Using Asynchronous Client-Side User Monitoring to Enhance User Modeling in Adaptive E-Learning Systems

David Hauger
hauger@fim.uni-linz.ac.at

Institute for Information Processing and Microprocessor Technology,
Johannes Kepler University, Linz, Austria
http://www.fim.uni-linz.ac.at

Abstract. Traditionally, user modeling in adaptive hypermedia systems is based upon monitoring requests of resources on the server. This information, however, is relatively vague and may lead to ambiguous or uncertain conclusions. One approach to improve granularity and accuracy in user models is monitoring "client-side" user interactions in the browser. Despite their steadily growing importance, especially along with the emergence of Web 2.0 paradigms, up to now such interactions are hardly being monitored. This paper shows how, by means of Web 2.0 technologies, additional information on user interactions can be retrieved, and discusses the benefits of this approach for user models (like accuracy, granularity and immediate availability of data out of the request-response cycle) as well as for subsequent adaptations.

Key words: user monitoring, interaction, Web 2.0, user modeling, asynchronous adaptive hypermedia

1 Introduction

Traditional user modeling techniques of adaptive hypermedia systems (AHS) monitor requests of resources on the server to make assumptions on users. However, the fact that a document has been requested might be an insufficient source of information to determine whether it has been read. After servers have transmitted a hypermedia document, there is usually no further feedback to tell whether it is still open or which of its sections is being read. If a browser's request to the server (which refers to pages as a whole) is the only source of information available, then one cannot make assumptions in relation to the document's comprising parts or sections. Therefore, documents have to be treated as atomic items.

Additional information may be retrieved by calculating the time between requests (e.g. in Knowledge Sea [1]) or by adding semantic meta data (e.g. by mapping pages to concepts as in AHA! [2]). More and more elaborate algorithms analyze the limited available data to increase granularity and accuracy of user models, while still lacking information on user interactions inside the browser.

In contrast to these approaches, the one put forward in this paper aims to improve the monitoring process itself, and suggests a way to asynchronously monitor client-side user interactions, which user models and consequently AHS should benefit from. Although, especially with the emergence of Web 2.0, the importance of asynchronous client-server communication and client-side interactions have been increasing, they are hardly being used in terms of user modeling at the moment.

2 Client-Side User Monitoring

Instead of exclusively monitoring requests on server-side, interactions could as well be monitored directly within the browser.

2.1 Limitations of Traditional Approaches

As long as observation is reduced to server-side monitoring, user interactions that do not cause browser requests can not be observed, which limits possible assumptions on user behavior. Some of the questions that can not be answered due to the lack of information on client-side interactions are [3]:

- Whether a user has spent time on the page
- Whether a user has had a look at the whole page
- Whether a user has read a page
- Whether a user is interested in the page
- Whether a user has understood a page
- Why a user has skipped a page

Moreover, although Brusilovsky [4] stated that “the user can prefer some nodes and links over others and some parts of a page over others” already in 1996, current approaches are still not able to tell these differences for text nodes, as reading or copying text does not result in new requests. Retrieving more fine-grained information on which parts of a page have been read would consequently lead to better results than regarding a whole page as an atomic unit and mapping it to a concept.

2.2 Anticipated Benefits through Client-Side User Monitoring

As client-side monitoring provides more information on the users’ interactions, it allows additional assumptions and increases the level of certainty for conclusions. This might help to improve both models and resulting adaptations.

Improving Accuracy of Existing User Models One evident example where user models can be improved is the assumption of a user “having read a page”. Traditional systems make use of the information that a document has been requested. Some more advanced systems consider the time of a subsequent request to identify the “time spent reading” [1]. However, a user might have opened several documents in different browser windows or tabs at the same time. Switching between them leads to wrong values for the "time spent reading" as the user is always assumed to be reading the most recently requested page. Moreover, there is no information on whether the user has seen the whole page (e.g. scrolled down if the document is larger than the browser window). In most cases the document being read will have the focus, which can be checked by client-side monitoring. Being displayed within the active browser window is a stronger indicator for determining which page is being read, than supposing that the active document is always the most recently requested one. Moreover, it can be assumed that a document is being read if the user is interacting with it (periods of inactivity can be identified). Monitoring scroll events allows for checking whether all parts of a hypermedia document have been visible within the browser window and can be used to calculate the time spent on each segment of the page.

Improving Granularity of Existing User Models Hijikata [5] showed that text tracing, link pointing, link clicking and text selection may be used to identify interest. Of all these interaction types only link clicking causes a server-side event. The others are not being monitored by traditional systems. Therefore the systems usually take the amount of requests (sometimes with respect to the assumed time spent reading) to determine interests. Using client-side interaction-based user monitoring, all types of interactions can be used to set up a more detailed user model. Moreover, it is possible to define more granular user model attributes; e.g. not only read/unread, but also the percentage of a document or concept that has been read.

Independence of the Request-Response Cycle If the transmission of data depends on subsequent requests, information is lost, if users leave a page. When using browser history or parallel windows for navigation, incorrect data is transmitted. Asynchronous monitoring is able to overcome these restrictions by retrieving data out of the request response cycle, and additionally results in a model always being up-to-date. Moreover, if changes in the user model require immediate adaptations, this can be achieved as well. As the communication takes place asynchronously, the system might offer help, highlight important elements or provide any other (non-distracting) adaptation while the user is still interacting with the page [6].

3 State of the Art

Attempts concerning client-side monitoring have already been made by Goecks and Shavlik [7] who used JavaScript to log mouse and scrolling activities and summed it up into a "level of activity". Mapping this value to a set of keywords extracted from the visited page has been used to determine the "interests of the user". Hofmann et al. [8] extended this approach by sending information on all events to the server (especially timestamps).

Due to technical boundaries (workarounds like refreshing iframes, Java applets or flash movies have hardly been used) both monitoring and adaptations were traditionally bound to a strict request-response cycle. In order to overcome these limitations, custom browsers have been developed (like “AVANTI” [9], which monitored the time spent reading and client-side interactions in order to set up a user model and determine disabilities); or browser plugins (like the agent-based approach in [7] monitoring the time spent on a page, mouse events, scrolling and status bar changes and using this information as input for a neural network). While effective, the necessity to install additional software limited the applicability of the approach.

Within “The Curious Browser” [10], the browser itself was monitoring and storing user interactions and compared this information to explicit ratings of users. Analyzing the results, Claypool et al. discovered that the amount of scrolling, the time spent on a page and combination of these two pieces of information is strongly correlated with the explicit ratings. Individual scrolling methods and mouse clicks were ineffective in terms of determining strong explicit interests, but helped (when missing) to identify the least interesting content. These results prove that client-side interactions may be used to get valid information on users’ interests.

With the emergence of Web 2.0 and asynchronous web technologies like AJaX (running in the background, not requiring additional plugins) some limitations for continuous and unobtrusive monitoring of user interactions are now addressed. Using these technologies not only for presentation, but also for user monitoring has already been suggested for help systems by Putzinger [6]. He used mouse and keyboard events to determine the “locus of attention”, especially focusing on input elements. The gained information has been used to adaptively provide help for the user.

As Web 2.0 technologies extend the range of what is technically possible, it is now up to researchers to answer questions that could not be addressed earlier. This includes increasing granularity and accuracy of user models [3]. The current challenge lies in discovering how client-side interactions can be monitored and processed to retrieve more semantic information about users and their behavior.

4 Asynchronous Client-Side User Monitoring

In order to monitor user interactions inside the browser without requiring additional software, client-side scripting languages are required. Although there are several possibilities, for the system currently being developed JavaScript has been selected as the most appropriate choice, as it is widely supported and used. AJaX is used to send the collected data to the server as it is unobtrusive, not bound to the request-response cycle and natively supported by modern browsers.

4.1 Monitoring Client-Side Events

JavaScript already provides a set of events that can easily be monitored. The current system uses various mouse and keyboard events, scrolling, window resizing, window blur, window focus and document ready.

However, some of these events occur too frequently for continuous monitoring, which is especially true for mouse moves and scrolling events. They are an indicator for interest and need to be monitored. For a single mouse move the browser may generate a large amount of JavaScript events, which would cause a lot of network traffic if every event would be sent to the server. Moreover, the area of the mouse move is of interest and not the exact route and all mouse positions. Therefore, these events need to be preprocessed and aggregated (e.g. all mouse-move or scrolling events within a small period of time are combined to a single one). In order to achieve this, the library developed along with the approach put forward in this paper supports the definition of custom events. They can also be used for decreasing the level of detail (e.g. if exact positions are not required), adding supplemental information (e.g. based on the event context) and for creating specialized monitors. Additionally, custom events are necessary, because not everything that is monitored on the client side directly causes a JavaScript event. In the case of selecting text (which needs to be monitored e.g. to identify text tracing), different events and document states have to be monitored to be able to tell, when a “text selected” event occurred. Combining this event with additional mouse or keyboard events can show if a piece of text has been copied for further processing. Furthermore, custom events may use their own triggers, e.g. timers for events of a temporal basis.

The data provided by JavaScript events may be both too detailed and/or insufficient. In order to determine parts of a page a user has been interacting with, exact mouse positions hardly serve, if the content is not bound to fixed positions or dimensions. Different font- or window sizes may change the absolute positions of content. In order to overcome these limitations the concept of page fragments is being introduced.

4.2 Monitoring Page Fragments

In order to be able to not only use a general “level of activity” within a page, but to treat parts of a page differently, events need to be mapped to semantic meta information, and/or their spatial location needs to be identified.

One approach is to analyze the text a user is currently interacting with. For example the event triggered on text selection contains information on the currently selected text, which may be used for keyword analysis. For other types of events, like mouse-over events, analyzing keywords may be less effective. These events are better suited to indicate activity within an area, than to assume the user is particularly interested in the words being hovered above. Furthermore, mapping events to keywords does not differentiate between parts of a page using the same keywords and makes it therefore difficult to tell what percentage of a page a user has read. Despite being useful for some types of events, exclusive use of this approach is not able to address some of the difficulties mentioned in previous sections.

Monitoring all HTML DOM elements and mapping events to them brings about problems as well. If a hypermedia document is highly structured in terms of markup (an element may contain a single letter), the result might be too detailed and consequently not useful. For some interaction types the exact position is only an approximation for the locus of attention. A user does not need to interact with every single element while reading it. Therefore, monitored areas must not be too small or detailed. On the other hand, within unstructured documents a single HTML element may contain the whole content, which results in treating the page as a single unit again.

Splitting hypermedia documents manually and providing semantic meta data for each section might be effective, but requires additional authoring effort and might therefore cause acceptance problems. The solution to be chosen should offer the possibility of manual sectioning, but provide its advantages also for unstructured contents.

The suggested solution tries to meet these demands by introducing "page fragments". By default, a page is vertically split into a number of k sections, each representing $1/k$ of the height of the document. Changing the width of a browser window may change the height of the document, but the height of the fragments will change as well. For $k = 5$ the third fragment will represent the center of the page; everything between 40% and 60% of the total height of the page. As this percentage will not change through resizing or using different font sizes (although the absolute postitions might change), this approach might be better suited for locating events. For some items, like images or spatial layout elements, resizing influences exact positioning more strongly; but nevertheless, the approximation should be sufficient.

One benefit of this approach is the independence of the structure of the content. Even unstructured documents may be segmented this way. For structured contents Therefore, no additional adaptive authoring effort is required.

The actual size of a fragment depends on the height of the document and on the value chosen for k. The higher the value for k, the higher the level of detail. However, if k is too high and/or there is little content on a page the height of one fragment might be less than the line height, which means that clicks on the top of a word are treated differently from clicks on the bottom of a word. Determination of optimal values of k for different circumstances is part of the ongoing work. As each fragment represents a fixed percentage of the page and each fragment is monitored separately, it becomes easier to tell how much of a page has been read; based on how many fragments are regarded as "having been read". Although the claim something has been read still remains an assumption, having gone through the whole page, spending a reasonable amount of time on all of its parts and interacting with them should be a stronger indicator for reading than simply requesting a document and possibly spending time on it.

Although using fragments instead of absolute positions to retrieve spatial information results in a loss of precision, this approximation should be sufficient for most applications, as e.g. mouse positions themselves are only an approximation for the locus of attention.

Within relatively static documents a fragment represents not only some percentage of the page, but also the actual content, which is important to determine interests. However, if parts of a page are adaptively included, the text within a fragment might shift, which requires a different technique to identify the actual content a user has been interacting with. Moreover, if semantic information is already available, it would be interesting to use it. Therefore, in addition to the predefined fragments, "custom fragments" can be defined, i.e. any HTML element can be specified as an additional fragment. Such custom fragments may be images (if they should be treated differently), sections automatically detected by means of topographical information (e.g. using headlines as separators) or elements that are already mapped to concepts or semantic meta information. When assembling hypermedia documents from multiple sources (e.g. due to conditional fragment inclusion), fragments can, during assembly, automatically be denoted and semantically characterized on the basis of their source, their "function" in the assembly, etc..

Generally, custom fragments are treated like predefined ones, so events are mapped to them as well (they can be mapped to more than one fragment). For instance, the time spent on images can be automatically calculated without requiring further modifications. Moreover, events can be defined for single fragments only, e.g. to specify the required reading time for each image separately (based on the actual content) and to trigger an event, if the time is exceeded and the image is regarded as visited.

5 Proposed Adaptations and Applications

Having monitored the user's interactions, the next step is to use this information within an AHS.

5.1 Using Existing Adaptation Techniques

The acquired data can be used directly within existing AHS, for instance, by replacing values in the user model. As an example, the values for "having read a page" or "knowing a concept" may be redefined within the user model by something more accurate, like "having spent at least x seconds on each page fragment". In this case, existing adaptive content might be used as is, while still profiting from more adequate data. Secondly, the level of certainty for attributes within existing user models may be increased. A third example on how to make use of the available information (using existing adaptation engines) is the definition of new and more granular adaptation rules like "probability of having read at least half of the page is greater than 80%". This helps authors of adaptive content to use the same authoring process as before, while still benefiting from user modeling based on client-side interaction monitoring.

5.2 Social Navigation Support

"Social navigation" has been defined as "moving 'towards' a cluster of other people, or selecting objects because others have been examining them"[11]. In

Knowledge Sea II [1] page visits and the time spent reading have been used to determine the most interesting documents.

Using the activity information on single page fragments, social navigation support can be provided not only on the level of documents and links, but also to highlight sections of a document. Fragments with a high number of interactions and reading time might be the most attractive and relevant ones. As users tend to follow highlighted links [12], highlighting these "popular" fragments might help readers to get a quick overview on the most interesting parts of a content.

Moreover, social navigation support may help authors to improve their contents by noticing, which parts of a document users found interesting. If within a block of important content users tended to spend time on the first lines only and skip the rest, the content might be too difficult, already known or not interesting enough. Important sections that tended to be skipped might need to be improved.

5.3 Identifying Learning Styles

One example where information on custom fragments opens new perspectives is the identification of learning styles. If images are treated as special fragments the system may identify the time spent on pictures and the interactions performed on them. These values may be compared to the average number of interactions / time spent on other parts of the page for the current user, which should allow for conclusions on the user's preferred learning style.

Having included this information into the user model, course material may be adapted accordingly. Contrary to traditional systems asking for learning style preferences, this approach might help to identify the learning style automatically by monitoring user interactions.

5.4 Retrieving Additional Information by Analyzing Usage Patterns

Perkowitz and Etzioni [13] have addressed the identification of usage patterns based on access logs. This approach can be extended by additional interaction data. If users stay on a page for a short time this might be because the content is too difficult, irrelevant or already known. Till now it has not been possible to tell the differences, because from a request-based point of view the users act in the same way. However, they might differ in their way of interacting with the system. A user not interested in the content might stay at the top of the page, read the headline and move on. Novice learners may start reading and scroll back again and give up before reaching the end of the page. Experts might have a quick glance at the whole page reading just the headlines and move on to an advanced topic. These are just some ideas on what might be found when analyzing interaction data. As there is more data available, new challenges arise concerning the analysis of this data. Finding patterns might be an important further step in improving AHS user models.

6 Ongoing Work and Future Perspective

A JavaScript library for asynchronous client-side user monitoring as described in section 4 has been developed. The implementation aims to be both easy to use and simple to extend with respect to client-side observation as well as server-side data processing. Existing courses can easily be provided with the core functionality of the library, while still allowing experts to add advanced features. Ongoing work on prototypes aims to provide and improve sample implementations according to the proposals in section 5. In order to provide the results of this work for a larger audience, the system is currently being integrated into a version of AHA! [14] embedded in the Sakai e-learning platform [15].

Evaluation work planned for the immediate future aims to determine an appropriate value k for the number of fragments a page will be divided into. It is presumed that in order to have comparable data, k should be constant within a system. However, for different types of systems a different k might fit best, depending on the actual content and its length. For example, in an AHS presenting slides (generally fitting in one screen), a lower k might be sufficient, in contrast to one providing e-books, where a page may contain a whole section or chapter. Two approaches will be used to determine the optimal k within specific contexts. The first one will start with a high amount of fragments. Values for neighboring fragments will be compared in order to find out, how many significantly different sections this number could be reduced to. A second approach would allow users to manually adjust the value for k, so that it subjectively results in the most appropriate user model.

Another challenge is to determine suitable weights for interaction types when combining different types of interactions to a "level of activity" for specific page fragments. A model capable of theoretically characterizing but also quantifying different types of activities will need to be selected and applied. The quantified "level of activity" resulting from said model will have to be verified with real end users' subjective perceptions of activity.

At a more general level, to evaluate the overall approach put forward in this paper, one would have to ascertain that: (a) it results in a more fine-grained and significantly more accurate user model and, (b) that this higher level of detail and the improved accuracy result in better adaptations.

For the tests, a version of AHA! based on traditional modeling techniques will be compared with an extended version analyzing client-side user interactions. To check the accuracy, a summative evaluation will be done through user studies. Students will work on their AHA! courses and will have to evaluate the two user models and decide which one they find more appropriate. Care will be taken to ensure that users do not just select the model presenting the values most favorable to their person (e.g. showing higher values for knowledge for a large number of concepts). Alternatively, only one model could be used, presented and evaluated and the average evaluation of both models would then be compared. Another scenario would be a self-evaluation of students and a comparison of these results to both models.

Acknowledgments

The work reported in this paper is funded by the “Adaptive Support for Collaborative E-Learning” (ASCOLLA) project, supported by the Austrian Science Fund (FWF; project number P20260-N15).

References

1. Farzan, R., Brusilovsky, P.: Social Navigation Support in E-Learning: What are the Real Footprints? In: Third Workshop on Intelligent Techniques for Web Personalization (ITWP ’05). At 19th Int. Joint Conf. on Artificial Intelligence. (2005)
2. De Bra, P., Calvi, L.: AHA! An open Adaptive Hypermedia Architecture. The New Review of Hypermedia and Multimedia **4** (1998)
3. Hauger, D.: Fine-grained user models by means of asynchronous web technologies. In Hartmann, M., Krause, D., Nauerz, A., eds.: ABIS 2008 - Adaptivity and User Modeling in Interactive Systems, Würzburg, Germany (2008) 17–19
4. Brusilovsky, P.: Methods and techniques of adaptive hypermedia. User Modeling and User-Adapted Interaction **6**(2-3) (1996) 87–129
5. Hijikata, Y.: Implicit user profiling for on demand relevance feedback. In: IUI ’04: Proc. of the 9th int. conf. on Intelligent user interfaces. (2004) 198–205
6. Putzinger, A.: Towards asynchronous adaptive hypermedia: An unobtrusive generic help system. In Hinneburg, A., ed.: LWA 2007: Lernen - Wissen - Adaption, Halle, Germany (September 2007) 383–388
7. Goecks, J., Shavlik, J.W.: Learning Users’ Interests by Unobtrusively Observing Their Normal Behavior. In: Int. Conf. on Intelligent User Interfaces - Proceedings of the 5th int. conf. on Intelligent user interfaces. (2000) 129–132
8. Hofmann, K., Reed, C., Holz, H.: Unobtrusive Data Collection for Web-Based Social Navigation. In: Workshop on the Social Navigation and Community based Adaptation Technologies. (2006)
9. Stephanidis, C., Paramythis, A., Karagiannidis, C., Savidis, A.: Supporting interface adaptation: The avanti web-browser. In Stephanidis, C., Carbonell, N., eds.: Proceedings of the 3rd ERCIM Workshop “User Interfaces for All”. (1997)
10. Claypool, M., Le, P., Wased, M., Brown, D.: Implicit interest indicators. In: Intelligent User Interfaces. (2001) 33–40
11. Dourish, P., Chalmers, M.: Running out of space: Models of information navigation. Proceedings of HCI’94 (1994)
12. Farzan, R., Brusilovsky, P.: AnnotatEd: A social navigation and annotation service for web-based educational resources. New Review in Hypermedia and Multimedia **14**(1) (January 2008) 3–32
13. Perkowitz, M., Etzioni, O.: Adaptive web sites: Concept and case study. Artificial Intelligence **118**(1) (2001)
14. De Bra, P., Ruiter, J.P.: AHA! Adaptive Hypermedia for All. In: Proceedings of the WebNet Conference. (2001) 262–268
15. Sakai: collaboration and learning for educators, by educators, free and open source. http://sakaiproject.org (2009)