Modeling Parallel Web Browsing Behavior for Web-based Educational Systems

Martin Labaj and Mária Bieliková
Slovak University of Technology in Bratislava, Faculty of Informatics and Information Technologies
Ilkovičova 3, 842 16 Bratislava, Slovakia
{labaj, bielik}@fiit.stuba.sk

***Abstract*—When learners access web-based educational systems, they often take advantage of what current web browsers offer: multiple tabs with different pages. Whether they are solving an exercise or learning a whole new topic, they can concurrently browse other relevant objects in the educational system or even resources outside the system, on the "wild" Web. This behavior goes unnoticed or it is only estimated in the traditional web usage mining, where it is supposed that the user opens only one page at a time. However, if we could record the parallel browsing behavior, we could better understand the user's activity, goals and improve learner model employed for personalization. The paths through resources (including spawning new tabs and switching between them) of many learners also express relevance between these resources and this can be leveraged in applications such as recommendation to other learners browsing similar learning objects. In this paper, we introduce a model for parallel browsing behavior based on events tracked with client-side scripting. We realized the proposed model in the ALEF educational system, which we use in several courses and evaluate learner behavior in the system.**

I. Introduction

Current educational systems are often realized as web-based systems, allowing their users (learners) to work from a range of platforms and even remotely from their home. The learners often take advantage of current web browsers and open multiple resources in parallel, be it in multiple browser tabs or multiple browser windows. In this way, learners can open multiple objects (web pages, learning objects) from the educational system and switch between them. If we could record and analyze this parallel browsing, we could better understand learners' behavior and leverage their activity for information personalization.

Educational systems often provide different types of objects: new topics are introduced with video-clips, presentation slides, or coursebook texts; user's progress is tracked with quizzes, questions or tasks. In this paper we present our results on the ALEF system. ALEF stands for Adaptive LEarning Framework [1] and it is an adaptive educational system being used at the Slovak University of Technology in Bratislava in several courses: Procedural Programming, Functional and Logic Programming, and Software Engineering.

ALEF offers several types of *learning objects*. *Explanations* present basic educational content explaining a part of topic, e.g. recursion in procedural programming. *Exercises* are tasks that users can solve in the system and submit their solutions for evaluation, e.g. "Create a program which…", "Draw a UML model of…". *Questions* are quizzes, where the interaction is short – user has to select one or multiple correct answers, put answers in the correct order, or finish a free text answer.

When a learner reads for example a math coursebook and starts solving a mathematical problem, it is natural that she looks up relevant formulas or theorems presented in the current chapter. This behavior can be similar in an educational system – when a learner starts some exercise and needs to look up additional information, she can open corresponding explanations or even external resources on the Web in other tabs and switch back and forth while constructing the solution. Similarly, when the user is learning a new topic and she starts with an Explanation object, she may switch to relevant exercises and questions to practice the presented information.

Relevant exercises or explanations in these examples can be selected manually by a teacher during course authoring or even automatic methods can be applied, such as content-based methods using similarities between learning objects. However, knowing users' browsing behavior allows us to better model the user, what she reads in what situations and the better user model alone can be used for better information personalization for the user. For example, in a recommendation based on the learner's current knowledge combined with time left for learning and her learning goals [2], it is very important to accurately model these goals. By recognizing known user tasks in the parallel browsing, e.g. the user is opening exercises into new tabs, which she keeps open as bookmarks until she proceeds to solve them later, we can better model user goals.

Users can also find learning objects and external sources relevant to their current activity outside of the relevance expressed in the domain model of the content and outside of relations discovered by content-based methods or created by the teacher. The relevant explanations and external sources can be recommended to other users struggling on the same exercise on which the user opened them. Relevant exercises can be recommended to users reading an explanation, the domain model can be augmented with discovered relevance, etc.

However, the traditional view on user's behavior in the web usage mining is more based on the linear browsing model. It is often assumed that when a page is visited (loaded), the user gives it all her attention until visiting another page, replacing the previous one. However, in the more accurate parallel browsing view, multiple pages are opened concurrently and the user's attention is split between them. User modeling from server-side information is most affected by this difference, because at the server side, we do not have enough information to discern how the page was opened (In a new tab or replacing the current page?).

On the client side, we can exactly track what a user does by augmenting the browsing software, but this is limited only to users in a closed experiment in a lab where the software is provided by the researchers, or to a portion of users that are willing to install tracking extensions. The necessity of additional software limits its use in real settings and it does not provide information on all users.

We proposed a model for parallel browsing behavior based on events observed by client-side scripting. As such, it does not need installation of any additional software (e.g. browser extensions). In this way, we capture behavior of all learners in a web-based system, rather than a limited number of them in a closed study. Users also act more naturally than when they know that they are to be tracked, which they do know when we require them to install a tracking browser extension. We show how to detect which resources are being used in parallel by a user. The proposed model was realized in a live instance of the ALEF framework being used by university students during semester. We evaluate students' parallel browsing activities while learning in ALEF.

II. PARRALEL WEB BROWSING STUDIES

Opening multiple windows and having multiple web pages open at the same time have been supported by web browsers for a long time since their soon versions. Afterwards, tabbing interfaces were added, which allowed multiple pages to be opened in the same window in a user interface elements called tabs. First, they were supported through browser extensions and later natively by browsers themselves. Since the release of Internet Explorer 7 in the 2006, all widely used browsers support tabbing. Both tabbed and multi-windowed browsing methods are very similar, the main difference is that with multiple windows a user switches between opened web pages via main taskbar of the operating system (as when switching between applications) and with multiple tabs, the switches are made via the list of tabs inside the browser.

While differences between tabbing and multi-windowing are important from the perspective of the human-computer interaction and user interfaces, in the web usage mining, we only need to know which resources (pages) are being browsed and how the user travels through them. For simplicity, we address the use of multiple browser tabs and multiple browser windows as tabbing and call both the browser windows and tabs as tabs.

Parallel browsing has been previously studied through various methods selected for given research goals. For example, in order to just survey how users work with web browsers and search engines while searching for information, a questionnaire can be used. In [3] having multiple tabs open while searching was found to be common.

There are several approaches for the user behavior tracking. Basically, we can realize user behavior tracking either as server-side, or client-side.

In *server-side tracking,* server logs contain only pageloads (target address, previous address (referrer), timestamp) and no direct or indirect information on tabbing. Parallel browsing can be only estimated when using this kind of tracking. In [4], authors introduced a model for both linear browsing (action of following a link) and parallel browsing (actions of opening a tab and switching between open tabs) and subsequently used the server-side data for modeling the user behavior. Only linear browsing is accurately observed from server data and only an estimate of possible clicktrees and paths through them can be made for parallel browsing, giving only information on how the users *could* be browsing in parallel, giving many possibilities. Authors estimated the parallel browsing behavior in the study to be in a large range of 4% to 85%.

In [5], a model based on stochastic process was applied to logs of visited pages and their referrers. An improvement was shown, but the model cannot distinguish between branching (opening multiple tabs from single page) and backtracking (opening a page in the same tab, returning back and opening another page).

In [6], all-*k*th-order (AKO) Markov model was used on tasks that were extracted from browsing history. Tasks are one of the user motivations for using the tabs (a task of *multitasking* – assuming the use of different tabs for different tasks) and an enhancement of 27.5% in recommendation was shown over traditional AKO model. However, this approach does not include any other uses of multitabbing, where the use of tabs is not correlated to tasks.

While server-side logs are available from all users by default, their limited data restricts us only to estimations of user behavior.

Client-side extensions or *plug-ins* are used to augment the web browsing software. In this way, everything about the user activity can be recorded, since the browser sees all its opened tabs and operations with them. However, requiring the augmentation of the browser to take place, this approach is more suitable for studies, where the users voluntarily install the extension or where the users are seated to computers with software prepared by the researchers (controlled experiments). In fact, this approach has been used as a basis for several studies on tabbing.

Combined clickstream logging and diaries have been used in [7] with 21 participants to study user work with multiple tabs. In [8], the same approach was used with 20 participants to study revisitation in the context of the tabs. In [9], a browser plug-in was used with 10 participants to analyze exploratory tasks. During a search session, users opened at most 2 additional tabs. Large logs from a plug-in were used in [10] (60 billion pageviews from 50 million users) and authors looked for situations where a user opens a page and the tab identifier changes.

In *client-side scripting*, a JavaScript code is embedded within a page and it can see how the user works with the page (where she clicks) or when she leaves. This approach collects information from every user, but the code is always running within a sandbox of the single page, in which it was embedded. This page is present in a single tab and as such, the tracking code sees no information about other tabs. In a recent study [11], tracking code was embedded in the results page of a search engine. Presence or absence and the order of pageloads and clicks on the results were used. For example, a pageload and two following clicks mean that the result must have been opened in another tab (branching), because the source page still existed for the second click to occur. This approach is limited to detection of the initiation of parallel browsing within the results (e.g. no data is available on switching between the result pages) and there are branching situations which are not covered by this approach.

Overall, it has not been sufficiently studied how to use the available events to reconstruct the parallel browsing with the client-side scripting methods, thus covering natu-

ral behavior of all users in their own settings in an uncontrolled experiments, where they are typically not presented with augmented software.

Several studies have shown that it is important to consider, explore and exploit the parallel browsing behavior. Parallel browsing is now more common than the linear browsing. It may differ by current user tasks [10]. Users can create 4 to 22 tabs per 100 navigations while using the multiple tabs for reminding, scanning through search results, multitasking, comparing pages, or even bookmarking [7], and all these usages are relevant to the learning path. Tracking tab usage is also important when considering traditional web usage metrics, e.g. the revisitation.

In [7], 30% of created tabs were selected multiple times, in large contrast to traditional view based on linear browsing where each loaded page is viewed exactly once. A loaded page can be never viewed (a false visit in the traditional approach), or it can be left and visited again by switching out of its tab and back to it again (a revisit invisible to traditional approach). Study in [8] reports traditionally computed revisitation rate 39.3% and effective revisitation with tabs actually 59.6%.

III. Tracking the Parallel Browsing Behavior

We proposed a model for parallel browsing, in which we use events observed via client-side scripting.

A. Tracking Scheme

Commonly tracked information about opening a page (pageload) and from which page it was navigated to (referrer) is insufficient to tell when the page was closed. Based on the server-side data, we can only estimate that the page in question was closed in some time after the last click on a link in it occurred (the page had to still exist when the user has opened a link from it). It is not known whether a referred page was opened in a new tab or it reused the same tab. For this, it is necessary also consider the page unload event, which can only be tracked at the client side.

Suppose we have pages labeled p_1, p_2, etc. and tabs t_1, t_2, etc. We mark the pageload event with the address of a page p and a referrer r as $\mathrm{PL}(p, r)$ and the page unload event of a page p as $\mathrm{PU}(p)$. The parallel browsing is modeled from occurrences of the following sequences:

1) Opening new pages

User can open a new page by typing its address into address bar, by using bookmarks, clicking links in external applications, etc.:

- $\mathbf{O_1}$ Opening a new page p_1 in existing empty tab t_1 (type-in, bookmark, etc.) → $\mathrm{PL}(p_1, 0)$
- $\mathbf{O_2}$ Opening a new page p_2 in existing tab t_1 with page p_1 open → $\mathrm{PU}(p_1)$, $\mathrm{PL}(p_2, 0)$

2) Opening pages by following links

User can follow a link in the **L**inear way (the page opens in the same tab) or with **B**ranching (parallel browsing, the page opens in new tab):

- **FL** Following a link to page p_2 from page p_1 in the same tab t_1 → $\mathrm{PU}(p_1)$, $\mathrm{PL}(p_2, p_1)$
- **FB** Following a link to page p_2 from page p_1 in tab t_1 to a new tab t_2 → $\mathrm{PL}(p_2, p_1)$

3) Closing the pages and tabs

Finally, the user can close a page or a tab at any time:

- $\mathbf{C_p}$ Closing a page p_1 → $\mathrm{PU}(p_1)$
- $\mathbf{C_t}$ Closing existing empty tab t_1 with no page → undetected
- $\mathbf{C_{pt}}$ Closing existing tab t_1 with page p_1 → C_{p_1}, C_{t_1}
- $\mathbf{O_t}$ Opening a new empty tab t_1 with no page → undetected

Actions O_t and C_t are not observed, since no pages (with tracking script) exist in a tab t_1 when these actions occur. However, empty tabs are not important from the point of page browsing behavior. Tabs can be also substituted with windows (or used concurrently) in the same scheme. As we cannot record which tabs are open in which windows, we treat a single-page window as another tab and multiple multi-page (multi-tab) windows as just single collection of tabs.

Fig. 1 shows an example of a parallel browsing session with three tabs t_1, t_2, and t_3. All proposed operations are illustrated. Fig. 2 contains events captured for tabs t_1 and t_2 and operations with them from Fig. 1. Following a link with branching (FB) occurs as a pageload with a referrer to existing page that has not yet been unloaded. Following a link in a linear manner (FL) occurs as a page unload and a subsequent page load with the referrer of unloaded page.

If a user opens a page manually, it occurs as a pageload with no referrer (O_1). If an existing tab with currently opened page is being reused (O_2), page unload of this page occurs just before it. Notice how O_2 and FL are similar, but they are distinguished by matched/unmatched referrer.

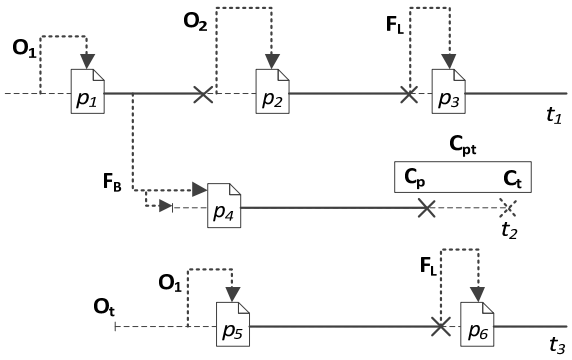

Figure 1. Visualization of actions from the presented model on an example browsing session with three tabs t_1 to t_3.

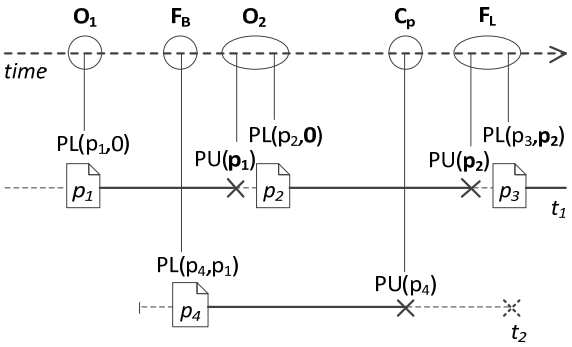

Figure 2. Events recorded for selected tabs (t_1 and t_2) from browsing session in Fig. 1. Corresponding user actions detected from these events are shown on a time axis.

B. Fragmented Spent Time

In order to track the work with opened tabs (tab switching) in addition to opening and closing of the tabs, we track fragmented spent time. Spent time is commonly tracked as a time that the user has spent on the page [12] – in the most basic form as a difference between two subsequent pageload events. We fragment the time spent and track each bringing of the page into the focus (*focus*) and each losing of it (*blur*) together with timestamps. These events can be tracked from client-side scripts included within a page. Let $\mathrm{B}(p)$ be the blur event of page p and $F(p)$ be the focus event of page p:

- **S** Switching from tab t_1 with page p_1 to tab t_2 with page $p_2 \rightarrow \mathrm{B}(p_1), \mathrm{F}(p_2)$

However, the switching can also occur to/from outside of the set of pages, which we can track – a user can switch to an empty tab with no page, or foreign page without script, or out of the browser. Since the tab switch is the event of leaving one tab and the event of arriving to another tab, such situation only disconnects the events:

1. Switching from tab t_1 with page p_1 to out of the scope $\rightarrow \mathrm{B}(p_1)$.
2. (Untracked activity: switching between untracked tabs, applications, etc. → undetected.)
3. Switching from out of the scope back to tab t_2 with $p_2 \rightarrow \mathrm{F}(p_2)$.

Fig. 3 shows an example of a possible tab switching session made on tabs t_1 and t_2 from Fig. 1. Bold line shows user navigation between the tabs. While the user opens/closes page p_4 in tab t_2, a different tab is active (t_1). The page was opened by following a link to a new tab (FB) and since the user did not switch to that tab, it must be opened "in background" (the new tab is spawned and the page is loaded in it, e.g., by using context menu or holding the CTRL key and clicking, but it is not activated until the user switches to it). When the user changes current page in the same tab, (action O_2 or FL), B and F events also occur, such pairs are in boxes. Note that tab switching and tab closing actions are independent; the user can close any tab without activating it.

When no matching focus event is recorded after a blur event has occurred, this means that the user have switched out of a browser, a pop-up dialogue may be displayed, or an untracked tab may be activated. We consider such time "out of scope".

C. Scope of the Tracked Behavior

As we implied above, we may not always be able to track all pages in all tabs – branching does not occur only within pages from a single site with our scripts included, but also between multiple sites. The presented scheme however applies to the entire set of tabs within a browser when the tracking is happening inside each tab, which can be accomplished by employing a proxy (such as one proposed in [13]), which modifies server responses and makes it possible to insert the tracking code into any page.

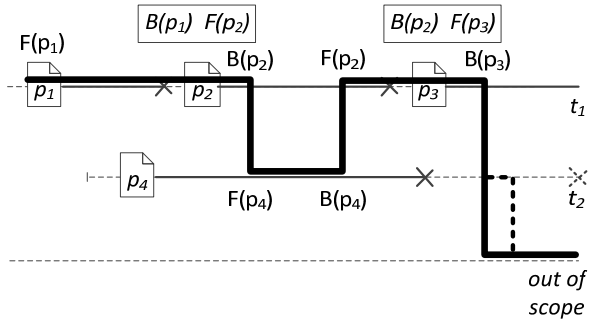

Figure 3. Example of possible switching between tabs (t_1 and t_2) with relevant events.

Without the proxy or other similar modifications, we typically do not have our pages loaded in every tab of the user's browser. We can consider such tabs "empty" for branching behavior, which also makes them "out of scope" for tab switching. However, our pages can be visited from any arbitrary web page and such web page is now considered an "empty tab". Thus when we mark tracked websites $P = \{p_1, p_2, \ldots\}$ and all other websites $Q = \{q_1, q_2, \ldots\}$, $P \cap Q = \emptyset$, we can consider any pageload event $\mathrm{PL}(p_i, q_j)$ to be equivalent to $\mathrm{PL}(p_i, 0)$.

IV. Browsing Behavior Reconstruction

We have presented our browsing behavior model with observable events assigned to various user actions. However, when analyzing the browsing behavior, the events are collected *within* the pages visited by a given user and we are interested in the reverse mapping – the reconstruction of user actions *between* the pages.

Since each page is opened in its own sandbox and exists from the script's point of view only from its load to its unload (sections bounded by "document" symbol and "X" symbol in the Fig. 1 to 3), the events we collected on a given page were represented as a vector of event-timestamp pairs that occurred during a lifetime of that page. Additional information, including timestamps, user info, etc., was also tracked (see example). Application-specific information was added to generic tracking. In a web-based educational system, we add learning object ID and course/setup ID (one learning object can be present in multiple courses or setups).

Data Collected for a Single Page: TabInfo Record

```
events_vector:
[["PL",1335002774493],["B",1335002774645],
 ["F",1335002774724],["B",1335002774787],
 ["F",1335002844169],["B",1335002874645],
 ["UL",1335002881229]]
page:
"http://alef.fiit.stuba.sk/learning_objects/112"
referrer:
"http://alef.fiit.stuba.sk/learning_objects/111"
user_agent:
"Mozilla/5.0 (Windows.."
client_timestamps: (..), server_timestamps: (..)
user: 149, learning_object: 112, course: 3
```

These vectors were combined together by the browsing reconstruction algorithm. Note that the algorithm treats user actions defined for basic tracking scheme (see Section III.A) independently from actions for switching between tabs (see Section III.B). As noted, a user can operate other tabs, i.e. close them, without first switching to them. A conceptualized version of the algorithm is presented below.

Algorithm for Browsing Behavior Reconstruction

```
function GetBrowsingBehavior(user)
 var behavior = [];
 var global_vector = [];
 foreach info in GetUserTabInfos(user)
```

```
   foreach ev in info.events_vector
     global_vector.add(ev.name, ev.timestamp,
                     info.page, info.referrer);
   end
 end

 SortByTimestamps(global_vector);
 foreach info in global_vector
   case info.event
   "PU":
     var next = FindAndRemoveNextPL(info.page);
     if next == NULL
      behavior.add("CP", info.page);
     else if next.referrer == NULL
      behavior.add("O2", info.page, next.page);
     else
      behavior.add("FL", info.page, next.page);
     end
   "PL":
     if info.referrer == NULL
      behavior.add("O1", info.page);
     else
      behavior.add("FB", info.referrer,
                      info.page);
     end
   "B" :
     var active = FindAndRemoveNextFocus();
     behavior.add("S", info.page, active);
   "F" :
     behavior.add("S", OUT_SCOPE, info.page);
   end
 end

 return behavior
end
```

Initially, all event vectors for a given user are combined by ordering them according to their timestamps. Next, the algorithm processes the events sequentially, looking ahead for subsequent matched events. *FindAndRemoveNext-PL(referrer)* looks for follow-up pageload event with no referrer or if a referrer is logged, it must match given parameter – current page, whose PU event is being processed. *FindAndRemoveNextFocusTarget()* looks for follow-up focus event or returns out of scope flag if it is not found within the time span or if blur event is encountered first. If the desired events are found, they are removed from further processing, therefore only remaining unmatched PL and F events are encountered later. Since the presence or absence of follow-up events differentiates user actions (e.g. FL and C_p) or out of scope operation, these functions look ahead only within a short time span.

If the look-ahead time span is too short or too long, in tab switching, either "ghost" out of scope operation would be inserted, or a real out of scope operation would go undetected. Since length of those would be comparable to the length of look-ahead time span, they are very short and nevertheless, the information between which pages has the user finally travelled disregarding the inserted out of scope operation is preserved. However, in page navigation processing, a pageload from a different tab can be found instead of absenting pageload in the same tab. Therefore names consisting of generated universally unique identifiers are assigned to individual tabs. Such tab name persists between pages in the same tab unless it is changed. The lookahead of PL and PU events is made only within events with the same tab identifier, if the identifier is available.

V. EVALUATION

We realized the proposed parallel browsing model and browsing behavior reconstruction within the Adaptive LEarning Framework (ALEF) [1]. In our experiments, ALEF was used by students both during classes and at home to learn new topics, repeat content from lectures and solve exercises. They were using various setups without supervision or specific instructions for the tabbing.

For efficiency reasons, especially since a user may do many tab switches in succession, we initialize and keep the vector of event-timestamp pairs upon a pageload and report it to the server together with other information on page unload. A disadvantage is that we lose this information when it is not possible to send the data during unload (e.g. when the user loses internet connection, or when some browsers do not trigger the unload event on specific leaving methods). We found that we did not receive the tracking information in 8.1% cases of learning object visitations (these visits were recorded in request logs, equivalent to pageloads, but not in tab tracking logs). This is acceptable, so we took this into consideration in the processing algorithm and proceeded with the described setup. Current emerging technologies such as WebSockets can be later used to have partial input available even when the complete vector could not be received.

Within the frame of study, 143 out of 254 active users (students) (56.3%) were browsing pages of ALEF system in parallel with two to eight ALEF pages opened concurrently. From all 80,566 navigations, 1,311 were made via following a link to a new empty tab (branching, FB), 5,360 were made via typing in an address or following a link from outside of the ALEF system to an empty tab and 1,628 were made by typing in an address to an existing tab with an ALEF page. The rest was made via linear browsing. Most of the branching interaction (70.8%) was performed from explanations to other explanations.

While the numbers for branching actions may seem rather low to overall number of navigations, there may be two reasons for this. First, in order to be browsing multiple resources in parallel, it is enough to open a few new tabs and then continue browsing in them in a linear way, or even open an empty tab and visit a home page of the ALEF and browse from there. This parallel existence of tabs without common point of branching one from another should be also considered in addition to the spawning of tabs. Second, the previous generation of ALEF system discouraged tabbed browsing because of stateful adaptive tools embedded in learning objects. While this constraint and discouragement was removed, we did not instruct student explicitly to do tabbing now in order not to influence their behavior towards the parallel browsing and our results show that despite previous discouragement they tried and used tabs.

Considering the tab switching, only 3 out of 254 active users never switched a tab containing an ALEF page. 215 users (84.6%) were switching intensively (occurrences of more than 5 tab switches on a page). The distribution of tab switch actions was: 50,266 switches from an out of scope operation to an ALEF page, 49,283 switches from an ALEF page to out of scope operation, 9,413 switches from an ALEF page to the same page (e.g. pop-up alerts), and 2,783 switches from an ALEF page to another ALEF page. Note that some switching from an ALEF page to another ALEF page may be "disconnected" by out of scope operation as noted before, but it can be easily joined in further processing.

Similar to branching actions, switches were most commonly performed from an explanation to another one (32.2%), however from questions and exercises, users mostly switched to explanations for help. Question-to-

explanation was 83.3% of switches from questions, exercise-to-explanation was 55.8% of switches from exercises. Given the number of students and high numbers of available learning objects, such switches were too distributed among objects in order to pair questions or exercises to explanations with enough confidence. However, considering learning object pairs with most switches between them, we were able identify new relations not explicitly represented in the domain model. E.g., in the Software engineering course, such pairs were explanations for: Sequence Diagram and Communication Diagram, Sequence Diagram and State Machine Diagram, and Data Flow Diagram and Use Case Diagram. These diagrams are often compared in the real world. Another example pair of learning objects is from Prolog course, consisting of an exercise for finding the longest sublist in any depth and an exercise for finding the maximum depth within a list.

Performed study demonstrated viability of our proposed model of parallel browsing. It allows browsing behavior reconstruction in real time and its realization by client-side scripts embedded in the web page allows detailed tracking without the need of extending the browser, which would cover only a partial group of users, nor maintaining different extensions for several different browsers.

VI. Conclusions and Further Work

Employing our proposed model for parallel browsing behavior allows monitoring the user behavior more accurately as considering the tabbing brings new patterns important for understanding the user behavior. We realized our model based on events tracked by client-side scripts embedded in a web page. At the expense of looking for combinations of events rather than tracking the tabbing explicitly via a web browser extension, it is tracked unobtrusively regardless of user's environment and software setup, making parallel browsing data ready to be easily used as a general input for a recommender system or other information personalization applications.

Our experiments with parallel browsing behavior suggests that the tabbing behavior is common within a technology enhanced learning system, even under unfavorable conditions of users being discouraged from parallel browsing in the past. It can be expected that within previously unrestricted system, or when encouraged, the parallel browsing could be even more widespread. Qualitative observation of the users both in recorded logs and personally during their work in seminars has shown that they very often use multiple tabs to prepare multiple explanations by opening them in tabs and then working through them, or they use explanations in other tabs as a help when working on exercises or questions.

Existing recommendation where the parallel browsing is in most cases completely ignored can be combined with the real browsing behavior in order to improve recommendations. New previously unexplored information sources can be also researched on real user behavior. E.g., having multiple web pages open in parallel and switching between them demonstrates various paths through them (the current recommender task of finding novel paths through resources in educational domain). By spawning new tabs and keeping the existing ones open concurrently, users show relations between the pages.

In the educational domain we can discover and track various behavioral patterns, which allow better knowledge estimation and consequently more accurate recommendation of learning objects. Tracking the behavior allows for discovering relationships between learning objects and domain terms, which describe these learning objects. For example, if a learner opens several tabs with explanations from a tab with an exercise and closes some of them while keeping others open and switching to them several times, we can assume that those learning objects explain topics (domain terms in light semantics) related to that exercise.

By combining our browsing model with a comprehensive model of monitoring (e.g. through a proxy [14]) to cover the entire browsing session, the recommender system can even take into account how users combine multiple heterogeneous web sources. Novel resources outside of an educational system, but related to the current learning object, can be selected and recommended by considering how users switched from such page of the educational system to the external pages.

Acknowledgment

This work was partially supported by grants VG1/0675/11, VG1/0971/11 and APVV-0233-10.

References

[1] M. Šimko, „Automated Acquisition of Domain Model for Adaptive Collaborative Web-Based Learning," in *IST Bulletin of the ACM Slovakia*, vol. 4, no. 2, pp. 1–9.

[2] P. Michlík and M. Bieliková, "Exercises Recommending for Limited Time Learning," *Procedia Computer Science*, vol. 1, no. 2, pp. 2821–2828, Jan. 2010.

[3] A. Aula, N. Jhaveri, and M. Käki, *Information search and reaccess strategies of experienced web users*. New York, New York, USA: ACM Press, 2005, pp. 583–592.

[4] M. Viermetz, C. Stolz, V. Gedov, and M. Skubacz, "Relevance and Impact of Tabbed Browsing Behavior on Web Usage Mining," in *2006 IEEE/WIC/ACM Int. Conf. on Web Intelligence - WI'06*, 2006, pp. 262–269.

[5] F. Chierichetti, R. Kumar, and A. Tomkins, "Stochastic Models for Tabbed Browsing," in *Proceedings of the 19th int. conf. on World wide web - WWW '10*, 2010, pp. 241–250.

[6] G. Bonnin, A. Brun, and A. Boyer, "Towards Tabbing Aware Recommendations," in *Proceedings of the First International Conference on Intelligent Interactive Technologies and Multimedia - IITM '10*, 2010, pp. 316–323.

[7] P. Dubroy and R. Balakrishnan, "A Study of Tabbed Browsing Among Mozilla Firefox Users," in *Proc. of the 28th int. conf. on Human factors in comp. systems - CHI '10*, 2010, pp. 673–682.

[8] H. Zhang and S. Zhao, "Measuring Web Page Revisitation in Tabbed Browsing," in *Proc. of the 2011 annual conf. on Human factors in computing systems - CHI '11*, 2011, pp. 1831–1834.

[9] G. Singer, U. Norbisrath, E. Vainikko, H. Kikkas, and D. Lewandowski, "Search-logger analyzing exploratory search tasks," in *Proc. of the 2011 ACM Symposium on Applied Computing - SAC'11*, 2011, pp. 751–756.

[10] J. Huang and R. W. White, "Parallel browsing behavior on the web," in *Proc. of the 21st ACM conf. on Hypertext and hypermedia - HT'10*, 2010, pp. 13–17.

[11] J. Huang, T. Lin, and R. W. White, "No Search Result Left Behind: Branching Behavior with Browser Tabs," in *Proc. of the 5th ACM int. conf. on Web search and data mining - WSDM'12*, 2012, pp. 203–212.

[12] P. Hofgesang, "Methodology for Preprocessing and Evaluating the Time Spent on Web Pages," in *2006 IEEE/WIC/ACM Int. Conf. on Web Intelligence - WI'06*, 2006, no. 1994, pp. 218–225.

[13] M. Barla and M. Bieliková, "Ordinary Web Pages as a Source for Metadata Acquisition for Open Corpus User Modeling," in *IADIS Int. Conf. WWW/Internet 2010*, 2010, pp. 227–233.

[14] T. Kramár, M. Barla, and M. Bieliková, "PeWeProxy: A Platform for Ubiquitous Personalization of the 'Wild' Web," in *Adjunct Proc. of the 19th Int. Conf. on User Modeling, Adaption, and Personalization - UMAP'11*, 2011, pp. 7–9.