Towards an Automata-Based Navigational Model for the Specification of Web Sites[⋆]

Graçaliz Pereira Dimuro[1] and Antônio Carlos da Rocha Costa[1 2]

[1] Escola de Informática, Universidade Católica de Pelotas,
R. Felix da Cunha 412, 96010-000 Pelotas, Brazil
{liz,rocha}@atlas.ucpel.tche.br
http://descartes.ucpel.tche.br

[2] Programa de Pós-Graduação em Computação,
Universidade Federal do Rio Grande do Sul,
Av. Bento Gonçalves 9500, Bloco IV,
91501-970 Porto Alegre, Brazil
http://www.inf.ufrg.br

Abstract. This work presents a formal model for the specification of user navigation in web sites, based on the concepts of automata theory. The finite automata that constitute the model are composed by a set of web pages representing states with links interpreted as transitions between pages, either simple transitions, transitions with associated actions or even guarded transitions. Frame sets are denoted in a matrix-oriented notation, and transitions between frame sets are represented by arrows between such matrices. Sub-sites are defined as sub-automata, that is, subsets of pages closed to the transition function. Dynamic-page navigational automata are introduced for modelling navigation in web sites with dynamic pages. A brief description is given about the role the model can play with respect to the use-case specification of web sites.

1 Introduction

This work presents an automata-based navigational model for modelling web sites from the user's point of view. It is the first step of a research directed to the use of statistical and probabilistic models (like Markov Chains and Hidden Markov Models) as tools for the design and analysis of interactive Internet applications. Automata are the structures underlying such models and the navigational model presented here shall be the structure underlying the adaptation of those models to our envisaged applications.

In this paper, a definition of *navigation automaton* is introduced, based on the concepts of automata theory (finite automata, non-deterministic automata, Mealy machine, etc. [7, 9]).[3]Four kinds of navigation automata are considered:

[⋆] This work has financial support from CNPq and FAPERGS.

[3] We note that automata have been used in connection to navigation processes in web sites, e.g., as a user tool to constrain search processes [10]. In this paper, navigation automata are used as a design tool to specify all the possible user navigations in a web site.

1. *simple-page automata*, modelling the navigation in *web sites of simple pages* – sites composed by finite collections of static web pages, exhibited one by one, with links, called *simple-page links*, attached to anchors[4], allowing the navigation from one page to another;
2. *frame-set automata*, for *web sites with frame sets* – sites presented as finite collections of frame sets[5], each one showing several static pages at the same time, with links, called *frame-set links*, allowing the navigation from one page to another page which is to be exhibited in the same or in a different frame, within the same or in another frame set;
3. *dynamic-page automata*, modelling the navigation in *web sites with dynamic pages or frame sets* – sites having a set of links, called *dynamic links* (see sect. 2), that are attached to the same anchor, all of them pointing to pages, called dynamic pages, that are generated by a script performing a system-parameterized action associated with that link anchor;
4. *sub-site automata*, for sites presenting self-contained (encapsulated or non encapsulated) *sub-sites*.

We then briefly show that the navigation automaton is a mathematical formalism that can be used to represent in a intuitive way scenarios (both the primary scenario and the possible alternatives paths) of use cases of web sites. Such scenarios describe possible interactions between the site and its users (see the literature for such concepts, e.g., UML – Unified Modelling Language [1, 4, 8]).

The paper is organized as follows. Web sites and related concepts are formalized in Sect. 2. The concept of navigation automaton is introduced in Sect. 3. The simple-page navigation automaton model is presented in Sect. 4. Section 5 presents the frame-set navigation automaton. The dynamic-page navigation automaton modelling navigation in sites with dynamic pages is introduced in Sect. 6. Sub-sites are discussed in Sect. 7. Some envisaged applications are discussed in Sect. 8. The conclusion is in Sect. 9.

2 Using Graphs to Formalize Web Sites and Related Concepts

To allow the application of the formal model of finite automata to the navigation in web sites, a minimal formalization of web sites and their related concepts is

[4] An *anchor* is formally defined as the specific place in a web page where a link is pointing to. However, since the HTML element used to create a link is the same used to introduce an anchor, in practice it is common to call anchor the specific place in a page where the origin of the link is located. In this work, we say that a link is attached to an anchor to mean that this anchor is where the link departs from. It will be clear from the context when we refer to an (target) anchor, i.e., the place in a page where the link is pointing to.

[5] A *frame set* is a set of frames dividing a window or another frame of this window. A frame set allows the browser to load several pages at the same time, each one in a different frame of the frame set. A frame set of just one frame may be identified with a *simple page*.

required. In this section, we briefly present a graph-based formal concept of web site, taking into account its external observable behavior. This means that we do not consider the way the system is internally implemented, but the way it interacts with the user.

The concepts of *web page* and *link anchor* are considered primitive. The collection of all web pages is denoted by $\mathcal{P}$, and A is the collection of all link anchors. The concept of link between web pages is formally defined. In order to unify the nomenclature, several links shall be considered:

Definition 1. *A* simple-page link t_a *attached to an anchor* $a \in A$ *in a page* $P_1 \in \mathcal{P}$*, and pointing to a page* $P_2 \in \mathcal{P}$*, is defined as a pair* $t_a = \langle P_1, P_2 \rangle$*.*

A *user click* on the anchor a is the event that activates the link t_a in the page P_1, allowing the browser to load the page P_2.

If $P_1 = P_2$, then $t_a = \langle P_1, P_1 \rangle$ is a simple-page *self*-link. Simple-page *internal* links are denoted by $t_a(a') = \langle P_1, P_1(a') \rangle$, where a is the anchor in P_1 where the link $t_a(a')$ is attached to, and a' is the anchor in P_1 where $t_a(a')$ is pointed to.

Simple-page links *with associated actions* $x \in \{$download, upload, create e-mail message, open new window, $\dots\}$ are denoted by $t_a[x] = \langle P_1, P_2[x] \rangle$.

A link anchor may be associated with a parameterized action. We consider two kinds of parameters: *external* parameters (given only by the user) and *internal* parameters (supplied either by the browser or by the web site, and not controlled by the user). If the action is a test on the parameter given by the user (for instance, in the authentication of a user password), then the link is activated if only if the condition of the test is verified. Otherwise, an alternative page indicating the error may be loaded. This leads to the following definition:

Definition 2. *Let* $a \in A$ *be an anchor in a page* $P_1 \in \mathcal{P}$*, with an associated action parameterized by an external condition* c*. A simple-page* guarded link $t_{a/c}$ *is defined as a pair* $t_{a/c} = (t_{a/true}, t_{a/false})$*, where* $t_{a/true} = \langle P_1, P_2 \rangle$*,* $t_{a/false} = \langle P_1, P_2' \rangle$*,* $P_2 \in \mathcal{P}$ *is the page loaded by the browser if the condition* c *is verified, and* $P_2' \in \mathcal{P}$ *is the (error) page loaded otherwise.*

Consider a link anchor with an externally parameterized action that allows the user to choose one of a fixed set of external alternatives. In this case, there exists a pre-determined set of links attached to that anchor, each to be activated according to the parameter chosen by the user. When the user clicks the anchor introducing a parameter, the action associated with the anchor uses that parameter to select one single link, allowing the browser to load just one page.

Definition 3. *Let* $I = \{0, 1, 2, ...n\}$ *be a fixed set of alternative external parameters, with* $i \in I$*. Let* $a \in A$ *be an anchor in a page* $P_1 \in \mathcal{P}$*, associated with an action parameterized on* I*. A simple-page* selection link $t_{a/i}$ *is a i-uple* $t_{a/i \in I} = (t_{a/0}, \dots, t_{a/n})$*, where* $t_{a/0} = \langle P_1, P_2^0 \rangle, \dots, t_{a/n} = \langle P_1, P_2^n \rangle$*,* $P_2^0 \in \mathcal{P}$ *is the page loaded by the browser if the selection* 0 *is chosen by the user before clicking the anchor,* $P_2^1 \in \mathcal{P}$ *is the page loaded by the browser if the selection* 1 *is chosen, and so on.*

In this work, all the web pages mentioned above are said to be *static* in the sense that their content is navigation independent, that is, in any user navigation, if the same clicks are performed, with the same selected parameters, then the same links are activated targeting always the same pages with the same contents.

However, an action associated to a link anchor may have an internal parameter supplied by the system, which may not be controlled by the user. In this case, when the user clicks the anchor, there exist a (possibly infinite) set of links, each one of which may be activated, according the way the action processes that parameter. Different pages with different contents can be loaded by the browser in different user navigations, even if the same parameters are selected by the user. In this work, such pages are called *dynamic pages*.

It is important to remark that the notion of static and dynamic pages formalized in this work is an abstraction from the notion of static and dynamic pages commonly used to refer to the technique of implementing web pages. Since the set of all possible pages with all possible contents is a pre-fixed set $\mathcal{P}$, our concept of dynamic page is related to the way the user reads a page $P \in \mathcal{P}$, and not to the way that the page is constructed (since they are conceived as pre-existing in the set of all possible pages). A navigation through static pages is fully controlled by the user while a navigation trough dynamic pages may depend upon some responses of the system, and thus look non-deterministic to the user.

Definition 1 is extended to consider frame sets (Def. 2 and Def. 3 can also be extended to consider frame sets):

Definition 4. *A* frame-set link t_a *attached to an anchor* $a \in A$ *in a page* $P_1 \in \mathcal{P}$ *of a frame set* FS_1*, and pointing to a page* $P_2 \in \mathcal{P}$ *of a frame set* FS_2*, is defined as a 4-uple* $t_a = \langle P_1, FS_1, P_2, FS_2 \rangle$*.*

Now the notion of *web site* is finally formally defined.

Definition 5. *A* web site of simple pages *is a directed graph* $W = (P, T)$*, where* $P \subseteq \mathcal{P}$ *is a non empty set of (possibly dynamic) simple pages and* T *is a set of simple-page links attached to anchors in pages of* P*.*

Definition 6. *Let* $P \subseteq \mathcal{P}$ *be a non empty set of (possibly dynamic) web pages. A* web site of frame sets *on* P *is a directed graph* $W = (FS_P, T)$*, where* FS_P *is a non empty set of frame sets exhibiting pages of* P*, and* T *is a set of frame-set links attached to anchors in pages of* P*.*

Definition 7. *Let* $W = (P, T)$ *be a web of simple pages,* $P' \subseteq P$*,* $P' \neq \emptyset$ *and* $T' \subseteq T$*.* $W' = (P', T')$ *is said to be a sub-site of* W *whenever* $P_1 \in P' \Rightarrow P_2 \in P'$*, for every* $t_a = \langle P_1, P_2 \rangle \in T'$*.*

Definition 7 can be extended to consider sub-sites of web sites of frame sets.

3 Navigation Automaton

A navigation automaton is a non-deterministic finite automaton [7, 9]. The states, called *navigation states*, represent the pages (or frame sets) of the web site, and the transitions, called *navigation transitions*, represent the links between the pages. The input alphabet of a navigation automaton is an event alphabet, composed basically by one kind of event: the user's clicks on anchors of web-page links.

Navigation automata are usually non-deterministic, for two reasons. First, clicking on a same anchor in two different time instants can lead to two different navigation states (pages or frame sets) due to different conditions in the process that dynamically generates the web site pages. Secondly, empty transitions may occurs, being transitions due to the passage of time, and not to a user click.

Each navigation transition has an *anchor*, a *source state* – representing the page or frame set where the anchor is located, and a set of *target states* – representing the set of pages or frame sets that may be loaded by the browser when the transition is activated by a user click (see Fig. 1). Therefore, it is possible to define a navigation transition relation to model a user's action when she is navigating.

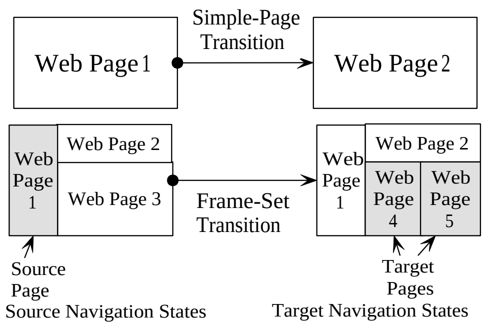

Fig. 1. Different kinds of navigation transitions

A navigation automaton usually does not present any final (i.e., recognizing) state, but it may have several *stop states*, which are states that are not sources of any transition. On the other hand, *application-oriented final states* may exist, that acknowledge a sequence of user's clicks leading to a successful end in a web application (such as, for instance, a successful buy in a virtual store)[6].

A navigation automaton is said to be *self-contained* if it presents no external transitions, that is, no transition that has the source anchor in the reference site and target page in an external site. The automaton of Fig. 2 is not self-contained.

[6] Navigation automaton with application-oriented final states are not considered in this paper.

4 Simple-Page Navigation Automata

Definition 8. *A simple-page navigation automaton of a web site $W = (P', T')$ of simple static pages is a structure*

$$SPNA = (P, S_{ext}, T, T_{ext}, X, f, Home, Stop) \ , \tag{1}$$

where:

1. *P is a finite non empty set of possible simple-page navigation states, representing the set $P' \subseteq \mathcal{P}$, $P' \neq \emptyset$, of simple static pages of W;*
2. *S_{ext} is the finite (possibly empty) set of external navigation states, representing external web sites accessible from W;*
3. *T is the finite set of simple-page transitions $t_a : s_{t_a} \longmapsto Tg_{t_a}$, with the source state $s_{t_a} \in P$ and the set of target states $Tg_{t_a} \subseteq P$, representing the set T' of simple-page links of W;*
4. *T_{ext} is the finite set of external transitions $t_a : s_{t_a} \longmapsto Tg_{t_a}$, with $s_{t_a} \in P$ and $(Tg_{t_a} \cap P) = \emptyset$, representing the links pointing to external web sites;*
5. *X is the finite set of actions associated to the transitions, including the empty action denoted by ε;*
6. *f is the partial simple-page navigation transition function*[7]

$$f = P \times (T \cup T_{ext}) \longrightarrow \wp(P \cup S_{ext}) \times X \ , \tag{2}$$

   associating each pair (s_{t_a}, t_a), where s_{t_a} is the source of the (possibly external) transition t_a attached to the anchor $a \in A$, to a pair (Tg_{t_a}, x), denoted by $Tg_{t_a}[x]$, where Tg_{t_a} is the set of target states, and x is a (possibly empty) action associated to t_a;
7. *$Home$ is the initial state of the navigation automaton and $Stop$ is the (possibly empty) set of stop states.*

Example 1. Figure 2 shows the graph of a simple-page navigation automaton with transitions representing some of the different kinds of links defined in sect. 2: *simple-page transitions* (t_2, t_3, $t_5, \ldots$), *external transition* (t_7), *simple-page self-transitions* (t_1 and t_8), *simple-page internal transition* (t_9), *simple-page guarded transition* (t_4), *simple-page transitions with associated actions* (t_1, t_6, t_7 and t_8).

Example 2. The graph of Fig. 2 corresponds to the navigation transition function $f = P \times (T \cup T_{ext}) \longrightarrow \wp(P \cup S_{ext}) \times X$ of the simple-page navigation automaton $SPNA = (P, S_{ext}, T, T_{ext}, X, f, Home, Stop)$, where $P = \{\text{Home, P1, P2, P3}\}$, $S_{ext} = \{\text{External Site}\}$, $T = \{t_{1_{a_1}}, \ldots, t_{6_{a_6}}, t_{8_{a_8}}, t_{9_{a_9}}\}$, $T_{ext} = \{t_{7_{a_7}}\}$, $X = \{\varepsilon, \text{e-mail, download, new window}\}$, $Stop = \emptyset$, where $a_1, \ldots, a_9 \in A$. The transition function f can also be given by a double entrance table (see Table 1).[8]

[7] $\wp$ is the powerset operator.

[8] The empty action ε is not represented, since it can be understood from the context. Anchors are not represented when they are not important in the context.

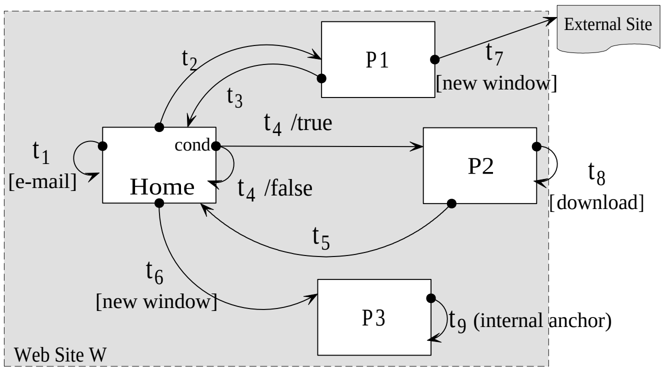

Fig. 2. Simple-page navigation automaton with different kinds of simple-page navigation transitions

Table 1. Table of the navigation transition function f given in Fig. 2

Navigation States	Transitions	f
Home	t_1	Home[e-mail]
Home	t_2	P1
Home	$t_4/cond$	{P2,Home}
Home	t_6	P3[new window]
P1	t_3	Home
P1	t_7	External Site[new window]
P2	t_5	Home
P2	t_8	P2[download]
P3	t_9	P3 (internal anchor)

5 Frame-set Navigation Automata

In this section we introduce the *frame-set navigation automata*. A matrix-oriented notation is introduced to represent the structure of the frame sets of the navigation states of a frame-set navigation automaton.

The main division of a frame-set structure is indicated externally, and the secondary divisions, if they exist, are indicated internally in the matrix, in a recursive way.

The division in n rows of a frame-set is denoted using brackets. We denote navigation states in the row structure, by indicating the page exhibited in each row. For example, for $n > 1$ rows $P_1, P_2, \ldots, P_n \in \mathcal{P}$, one has

$$\begin{bmatrix} P_1 \\ P_2 \\ \vdots \\ P_n \end{bmatrix} .$$

The division in m columns of a frame set is denoted using angled brackets. We denote navigation states in the column structure, by indicating the page exhibited in each column, using commas as separators. For example, for $m \geq 1$ columns $P_1, P_2, \ldots, P_m \in \mathcal{P}$, one has $\langle P_1, P_2, \ldots, P_m \rangle$.

Example 3. The navigation states FS 1, FS 3 and FS 5 of the frame-set navigation automaton of Fig. 3 are represented as:

$$\left\langle \text{P1}, \begin{bmatrix} \text{P2} \\ \text{P3} \end{bmatrix} \right\rangle, \left\langle \text{P1}, \begin{bmatrix} \text{P2} \\ \langle \text{P5, P6} \rangle \end{bmatrix} \right\rangle \text{ and } \langle\ \text{P9}\ \rangle. \tag{3}$$

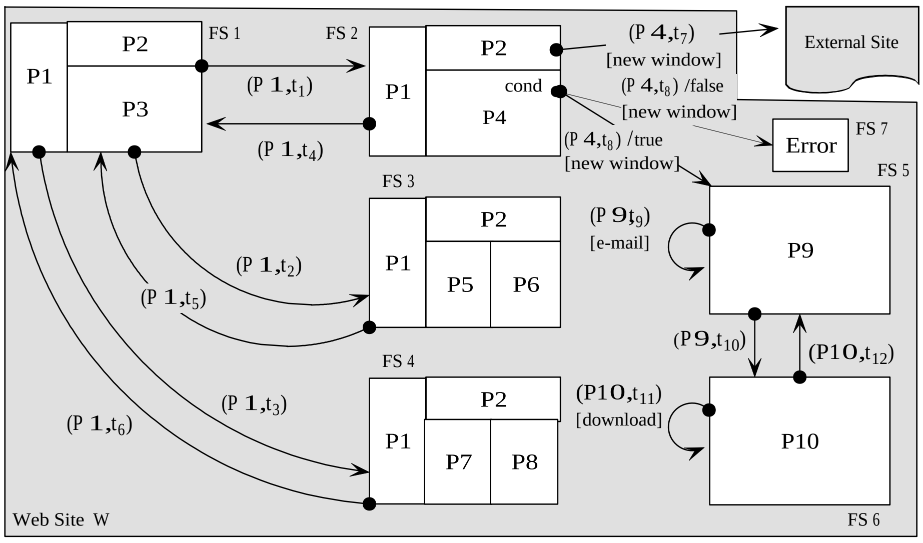

Fig. 3. Frame-set navigation automaton

Each defined frame-set transition attached to anchor $a \in A$ is denoted by a map $t_a : s_{t_a}/sp_{t_a} \longmapsto Tg_{t_a}/tgp_{t_a}$, where s_{t_a} is the matrix structure of the source state of t_a, indicating, at least, in which frame its source page sp_{t_a} is located, and Tg_{t_a} is the set of matrix structures of the target states tg_{t_a} of t_a, indicating, at least, where its target page or frame set of target pages tgp_{t_a} is located.

Example 4. The frame-set transitions $t_{1_{a_1}}$ and $t_{2_{a_2}}$, with $a_1, a_2 \in A$, of the frame-set automaton of Fig. 3 are represented as[9]:

$$t_{1_{a_1}} : \left\langle \text{P1}, \begin{bmatrix} - \\ - \end{bmatrix} \right\rangle \longmapsto \left\langle -, \begin{bmatrix} - \\ \text{P4} \end{bmatrix} \right\rangle \text{ and} \tag{4}$$

[9] In the graph of Fig. 3 and in the Table 2, each frame-set transition t_a presents an additional information, namely the indication of its source page sp_t, and is denoted as $t_a \equiv (sp_t, t)$, omitting the anchor a.

$$t_{2a_2} : \left\langle \text{P1}, \begin{bmatrix} - \\ - \end{bmatrix} \right\rangle \longmapsto \left\langle -, \begin{bmatrix} - \\ \langle\, \text{P5}, \text{P6}\,\rangle \end{bmatrix} \right\rangle \ . \tag{5}$$

Definition 8 is generalized to define the concept of frame-set automaton:

Definition 9. *A frame-set navigation automaton of a web site $W = (FS'_P, T')$ of static frame sets on $P \subseteq \mathcal{P}$ is a structure*

$$FSNA = (FS_P, S_{ext}, T, T_{ext}, X, f, \mathit{FHome}, \mathit{Stop}) \ , \tag{6}$$

where:

1. *FS_P is a finite non empty set of possible frame-set navigation states, representing the set $FS'_P \neq \emptyset$ of static frame sets on $P \neq \emptyset$ of W;*
2. *S_{ext} is the finite (possibly empty) set of external navigation states, representing external web sites accessible from W;*
3. *T is the finite set of frame-set transitions $t_a : s_{t_a}/sp_{t_a} \longmapsto Tg_{t_a}/tgp_{t_a}$, with $s_{t_a} \in FS_P$ and $Tg_{t_a} \subseteq FS_P$, representing the set T' of frame-set links of W;*
4. *T_{ext} is the finite set of external transitions $t_a : s_{t_a}/sp_{t_a} \longmapsto Tg_{t_a}/tgp_{t_a}$, with $s_{t_a} \in FS_P$ and either $(Tg_{t_a} \cap P) = \emptyset$ or $tgp_{t_a} \in (\mathcal{P} - P)$, representing the links pointing to external web sites;*
5. *X is the finite set of actions associated to the transitions, including the empty action denoted by ε;*
6. *f is the partial frame-set navigation transition function*

$$f = FS_P \times (T \cup T_{ext}) \longrightarrow \wp(FS_P \cup S_{ext}) \times X \ , \tag{7}$$

   associating each pair (s_{t_a}, t_a), where s_{t_a} is the source state of the frame set transition t_a attached to the anchor $a \in A$, to a pair (Tg_{t_a}, x), denoted by $Tg_{t_a}[x]$, where Tg_{t_a} is the set of target states and x is the (possibly empty) action associated to t_a;
7. *FHome is the initial navigation state and Stop is the set of stop states of the frame-set navigation automaton.*

Example 5. The graph of Fig. 3 corresponds to the frame-set navigation transition function $f = FS_P \times (T \cup T_{ext}) \longrightarrow \wp(FS_P \cup S_{ext}) \times X$ of the frame-set navigation automaton $FSNA = (FS_P, S_{ext}, T, T_{ext}, X, f, \text{FHome}, \mathit{Stop})$, where $FS_P = \{\text{FS } 1, \ldots, \text{FS } 7\}$ (some of the navigation states were shown in Example 3), $P = \{\text{P1}, \ldots, \text{P10}\}$, $S_{ext} = \{\text{External Site}\}$, $T = \{t_{1a_1}, \ldots, t_{6a_6}, t_{8a_8}, \ldots, t_{12a_{12}}\}$, $T_{ext} = \{t_{7a_7}\}$, with $a_1, \ldots, a_{12} \in A$, $X = \{\varepsilon, \text{e-mail}, \text{download}, \text{new window}\}$, $\text{FHome} = \text{FS } 1$ and $\mathit{Stop} = \emptyset$. The frame-set transition function f can also be given by a double entrance table (see Table 2).

Table 2. Table of the frame-set transition function f given in Fig. 3

Navigation States	Transitions	f_t
FHome	$(\mathrm{P1}, t_1)$	FS 2
FHome	$(\mathrm{P1}, t_2)$	FS 3
FHome	$(\mathrm{P1}, t_3)$	FS 4
FS 2	$(\mathrm{P1}, t_4)$	FHome
FS 2	$(\mathrm{P4}, t_7)$	External Site[new window]
FS 2	$(\mathrm{P4}, t_8)/cond$	$\{\mathrm{FS}\ 5, \mathrm{FS}\ 7\}$[new window]
FS 3	$(\mathrm{P1}, t_5)$	FHome
FS 4	$(\mathrm{P1}, t_6)$	FHome
FS 5	$(\mathrm{P9}, t_9)$	FS 5[e-mail]
FS 5	$(\mathrm{P9}, t_{10})$	FS 6
FS 6	$(\mathrm{P10}, t_{11})$	FS 6[download]
FS 6	$(\mathrm{P10}, t_{12})$	FS 5

6 Dynamic Navigation Automaton

The general idea of dynamic navigation automata was inspired by the statecharts of UML [1, 8].

The target state of a dynamic navigation transition is a collection of dynamic states, each representing one possible outcome page of the dynamic transition. The target state may be of one of two types:

1. *simple dynamic navigation state*: a collection of independent dynamic simple-page states or dynamic frame-set states, as shown in Fig. 4;
2. *compound dynamic navigation state*: a collection of nestled dynamic simple-page states or dynamic frame-set states, as shown in Fig. 5 and Fig. 6.

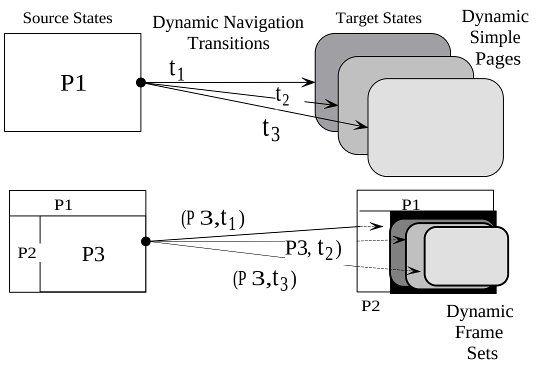

Fig. 4. Dynamic transitions with target given by a collection of simple dynamic navigation states: dynamic simple-page or dynamic frame-set states

A *compound dynamic navigation state* can be one of the following:

1. A collection of *sequential* nestled dynamic navigation sub-states: a sequence of simple dynamic navigation sub-states, with dynamic simple-page transitions, nestled together to form the compound state (as, for instance, the set of result pages produced by a query to a search engine). See Fig. 5.

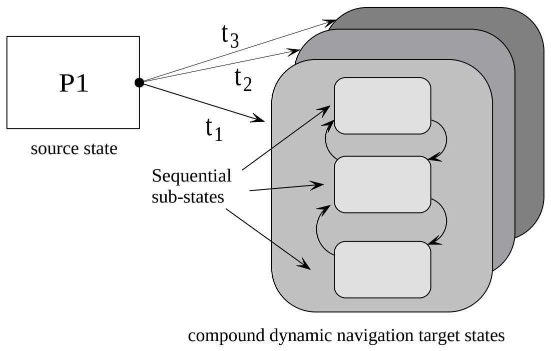

Fig. 5. Dynamic transitions with target given by a collection of compound dynamic navigation states with sequential sub-states

2. A collection of *concurrent* nestled dynamic navigation sub-states: a sequence of dynamic frame sets, exhibiting several dynamic pages at the same time, with dynamic frame-set transitions, nestled together to form the compound state. See Fig. 6.

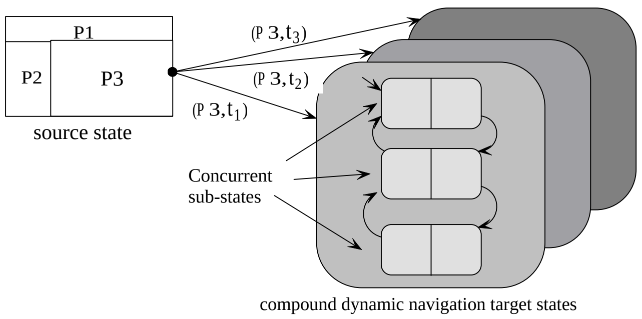

Fig. 6. Dynamic transitions with target given by a collection of composed dynamic navigation states with concurrent sub-states

Nestled sub-states may be considered as an equivalence class of sub-states with respect to the result of the internal-parameterized action that is associated with the the dynamic transition anchor.

In the following, the definition of dynamic navigation automaton is introduced considering sites with simple dynamic pages. This approach can be generalized to model sites with frame sets.

Definition 10. *A dynamic simple-page navigation automaton of a web site* $W = (P', T')$ *of dynamic simple pages is a structure*

$$DNA = (P, DP, S_{ext}, T, DT, T_{ext}, X, Df, Home, Stop) \ , \tag{8}$$

where:

1. *P is a finite non empty set of static simple-page navigation states and DP is a (possibly empty) infinite set of (simple or composed) simple-page dynamic navigation states, where $P \cup DP$ represents the set $P' \subseteq \mathcal{P}$, $P' \neq \emptyset$, of simple pages of W;*
2. *S_{ext} is the finite (possibly empty) set of external navigation states, representing external web sites accessible from W;*
3. *T is the finite set of static transitions $t_a : s_{t_a} \longmapsto Tg_t$, with $s_{t_a} \in P$ and $Tg_{t_a} \subseteq P$, and DT is the (possible) infinite set of dynamic transitions $dt_a : s_{dt_a} \longmapsto Tg_{dt_a}$, with $s_{dt_a} \in P \cup DP$ and $Tg_{dt_a} \subseteq DP$, where $T \cup DT$ represents the set T' of simple-page links of W;*
4. *T_{ext} is the finite set of external transitions;*
5. *X is the finite set of actions associated to the transitions;*
6. *Df is the partial dynamic simple-page navigation transition function*

   $$Df = (P \cup DP) \times (T \cup DT \cup T_{ext}) \longrightarrow (P \cup DP \cup S_{ext}) \times X \ , \tag{9}$$

   associating each pair (s_{l_a}, l_a), where s_{l_a} is the source of the (possibly dynamic) transition l_a attached to the anchor $a \in A$, to a pair (Tg_{l_a}, x), denoted by $Tg_{l_a}[x]$, where Tg_{l_a} is a set of (possibly dynamic) target states, and x is a (possibly empty) action associated to l_a;
7. *$Home$ is the initial state of the dynamic navigation automaton and $Stop$ is the (possibly empty) set of stop states.*

7 Sub-Site Automaton

In this section we introduce navigation automata associated to sub-sites of simple pages. The approach can be generalized to consider sub-sites of sites with frame sets.

The main characteristic of the sub-site automaton is to be self-contained in relation to the main navigation automaton, except probably by one special transition enabling the return to the outer automaton (see Fig. 8 and Fig. 7).

[9] In dynamic simple-page navigation automata, only states composed of sequential sub-states are considered.

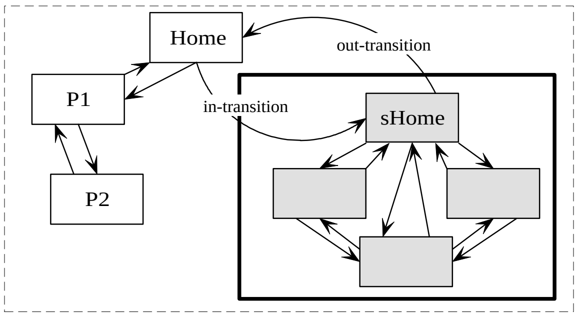

Fig. 7. A navigation automaton with a sub-site automaton

Definition 11. *Let $SPNA = (P, S_{ext}, T, T_{ext}, X, f, Home, Stop)$ be a simple-page navigation automaton of a web site $W = (P', T')$ of simple pages with a sub-site $sW = (sP', sT')$. A sub-site automaton SS of $SPNA$, denoted by $SS \Subset SPNA$, is a structure $\langle SS, it, ot\rangle$, where*

$$SS = (sP, sS_{ext}, sT, sT_{ext}, sX, sf, sHome, sStop) \tag{10}$$

is a simple-page navigation automaton and it and ot are simple-page navigation transitions, such that:

1. *$sP \subseteq P$ is the finite non empty subset of possible simple-page navigation states, representing the set $sP' \subseteq P' \subseteq \mathcal{P}$ of simple static pages of sW;*
2. *$sS_{ext} \subseteq S_{ext}$ and $sT_{ext} \subseteq T_{ext}$;*
3. *$sT \subseteq T$ is a finite subset of simple-page transitions $t_a : s_{t_a} \longmapsto Tg_{t_a}$, with $s_{t_a} \in sP$ and $Tg_{t_a} \subseteq sP$, representing the set sT' of simple-page links of sW;*
4. *$sX \subseteq X$ is the finite subset of actions associated to the transitions $t_a \in (sT \cup sT_{ext})$;*
5. *$sf = f|_{sP\times(sT\cup sT_{ext})}$ is the restriction of the partial simple-page navigation transition function f to $sP \times (sT \cup sT_{ext})$, that is,*

$$sf = sP \times (sT \cup sT_{ext}) \longrightarrow \wp(sP \cup sS_{ext}) \times sX \ ; \tag{11}$$

6. *$sHome \in sW$ is the initial state and $sStop \subseteq Stop$ is the set of stop states of the navigation automaton SS;*
7. *$it_a : (P - sP) \rightarrow sP$ is the in-transition usually given by $Y \in (P - sP) \mapsto sHome$, which is the simple-page navigation transition that permits the access to the sub-site automaton SS from the outer automaton $SPNA$, usually through its initial navigation state sHome;*
8. *$ot_a : sP \rightarrow (P - sP)$ is the out-transition usually given by $sHome \in (P - sP) \mapsto Y \in (P - sP)$, which is an optional simple-page navigation transition that permits to return to the outer automaton $SPNA$, usually through the the initial navigation state sHome of SS.*

In the items *(vii)* and *(viii)* of Def. 11, it is also possible to consider a set of in-transitions and a set of out-transitions. The in-transition (out-transition) can have its target (source) state in other navigation state than sHome.

The items *(ii)* and *(iii)* of Def. 11 mean that the navigation automaton SS is closed to the transitions $t_a \in (sT \cup sT_{ext})$, that is, every transition with source in the sub-site (except the out-transition ot_a) has target in the sub-site. Therefore, a SS is a self-contained navigation automaton (see Sect. 3) with respect to $SPNA$.

A sub-site automaton is *encapsulated* if its navigation states are not the target of any transition (except by the in-transition) whose source is outside of the sub-site automaton. An encapsulated sub-site automaton can be collapsed and represented by its initial state sHome. Figure 8 shows the graph of the navigation transition function of a navigation automaton and its encapsulated sub-site automaton, whose respective navigation transition function (hidden in this graph) is shown in Fig. 7.

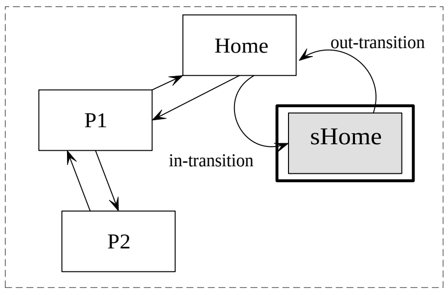

Fig. 8. A navigation automaton with an encapsulated sub-site automaton

8 Applications

In this section, we show how the automata-based navigational model can help the work on the requirement analysis of a web-based application using UML – Unified Modelling Language [1, 4, 8]. We also give a brief discussion of the application of the model in the analysis of a user navigation in a web-based educational system.

8.1 Navigation Automata Representing Scenarios of Use Cases

Use cases, first put forward in [8], are a powerful technique for capturing and expressing the interaction between system users (called *actors*) and the system itself. The term *actor* represents a generic user role. In this work, the term *user* denotes people navigating on the web site. Users may behave as many different actors in the web site system. A *use case* corresponds to a way of usage of the web site by an actor, that is, a pattern of interaction between the user and the web site. A *scenario* is a sequence of steps describing an interaction between the actor and the web site – the use case. Sometimes, there are different navigation alternatives resulting in different scenarios for the same use case. A use case is, therefore, a collection of scenarios related to a same user objective when interacting with the web site. There is always a main scenario, called *primary* scenario. The other scenarios are called *alternative* scenarios, or alternative paths.

We remark that a use case is written to express what a system should do, without constraining how it should do it. All of the behavior is presented in the form of observable results. In UML, relationship among use cases are usually documented in a use case diagram. The complete collection of use cases forms a use case model. The primary scenario is usually presented in textual form, as a labelled sequence of steps describing the sequence of transitions activated by the user. Alternative paths are described bellow the primary scenario. In the methodology for web applications presented in [4], these scenarios are usually diagrammed in sequence diagrams, a specific type of interaction diagram, which emphasize the time line. However, this methodology does not seem to have a special abstraction for representing the navigation in web sites.

Therefore, the navigation automata can be used as a mathematical formalism to represent in an intuitive way all possible scenarios (the primary scenario and all possible alternatives paths) of the use cases for a web site, describing all possible interactions between the site and the actors. In this sense, navigation automata are a means for the formal operational specification of use case models of web sites. The authors have used navigation automata for the specification of the navigation in the ENSINET[10]– a web-based educational system [5, 11].

8.2 The Control Language of a Navigation Automata

The set of all possible sequences of user clicks is the *control language* of a navigation automata, in the sense that the automaton is driven ("navigated") by those sequences along the possible paths of its state diagram. Such language is a representation of the user behavior and another possible formalization of use case scenarios. We shall explore the concept of control language in further paper.

In the system ENSINET, by registering the "words" of the control language that are generated by the navigations of a particular student, the system ENSINET/SPY is able to monitor the user behavior when navigating in the system. This is done in order to compare an actual student behavior with an expected or acceptable behavior. The analysis of the "words" generated by the users are to be used in the new features that are now been developed for the system ENSINET: (i) ENSINET/AVAL, a system to allow intuitive evaluations of the interactions performed by the users in the system, and (ii) ENSINET/ADAPT [6], a system to create user adaptive encapsulated sub-sites.

[10] The application of navigation automata to the ENSINET can be found at `http://gmc.ucpel.tche.br/ensinet/download.htm`. The system can be accessed at `http://ensinet.ucpel.tche.br`.

9 Conclusion

The automata-based navigational model for modelling web sites from the user's point of view can express all possible alternative sequences of transitions activated by the user during her navigation. In this sense, it describes all possible interactions between a web site user and the web site itself, describing the behavior of the web site as viewed from the outside.

The navigation automata can be used as a mathematical formalism to represent in a intuitive way all possible scenarios of the use cases for a web site, describing all possible interactions between the site and the actors. In this sense, navigation automata are means for the formal operational specification of use case models of web sites. Besides, they can be adapted to include probabilistic and statistical information on the user behavior.

The exploration of the ideas of using the analysis of the (probabilistic) navigation automata control language in adaptive hypermedia [2, 3, 12] is a subject of further work. Concerning other practical applications, we expect that navigation automata can be used as a foundation for development tools able to generate automatically web site navigation structures.

References

[1] G. Booch, J. Rumbaugh, and I. Jacobson. *The Unified Modeling Language User Guide*. Addison-Wesley, Reading, 1999.

[2] P. Brusilovski. Methods and techniques of adaptive hypermedia. *User Modeling and User Adapted Interaction*, 6(2-3):87–129, 1996. Special issue on adaptive hypertext and hypermedia.

[3] P. Brusilovski. Adaptive hypermedia. *User Modeling and User Adapted Interaction*, 11(1):87–110, 2001.

[4] J. Conallen. *Building Web Applications with UML*. Object Technology Series. Addison-Wesley, Reading, 2000.

[5] G. P. Dimuro, A. C. R. Costa, and F. P. M. Rodrigues. *Projeto ENSINET*. Escola de Informática/UCPel, `http://gmc.ucpel.tche.br/ensinet`, 1998.

[6] G. P. Dimuro, L. A. M. Palazzo, A. C. R. Costa, and R. M. Miranda. Sites internos adaptavivos para o sistema ensinet. In *Workshop de Software Livre*, pages 102–105, Porto Alegre, 2002. SBC.

[7] J. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Reading, 1979.

[8] I. Jacobson, P. J. Christerson, and G. Övergaard. *Software Engineering: A Use Case Driven Approach*. Harlow, 1992.

[9] H. R. Lews and C. H. Papadimitriou. *Elements of the Theory of Computation*. Englewood Cliffs, 1998.

[10] K. Lodaya and R. Ramanujam. An automaton model of user-controlled navigation on the web. In Univ. Western Ontario, editor, *CIAA Conference*, Canada, 2000.

[11] R. M. Miranda, G. P. Dimuro, and A. C. R. Costa. Ensinet: Um ambiente de suporte ao ensino integrado dos fundamentos matemáticos da computação. In *Workshop de Software Livre*, pages 40–42, Porto Alegre, 2001. SBC.

[12] L. A. M. Palazzo and A. C. R. Costa. Towards proactive hypermedia systems. In *III Encontro Nacional de Inteligência Artificial*, Fortaleza, 2001. SBC.