entry “note” may describes sources and characteristics of objects.

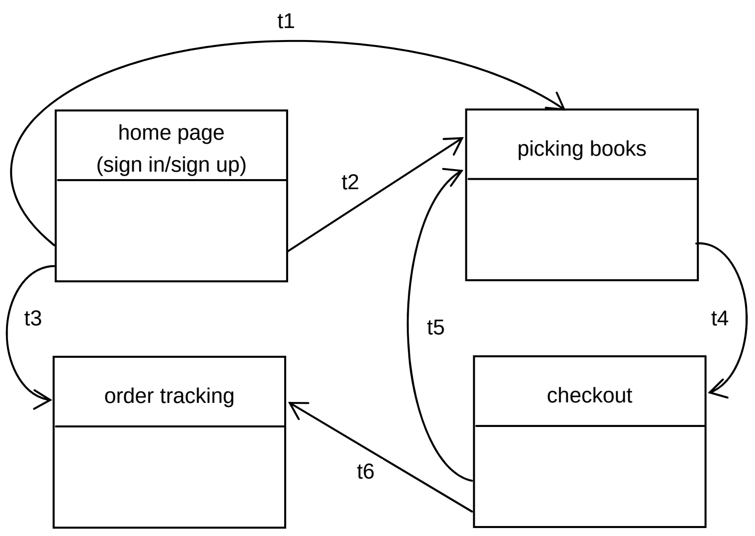

Figure 1. Example of a Hign-level Navigation Diagram

Abstract view: picking books	
Outline:	Picks books, and adds them to the shopping cart.
Scope:	Books; shopping cart.
Functions:	Searches books; displays books information; adds the book to the cart; edits the cart.
Note	The shopping cart may contain some items. Each item in the shopping cart has an index number. The shopping cart object had been saved in a specific Java Bean.

Figure 2. Example of a Specification Card for Abstract View: Picking Books

Figure 3 illustrates the specification card for the navigation “t1”. The card mainly describes the objective and parameters of the navigation link. Some navigation links carry some request parameters from the source view to the target view. The request parameters are divided into input parameters and context parameters. Input parameters are the data users enter into web form before the navigation is triggered. Context parameters are the data exist in source view. The context parameters for a navigation link should be set by application while the source view was rendered.

Navigation: t1	
Objective:	Turns to the view “picking books” to browse books’ information that meets the criteria.
Input parmeters:	The title or partial title of book
Context parameters:	None.

Figure 3. Example of a specification card for navigation: t1

In Figure 1, the navigation link t1 and t2 have the same source view (home page) and the same target view (picking books). However, their objectives and parameters are different. The objective of l2 is to browse current books’ information chosen, namely the contents of the shopping cart. It does not need to carry any input parameters or context parameters. With the specification card, we can easily distinguish them.

3. Classification and Representation of the Basic View

3.1. Classification of the Basic View

A view which can be implemented by a single web page is called a basic view or a concrete view. Without taking into account the request parameters, each basic view has a unique URL.

In general, the basic views can be divided into static and dynamic views. According to need, the URL of a basic view may or may not contain the request parameters.

a) Static View (SV). A static view always displays constant content each time. The data that are displayed by static views exist in the view files itself. Generally a request for a static view does not need request parameters.

b) Dynamic View. A dynamic view may display different contents each time. The data that are displayed by dynamic views can come from the files, databases, or memory areas.

a. No Parameters Dynamic View (DV-I). This kind of view displays the data which usually come from the memory areas. These data are usually temporary which are produced by data processing, and do not need parameters to locate, such as the data in shopping cart.

b. Dynamic View with Parameters (DV-II). The kind of view displays the persistent data which come from the files or databases. When a view is requested, the data are retrieved according to parameters and placed at specific memory areas, then displayed.

c. Default Parameters Dynamic View (DV-III). The kind of view also displays the persistent data which come from the files or databases. Unlike the dynamic views with parameters, before this kind of view is requested, the data have been retrieved according to parameters and placed at specific memory areas. When the view is requested, the data stored in the memory areas are displayed directly.

A dynamic view which displays persistent data can be designed to be a dynamic view with parameters, or to be a default parameters dynamic view. The choice depends on design strategy. If the view is regarded as a representation of a information resource, then it should be designed to be a dynamic view with parameters. URL of this kind of view is bookmarkable. For different parameters' values, the view has a different URL, and displays a specific resource data. If the view is regarded as a representation of current state data, then it should be designed to be a default parameters dynamic view. A default parameters dynamic view has a unique URL, and always displays the resource data that are retrieved recently.

3.2. Representation of the Basic View

The attributes and contents of an abstract view are rough or vague in general. An abstract view will be refined to some concrete views at the end, namely the basic views. In comparison with an abstract view, the attributes and contents of a basic view should be specific and definite. The attributes and contents of a basic view include four parts. Consequently, the rectangle which represents a basic view is divided into four columns; each column describes the corresponding attributes and content, as is showed in Figure 4(a). Figure 4(b) is a example of the basic view diagram.

a) Name, Type and Window Name of the View. In a web application, the name of every view is unique which can be regarded as part of this view's URL. The type of a view is defined according to the method mentioned in section 3.1 and can be SV, DV-I, DV-II or DV-III.

The window name is the name of the browser window or tab in which this view will render. In general, during navigation, the target view can be rendered in the window where the source view is, it can also be rendered in a new blank window or a window with the specified name. But sometimes, the different rendering strategies will bring about different user experiences. For example, the DV-I and DV-III view actually represent the current status of a kind of data resource in the system, and the state of the kind of data resource is changing, the content of the shopping cart can be taken as a good example. If it is allowed that these types of views can be rendered in different windows, then users may see multi representations of the kind of data resource, one of which is the latest state, while others may be out of date. So, we should usually specify the window name for the DV-I and DV-III view. The window name can be ignored if a view doesn't need to be rendered in the specified window or tab.

b) Query Parameters. They are used to locate the specific data resources that are displayed by the view. For the SV and DV-I view, query parameters are unnecessary in common. As a result, we can ignore the column. For the DV-II view, request parameters are part of the URL. The view itself is responsible for locating and extracting the data to be rendered according to the query parameters. For the DV-III view, request parameters are not part of the URL. Before requesting the view, other server components should locate and extracting the data that will be displayed by the view according to the query parameters in advance.

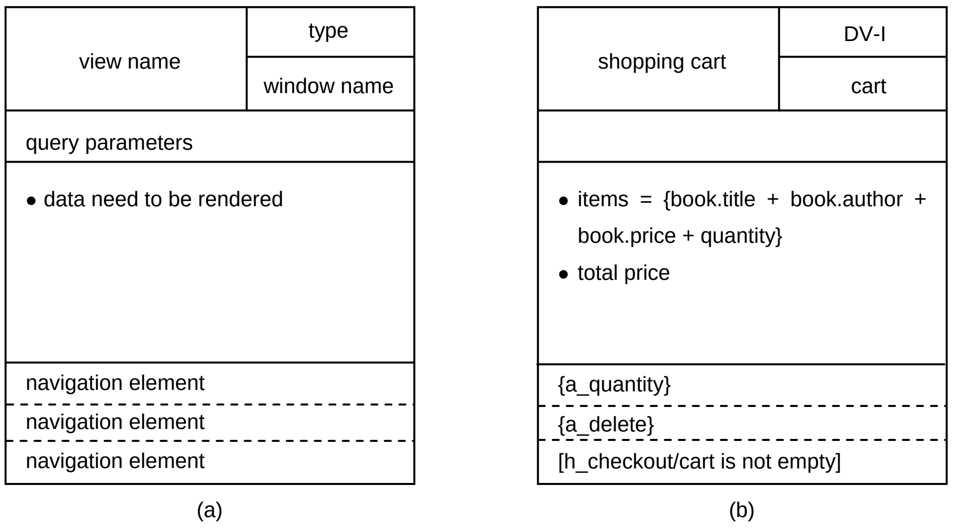

Figure 4. Basic view diagram and its example

An URL is able to contain several query parameters. Generally speaking, although changing the orders of the query parameters may not affect the semantics of the URL, the browser will treat it as a different URL. Sometimes it will give users some confused, resulting in a poor user experience. In order to avoid such situation, we should make sure to set the query parameters in a certain order for the URL. When describing the DV-II basic view, if there are several query parameters, they should be arranged in a certain order. In the design and implement phase, each query parameter should be arranged in the order mentioned above when setting the URL of the view.

c) Data Need to Be Rendered. A view can render an object or set of objects and their related objects. Every object usually contains a number of data items. When describing the data contents rendered by the view, the following symbols can be used.

a. =: stands for “be made up of…”

b. +: stands for “and”, it joins two or more elements

c. {}: stands for “repeat”, that is to say, the data in the brackets will appear in the view repeatedly.

d. [/] stands for “conditionally render”, namely the data is rendered only when the conditions are satisfied. The data separate from the conditions with the sign “/”.

d) Navigation Elements. In a view, some navigation elements are independent of the data elements, while others are embodied in the data elements. The latters are called context navigation [10]. In order to clearly indicate the navigation links between navigation elements and views, all the navigation elements need to be listed in the fourth column, and all navigation elements are separated by dotted lines.

Navigation elements belong to part of the view, the symbols used for describing data elements apply to navigation elements as well. For instance, when a group of navigation elements have the same target views and navigation behaviors, the symbol “{}” which stands for repeat can be used to represent the navigation elements. If a navigation element is available only under certain conditions, we can use the symbol “[/]”. In the next section, the representation of navigation elements and navigation behaviors will be discussed in detail.

A basic view diagram expresses the attributes and contents of a view in detail, so the specification card is not necessary for a basic view. Of course, if necessary, we can also establish specification card for the basic view, like the abstract view.

3.3. About Web Forms

In web applications, forms are everywhere. In this study, the form is considered as a tool to provide request parameters for navigation, so it isn't necessary to explicitly specifying the specific forms in the view. In system design and implement phase, developers may determine whether a view contains forms according to relevant navigation information. If navigation requires the input parameters, the view should contain the corresponding web form.

When we specify the input parameters, we may use symbol "+" to join two or more parameters. If the parameter is optional, we may the symbol "[]".

For example, suppose the input parameters are specified for the navigation:

username + password + email + [birthday] + [gender] + [country] + [phone]

Then the view implicitly contains a form consisting of seven fields. The last four fields are optional.

If navigation requires both the input parameters and the context parameters, the hidden fields standing for the context parameters can be included in the form.

4. Classification and Representation of Navigation

4.1. Classification of Navigation

In web applications, the so-called navigation refers to the placement of navigation elements in the view, so that the user can from one view to another view conveniently. The right navigations facilitate users to access the information they need and request the relevant business processing. Here are two types of navigation.

a) Non-action Navigation. Users will turn to the target view directly without any data processing after clicking the navigation elements in the source view. When representing this type of navigation elements in the source view, "h" should be prefixed before the element name; both are separated with an underscore, such as "h_go to cart".

The behavior of this type of navigation is indicated by arrowed line or arc. The starting point of the arrow is the corresponding navigation element in the source view, and the ending point is the target view. Apart from naming navigation element, navigation itself needs to be named as well. The navigation name should be marked beside the navigation lines or arcs. The notation has already been used in Figure 1 in section 2.

b) Action Navigation. Firstly, it will perform specific actions and complete related data processing. And then it will navigate to the particular target view which can usually show the results of data processing. The target view and the source view can be identical. When representing this type of navigation elements in the source view, "a" should be prefixed before the element name, between the two parts lies the underline, such as "a_delete".

There are two phases in the behavior process of this type navigation. A corresponding data processing should be carried out in first phase, and the target view should be rendered in second phase. We can use notation to indicate the process, see Figure 5.

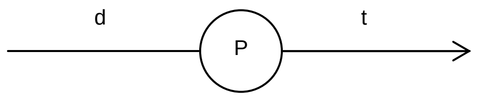

Figure 5. Notation of the Action Navigation Link

In this figure, "p" stands for data processing, "d" for the required data in data processing, and "t" for the navigation name.

4.2. Basic View Navigation Diagram

A basic view navigation diagram is a low-level navigation diagram which consists of basic views, navigation links. The navigation link is divided into non-action navigation link and action navigation link. A navigation link connects the two basic view, from the corresponding navigation element in the source view to the target view.

In this low-level navigation diagram, in order to highlight the navigation relationships between basic views, the data contents need to be rendered by views should be ignored. Namely the third column of the view box should be left blank.

Figure 6 is a example of the basic view navigation diagram. there are three concrete views and its navigation, which are refined from the abstract view “picking books” in Figure 1.

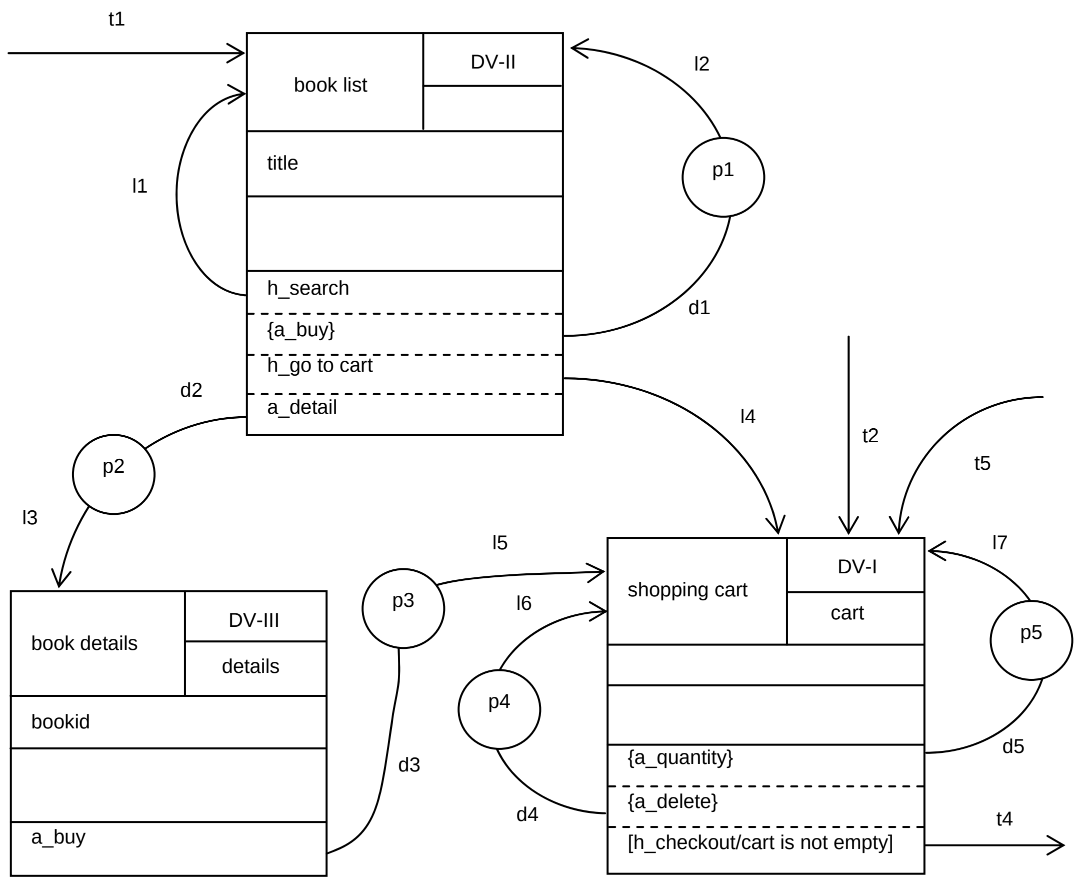

Figure 6. Example of a Low-level Navigation Diagram

The navigation in the basic view navigation diagram has richer information than the navigation in the abstract view navigation diagram:

(a) The navigation is triggered by the specific navigation element.

(b) Distinguishing between non-action navigation and action navigation manifests the fact: Which navigation needs to perform specific data processing, which does not? The data processing functions which are performed during action navigation should be consistent with the functions which the view need provide. For example, the basic view “shopping cart” includes “removes a item from the cart” and “modifies quantity for the items” functions, so there should be the corresponding action navigation elements in the view.

(c) Whether it is necessary for a navigation to carry request parameters depends on the necessity for query parameters of the target view. If the type of the target view is DV-II, the navigation implicitly carries the sequential request parameters specified in the target view. For other types of the target views, navigation doesn’t need to carry request parameters. The information is unnecessary to be indicated clearly in the navigation diagram. For example, navigation “l1” needs to carry a request parameter “query string”, while navigation “l3” doesn’t need to.

Nevertheless, the navigation information in the basic view navigation diagram is still incomplete. For example, are request parameters input parameters or context parameters? What is the specific function of a data processing? What are input data for a data processing? These informations can be expressed in specification card for non-action or action navigation.

4.3. Specification Card for Non-Action Navigation and Action Navigation

Non-action navigation and action navigation have different characteristics, so their specification cards are different. The following examples are based on the navigation diagram in Figure 6.

Figure 7 illustrates the specification card for the non-action navigation "h_search" in the basic view "book list". Compared with the navigation in abstract view navigation diagram, the navigation in basic view navigation diagram is stated in terms of the navigation element. For non-action navigation, the specification card only adds the entry "Render", used to describe how the navigation elements rendered in the view.

	Non-action navigation: h_search	**From:** book list
Render	A form used to accept input parameters should be rendered. When users click the submit button, the navigation will be triggered.	
	Navigation link: l1	**To:** book list
Objective:	Returns to the source view to browse books' information that meets the criteria.	
Input parmeters:	Title (The title or partial title of book).	
Context parameters:	None.	

Figure 7. Example of a Specification Card for Non-action Navigation: h_search

Figure 8 illustrates the specification card for the action navigation "a_delete" in the basic view "shopping cart". Entry "Function" describes the function of the data processing which is carried out during action navigation. The "input parameters" and "context parameters" make up the input data for the data processing.

	Action navigation: a_delete	**From: shopping cart**
Render	A delete button or hyperlink can be rendered for each item in the cart. When users click the button or hyperlink, the navigation will be triggered.	
Function:	Removes the current item from the cart.	
Input parmeters:	None	
Context parameters:	Index number of the current item.	
	Navigation link: l6	**To: shopping cart**
Objective:	Return to the source view to browse the status of the shopping cart.	
Parameters:	None.	

Figure 8. Example of a Specification Card for Action Navigation: a_delete

Entry "parameters" lists parameters which need to be passed to target view. It should be consistent with query string of the target view. These parameters are always set by the application automatically after the data processing is carried out.

From a logical point of view, some parameters passed to target are not contextual, but need to be entered by the user. In this case, these parameters should also be included in the Entry "input parameters".

4.4. Dynamic Navigation

A navigation element can have a fixed target view; also can dynamically determine a target view based on the current state of the application system or the user context information. This process of dynamically determining the target view is called navigation processing.

For non-action navigation, navigation processing occurs before the source view is rendered, as is showed in Figure 9(a). For action navigation, navigation processing occurs after completing the data processing, as is showed in Figure 9(b). In the figure, t1 and t2 stand for navigation names; c1 and c2 stand for the conditions to produce these navigations respectively.

Generally, we should avoid using dynamic non-action navigation. The reason is that navigation processing has done before the source view is rendered. That is to say, the target view has been determined. But the actual navigation behavior occurs until the user clicks the navigation element. During this period, if there are any changes in the states of the application system, the navigation element may be in an incorrect state.

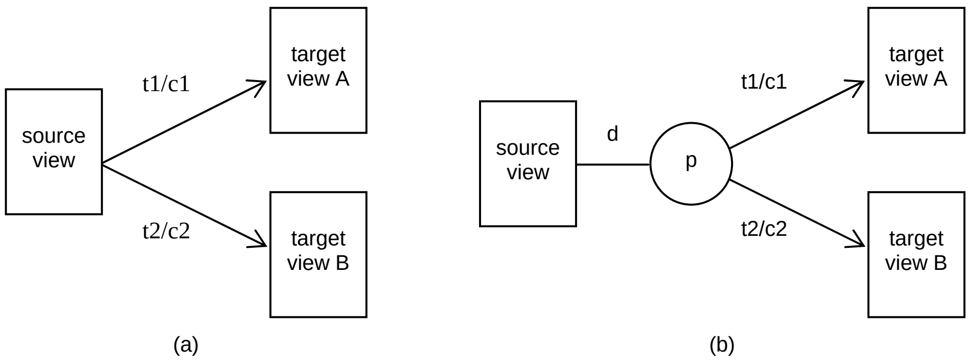

Figure 9. Notation of Dynamic Navigation

5. Navigation Implements Patterns

Web navigation is triggered by the HTTP Get or Post request. In accordance with the agreement, Get request is used for accessing the server resources, and is idempotent. Post request is used for updating the server resources, and is unsafe and non-idempotent. Generally speaking, non-action navigation should be implemented by Get request, while action navigation can be implemented by Post request. But simply using Post request will produce double submit problem and bring poor user experience.

Here are a few design pattern of action navigation, which can eliminate the double submit problem and bring good user experience.

a) PRG (Post-Redirect-Get) Pattern. At first, a Post request is sent to the server, and then the server handles the request and completes the corresponding data processing. At last, the browser sends a Get request for the target view via redirection mechanism.

As a result of using this pattern, there are only Get requests formed by redirect in the browser history stack, but not previous Post request. When users click browser refresh button, the browser just sends the GET request again, without giving annoying warning message or sending the POST request again.

b) A-PRG (Ajax-Post-Redirect-Get) Pattern. Firstly, an asynchronous Post request based on Ajax is sent to server. Secondly, the server handles the request and completes the corresponding data processing. In the end, the browser sends a Get request for the target view via the redirection mechanism.

Compared with the common Post request, an asynchronous Post request based on Ajax can selectively submit or processing the partial form data. The pattern is applicable to the situation that the target view is different from the source view, anywhere of which is unnecessary to be modified, and there is no need to provide users with the message box which displays the processing result.

c) A-PG (Ajax-Post-Get) Pattern. An asynchronous Ajax Post request is sent from client to the server. After accepting the request, the server processes the corresponding data, and then sends back an Ajax response to the client. After receiving the response, the client can partially update source view, provide user with the message box which displays the processing results if necessary. Then the client sends a Get request to the server to retrieve the target view.

In contrast with A-PRG pattern, this pattern can partially update the source view before navigating to the target view.

d) A-P (Ajax-Post) Pattern. An asynchronous Ajax Post request is sent from the client to the server. Then the server carry out the corresponding data processing and send back an Ajax response to the client. At last, the client can partially update source view based on the response message to reflect the latest state of the application.

This pattern is applicable to the situation that the source view is as the same as the target view. The view is always dynamic view, usually should be DV-I view. If there is little difference on the view before and after the processing, an extra message box may be provided in order to displaying the result information for users.

Ajax requests are not kept in the browser's history stack unless they are sent by users specifically. Clicking browser refresh button will not send the same request once more. So, if the target view is identical with the source view, it is unnecessary to load and render the source view via the redirection mechanism again.

6. Conclusion

View-oriented navigation modeling of web application can act as the methods and tools in system analysis phase. Developers can model the navigation structure with a combination method of top-down and bottom-up. Different scale web applications can have different level navigation diagrams and specification cards. In this paper, the basic view is divided into the static view, the no parameters dynamic view, the dynamic view with parameters and the default parameters dynamic view, which makes representation of the navigation itself more concise. For example, a navigation whose target view is a dynamic view with parameters implied needs to carry the corresponding request parameters. In the action navigation, the combination of data processing and navigation process makes the process where users interact with web application more intuitive and straightforward. The kind of model describes the navigation structure of web application, and its user interface. The concrete view mentioned in the model can lay the foundation of designing web page.

Acknowledgements

This work was financially supported by key education reform project of Capital University of Economics and Business (00791254210162) and Beijing Natural Science Foundation (9122003).

References

[1] Roger S Pressman, David Lower. *Web Engineering: A Practitioner's Approach*. New York: McGraw-Hill. 2009.
[2] D Schwabe, G Rossi. An Object Oriented Approach to Web-Based Application Design. *Theory and Practice of Object Systems (TAPOS).* 1998; 4(4): 207-225.
[3] S Ceri, P Fraternali, A Bongio. Web Modeling Language (WebML): a modeling language for designing Web sites. Computer Networks. *The International Journal of Computer and Telecommunications Networking*. 2000; 33(1-6): 137-157.
[4] J Conallen. Building Web Applications with Uml. Boston, MA: Addison-Wesley Longman Publishing Co., Inc. 2002.
[5] N Koch, AKraus. *The expressive power of UML-based engineering*. In Second International Workshop on Web Oriented Software Techonlogy (CYTED). Malaga, Spain. 2002: 105--119.
[6] N Koch, A Knapp, G Zhang, H Baumeister. UML-Based Web Engineering: An Approach Based on Standards. *Web Engineering: Modelling and Implementing Web Applications, HCI Series*. London: Springer-Verlag. 2008: 157-191.
[7] K Leung, L Hui, S Yiu, R Tang. *Modelling Web Navigation by StateCharts*. Proceedings: 24th Inter. Comp. Software and Applications Conf., Electronic Edition. IEEE Computer Society DL. Los Alamitos. 2000: 41-47.
[8] M Winckler, P Palanque. *StateWebCharts: a Formal Description Technique Dedicated to Navigation Modelling of Web Applications. Interactive Systems. Design, Specification, and Verification.* Lecture Notes in Computer Science. 2003; 2844: 61-76.
[9] D Harel. Statecharts: a visual formalism for complex system. *Science of Computer Programming*. 1987; 8(3): 231–274.
[10] Jesse James Garrett. The Elements of User Experience: User-Centered Design for the Web and Beyond, Second Edition. Berkeley, CA: Pearson Education, Inc. New Riders. 2011.