INTERPRETATION OF WEB SITE USER INTERACTION AS A BASE FOR CONTEXT-AWARE PAGE ADAPTATION

Pascal Bihler
University of Bonn – Institute for Computer Science III - Römerstr. 164, 53117 Bonn

ABSTRACT

The Web contains immense knowledge, presented mostly in form of HTML pages. Making this information available to mobile users in a context- and device specific way is one of the main challenges for mobile web usability. Context-aware, automatic web page adaptation builds upon semantic and structural knowledge, which can go beyond the DOM structure. The *Semantic Shadow* concept aims to provide such information about the inherent structural semantic of a web page, as well as hints about the contentual semantic, in an explicit way. This data, which is managed in a context-aware annotation layer, can be derived from observed user interaction, as the paper shows. Together with the general approach, basic methods for deriving fundamental annotations are presented. An exemplary showcase for using the context-aware annotations for adapting a survey web page is shown, based on the interaction data collected during a user study.

KEYWORDS

Semantic Shadow, Context-Aware Annotations, Usage Tracking, Mobile Web Adaptation.

1. INTRODUCTION

Over the last years, web usage has changed by a lot: In the beginning, the web's user group was compact, but today, the user groups of a certain web based service are very divers. The trend for context aware usage has been amplified by the advent of small, high-performance Internet devices like smartphones and netbooks. Usage context therefore defines another dimension in the web service design space.

Since the user and device variety is increasing more and more, manual adaptation of web pages for every usage scenario seems not to be a viable approach in the long run. Nevertheless, it is highly desirable that a service's user interface would be adapted for the specific user needs and device capabilities. As we argued in [3], an annotation of web page elements with semantic information revealing the structural and contextual content of the elements to processing algorithms, would support automatic page adaptation and other context-aware operations. The *Semantic Shadow (SemS)* concept is a model for such annotations.

The manual generation of such annotations, even if supported by a GUI based tool, is very troublesome and thus unrealistic, especially for existing pages and web services. This paper proposes another way of generating semantic annotations by deriving them from web page usage observation. The approach is motivated by the idea that the user reveals hidden structural information by the way he or she interacts with a web page.

In the next section, the relevant parts of the *Semantic Shadow* annotation model are recapitulated. The main section discusses the options for observing web page user interaction and elaborates the generation of several annotations from the base set. Based on selected results from a user study, the applicability of the derivation process is demonstrated and applied to the use case of mobile web page adaptation. A comparison with related work is followed by a summary and research outlook.

2. THE SEMANTIC SHADOW MODEL

The *Semantic Shadow (SemS)* [3] concept introduces context-aware annotations on HTML elements. This allows expressing additional knowledge about web pages, which cannot be encoded into the HTML tags themselves. We distinguish two different kinds of annotations:

- Annotations describing the *content* of the annotated element (i.e. some role, meaning etc.).
- Annotations describing the *structural semantics* of the annotated element (grouping, priority, etc.).

This paper concentrates on the generation and usage of *structural annotations*.

2.1 Structural Annotations

For the *Semantic Shadow* concept, a base set for such structural annotations has been defined:

isMemberOf(x,G): The element or group *x* is member of a semantic group[1] named *G*.

hasPriority(x,P[,G]): The element *x* has a relative priority *P* (*P* being a rational number), compared to other elements of the page or a specified group *G*.

receivesKeypresses(x): The element *x* can receive key presses.

supportsCharset(x,C): The element *x* receives characters from the given charset *C*.

hasValueLength(x,N): The value of element *x* has a length of *N* characters.

hasAttentionTime(x,T): The users attention is focused for *T* seconds on the element *x*.

followsFocus(x,y): The focus of the element *x* follows the focus of *y*.

hasFocusFollower(x,y): The focus of the element *x* is followed by the focus of *y*.

isSummary(x,Y): *x* is the summary of *Y*, whereas *Y* can be a page element or a group.

inducedBy(v_x,v_y): The element *x* has the value v_x, if *y* has the value v_y.

induces(v_x,v_y): If the element *x* has the value v_x, the element *y* has the value v_y.

dependsOn(x,v_y): If the element *y* has the non-default value $v_{y<}$, *x* can get a non-default value.

hasDependent(v_x,y): If *x* has the non-default value v_x, *y* also can get a non-default value.

While these annotation types certainly do not cover every possible structural relation, the open design of *SemS* makes it easy to extend this base set in future.

2.2 Annotation Model

A semantic annotation *(Shadow Annotation)* a is defined in *SemS* as the tuple $a = (s,t,p_1 ... p_k,\Gamma)$. *s* denominates the *Subject* of an annotation, i.e. the HTML element it relates to. *t* is the *Type* of the annotation, $p_1 ... p_k$ are *additional properties*, as required by the semantics connected with the type of the annotation. These additional properties can be scalar values, but they can also reference other web site elements or their values. Γ is the so called *Contextual Confidence* of the annotation, generally defined as $\Gamma = (\gamma,c)$, whereas *c* refers to a context object[2] and $\gamma \in [0..1]$ denotes the probability that the given annotation's statement is valid in the given context. A special case for Γ is $\varnothing = (1.0,\varnothing)$, saying that *a* is valid in every context.

The set of all known annotations is called the *Semantic Shadow*. This information can be stored and accessed using an RDF representation (see [3]) making it possible to use existing Semantic Web technologies for data processing.

3. FROM WEB USAGE TO CONTEXT-AWARE ANNOTATIONS

To extend web sites with context-dependent semantic annotations as described in the previous section, several methods can be applied: The most basic way is to construct the semantic annotations together with the rest of the web site while creating the page. A second possibility is to use dedicated annotation tools. A third way, proposed in this paper, is the derivation of contextual semantics by unobtrusively studying frequent user interactions, as given by the regular use of the web pages. This method requires several steps: Firstly, the usage of the web resource in question has to be tracked. Secondly, this tracking data has to be summarized for privacy and manageability issues. Thirdly, meaningful semantic annotations have to be extracted from this aggregated usage data and linked back to the related web site element.

[1] A *semantic group* is a named set of elements on the page.
[2] As the requirements for a "context" definition in the *Semantic Shadow* concept is reduced to the inclusion operation and the URI-based referencing to a compatible representation, any model supporting those requirements can be used.

3.1 User Tracking Methods

Several methods have been applied to observe the user while he or she is surfing the web and inspecting its pages. We separate these methods into four categories, each improving the quality (but also the amount) of tracked user data, but at the same time requiring more complex software and client-server interaction.

3.1.1 Inspecting HTTP Request Logs (L1)

In order to display a web page in the user's browser using the the HTTP protocol a GET or POST request is send to the server hosting the web page in question. Besides the requested web resource, the request contains meta information such as the browser's identification string ("User Agent") and the web site which was viewed before the browser issued the request ("Referrer"). Traditionally, web servers maintain log files were they add a line to for every handled request ("Access Log"). In general, the logged dataset contains information about the user's IP address, the request time, the requested resource, the user identifier (if the resource was password protected), the user agent and the request's referrer.

3.1.2 Inspecting HTTP Request Parameters (L2)

If a user fills out a form on a web page, the corresponding data is transmitted together with the next HTTP request. Unfortunately, not the input's field DOM position or ID is used as value identifier, but a dedicated *name* attribute of the corresponding form element. Using the page HTML data, the originating input element can be identified with the *name* attribute, but unfortunately, this association is not guaranteed to be unique. In addition, only the final value of the user input can be inspected using this method.

3.1.3 Inspecting Client Events (L3)

The first two methods can be implemented completely on the server side of a web page access. If more detailed information about the web site usage is desired, events like mouse movements, key presses and focus shifts have to be tracked on client side. An unobtrusive way to record events triggered by the client browser is the usage of embedded *JavaScript*, like done by [2, 14] Every time the user triggers a relevant event, an embedded script communicates with a proxy to log this event on the server.

3.1.4 Using Dedicated Hardware (L4)

The most sophisticated way of tracking the user's attention when scanning a web page is to follow his or her eyes while they slide over the browser's page visualization. Using eye trackers as computer input has long been under research [6, Chapter 20], often for the evaluation of web site interaction [8]. In experimental settings, an eye tracking hardware can be used to determine the user's points of fixation and scanpaths on the web page [15]. Concluding from the point of focus to the focus of interest is valid [9], but is not generalizable [16]. In addition, to create a critical data mass the mining of usage data for web site annotations needs to go beyond experiments, which requires an eye tracking being part of the standard web usage process.

3.1.5 Comparison of Tracking Methods

The advantage of eye tracking usage to determine user focus time on web elements is its higher precision compared to an approximation by interpreting the mouse cursor as point of interest (cf. the discussion in [2, Sec. 6.2]) On the other hand, measurements might still be imprecise, since eye tracking algorithms are very sensitive to external disorders like caused by the instability of the eyes, blinking and other external factors [6]. Additionally, eye tracking is a procedure the average user is not used to, so especially in the case where video data is evaluated, privacy concerns might make the user feel uncomfortable with the usage tracking method.

For the structural annotations, which are generated using the procedures and concepts presented in this paper, a detailed view on the user's interaction with the web page is required. On the other hand, in order to generate resilient context-dependent annotations, interactions from as many subjects as possible have to be aggregated, so the inhibition threshold of participating in the project should be low. This excludes the use of dedicated hardware and leaves the options of analyzing server-based and *JavaScript* generated tracking data. As usage data collection may raise privacy concerns, the user has to agree to the data collection.

3.2 Deriving Context-Aware Annotations

This section shows how the structural annotations from the base set can be derived from usage data, tracked as described before[3]. The information about the usage of web pages is normally stored in sequential lists of events, traditionally referred to as *logfiles*. In the following we assume that for every user u viewing a page p in a context c, a dedicated event list $E_v = (e_{v0}, e_{v1}, ..., e_{v(n(v-1))})$ for this view $v=V(u,p,c)$ is available. A page view v is defined as the process requesting the web page and interacting with its visualization, until requesting the next web page or canceling the web browsing session. The total number of events tracked during the page view v is referenced as $n_v = |E_v|$. Every event e_v consists of a timestamp t_{ev}, an event name n_{ev}, the event scope s_{ev} (i.e. the web page element x the event relates to) and additional, event specific parameters π_{ev}. The events of an event list E_v are ordered chronologically. All elements of a page p form the set A_p. In addition, the following set of children of an element $x \in A_p$ is defined: $O(x) = \{y \in A_p \mid (y = x) \vee (x \leftarrow y) \backslash\} \subseteq A_p$ where $(x \leftarrow y)$ symbolizes that the element x is the direct or transitive parent of y.

In the following, the tracked web page usage information is referenced as L1, L2, L3 and L4, depending on the different ways of data collection. If not stated otherwise, the event list E_v of a single view v is treated as input, and reasonable aggregations over all tracked page views (average, weighted average, global extreme values etc.) have to be made to calculate the overall valid predicates for all $x \in A_p$.

hasPriority(x,P): When deriving the priority from usage data, a simple approach is to assume that for active elements (e.g. buttons) the importance of an element is directly correlated with the activation frequency, which can be calculated observing the L3 *onfocus* or *onclick* events[4] For passive elements (e.g. paragraphs), the importance of an element can be seen as correlated with the time the user spends his attention on it and calculated by normalizing this time (as expressed by *hasAttentionTime(x,T)*) by the amount of information the element carries (e.g. number of characters).

receivesKeypresses(x): With L3 tracking information, this predicate can be set for every x, where an event e exists with $s_e = x$ and $n_e = onKeyPress$. If only L2 tracking information is available, the predicate can be deduced from the presence of an event with $s_e = x$, $n_e = value$, $\pi \neq \varnothing$. Unfortunately, this method can just be applied for text input fields in submitted forms and is only a sufficient criterion.

supportsCharset(x,C): This predicate can be determined using the same techniques as described for *receivesKeypresses(x),* extended with the concrete analysis of the input value. As soon as a key from a certain charset is detected, this charset can be flagged as "supported". When using L3 tracking information (events *onkeypress* and *onvaluechanged*), attention has to be paid to analyze only the final result of an input event set, because most probably the final input reflects the intended element input semantics better than intermediate typing errors.

hasValueLength(x,N): To get the length of an input value, input can be tracked as in *supportsCharset(x,C)*, and the length of the input can be evaluated. In case of E_v being L2 tracking information, the maximal submitted value length for the element x is taken. For L3 tracking information, the same measures as in *supportsCharset(x,C)* are required.

hasAttentionTime(x,T): The attention time of a certain element x can be tracked using L4 log data by analyzing the visual focus of the user. The attention time is then assumed as the duration from focusing the element x until focusing another element or loading another page. If only L3 tracking information is available, the same algorithm can be applied by replacing the event *attentionFocus* by the *onmousemove*. This seems to be a good approximation [2, Sec. 6.2].

hasFocusFollower(y,x) / followsFocus(x,y): In order to derive these predicates from usage data, L3 tracking information is required. The annotations can be extracted by observing the *onfocus* event.

induces(vy,vx) / inducedBy(v_x,v_y): These predicates can only be reasonably derived if a multitude of page views is analyzed. From the L2 tracking information (and from L3, when event interpretations like for *supportsCharset(x,C)* are taken into account), all submitted value tuples can be derived. To reduce the complexity, only the subset of inductions based on a direct focus switch are considered, filtering the tuples to those element pairs *(x,y)*, for which *followsFocus(x,y)* has been declared.

[3] Except for isMemberOf(x, G) and isSummary(x,Y), which can better be generated using static HTML document analysis, see the section on related work for examples.
[4] If only L2 tracking information is available, the *onclick* event can be "simulated" by analyzing the submitted button values.

hasDependent(v_y,x) / dependsOn(x,v_y): Using the correspondent *inducedBy(v_x,v_y)* predicate set, all v_y for which the *dependsOn(x,v_y)* statement applies can be calculated as $\bigcup_{v_x}\{v_x \in inducedBy(v_x, v_y) | v_y \neq DefaultValue(y)\}$. In the same way, *hasDependent(v_y,x)* can be derived from the *induces(v_y,v_x)* set.

To calculate the Contextual Confidence value, a reference value (e.g. the page view count in the page view's context) is stored together with the number of times a concrete annotation is valid in the request context. The Contextual Confidence Γ of an annotation a in the context c is then expressed by $\gamma = \frac{\sum_{v \in V_e} t(a,v)}{|V_c|}$ where V_c denotes all page views in a context *c* and *t(a,v)* is 1 iff a is valid for the page view *v*, 0 otherwise.

3.2.1 Context Information

Confidence level calculations for annotations in the *Semantic Shadow* concept are context related. Therefore, when generating annotations based on tracked usage information, the context of the corresponding page view has to be constructed and persisted as well. If no extension of the user's browser in the form of plug-ins is arranged, only the HTTP transfer data can be examined to build up the page view context, i.e. the connection information (particularly the user's IP address) and the exchanged HTTP headers.

Using a Web Service as the one provided by IPInfoDB (http://ipinfodb.com) the IP address can be used to guess the user's location when issuing the page request.[5] For more precise data, the W3C Geolocation API can be used to query the request location from the user using embedded *JavaScript* while collecting tracking information. With a reverse geocoding Web Service, the context data can be filled with region and country information. Based on the calculated geographic origin of the request, the user's time zone can be calculated and time stamp information can be shifted to match the user's time experience during the page view.

Another important regional aspect of the request context is the user's language. Browsers submit *Accept-ed-languages* in the request header, containing an ordered set of language identifiers the user has chosen. To get information about the user's browsing device, the submitted browser identification string (*User-Agent*) can be matched against a database of known devices.

4. EVALUATION AND APPLICATION

To evaluate the derivation of semantic annotations from tracked usage data, web page usage data has been collected during a one-week study and the structural annotations generated based on this usage data have been compared to expert knowledge. During this study, the participants were invited to take part in a web survey about a subject different from this research's one. Apart from the survey's explicit intension, the implicit interaction with the survey was recorded using the L2 and L3 tracking techniques. The structural annotations derived from the collected usage data were evaluated and visualized on the rendering of the corresponding web page to allow a correspondence verification with the semantic structure of the page.

4.1 Deriving Annotations

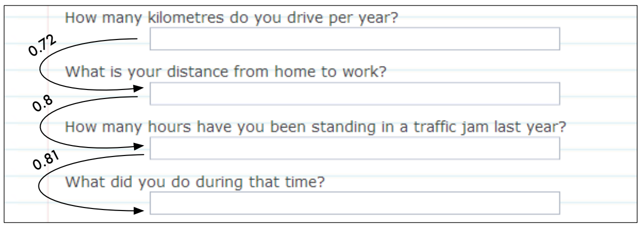
To demonstrate the effectiveness of the annotation derivation, participants of a user study had to fill out a short web-based survey. The survey consisted of four pages: On the introductory page, the user was introduced into the survey's topic and informed about the usage data tracking. In a form, the user entered "statistical data", i.e. his age, and gender[6], confirmed that he agreed with the data collection for scientific purposes and entered the survey form by clicking a button.

The other pages contained the actual survey, carried out as a classical HTML page, where every page element had a unique id. On the last page, the user had the option to leave his email address for participation in a small gift drawing. The usage of the pages was continuously tracked and logged on the server. After submitting every page to the server, the survey data is stored in a database to allow statistical evaluation. As a result of the last form transmission, the user received a "Thank you" page.

[5] This data is usually quite imprecise, since IP addresses from a locatable address block might be distributed over a wide area. Also Internet network technologies like virtual private networks (VPN) or network address translation (NAT) can lead to wrong locations.
[6] Data representing context information.

To evaluate the annotation derivation strategies, the survey included specific structures:

- Some questions only had to be filled out if on a precedent question a specific value was selected.
- Some input values induced other input values intentionally.
- Some fields required specific input value formats (e.g. email address).


How many kilometres do you drive per year?

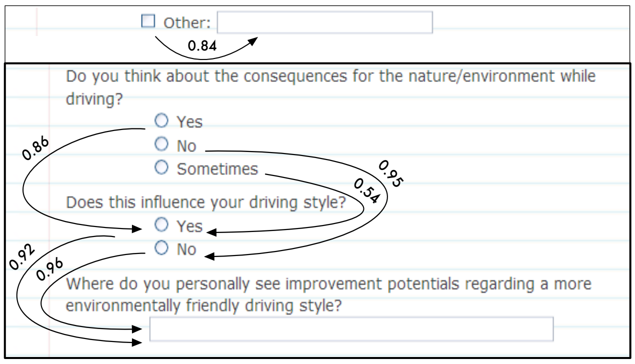
What is your distance from home to work?

How many hours have you been standing in a traffic jam last year?

What did you do during that time?

Figure 1. The natural attention sequence afforded by a series of text input questions is well visible when plotting the contextual confidence values (for c = ∅) of the corresponding *hasFocusFollower* annotations

The user tracking phase was conducted for seven days. During this time, 999 page visits from 353 individuals have been captured and 414,353 user events have been recorded using UsaProxy [2]. Using the algorithms presented in the last section, 13,178 annotations have been generated, 8,462 for the first page, 4,562 for the second page, and 154 for the third page of the survey.[7] The resulting annotations exceeding a contextual confidence level of 0.3 for an unrestricted context and based on more than ten user interaction observations have been visualized on the corresponding web pages. These visualizations have been compared to the semantic content structure of the page and to the results of the survey. This comparison showed that the derived annotations fitted well into the perceived semantic structure of the web pages (afforded by the visual flow of the survey and the connections between some form elements concerning the content of the questions) and therefore reflected well the inherent structure of the page (see for examples Figure 1 and Figure 2). Overall, deriving *Semantic Shadow* annotations from web page usage data has been shown to be accomplishable. Nevertheless, for production scenarios faster execution environments and an optimization of the algorithms are required.


☐ Other:

Do you think about the consequences for the nature/environment while driving?

○ Yes
○ No
○ Sometimes

Does this influence your driving style?

○ Yes
○ No

Where do you personally see improvement potentials regarding a more environmentally friendly driving style?

Figure 2.Contextual confidence values $\gamma > 0.5$ (c = ∅) of *hasDependent* annotations on selected parts of the first survey page (# of contributing user interactions > 10).

4.2 Adaptation for Context Aware Web Access

Using a simple web proxy, several HTML filters have been developed performing simple adaptations on the web page in order to optimize them for context-aware and especially mobile usage, within them:

Input Field Length Optimization: This adaptation maps the value of the *hasValueLength* annotation with the highest contextual confidence in the evaluation context to the *size* attribute of the corresponding input element. This adapts the visual size of an input field in a context-aware manner to the most probable length, giving the user unobtrusive assistance in filling out the web form.

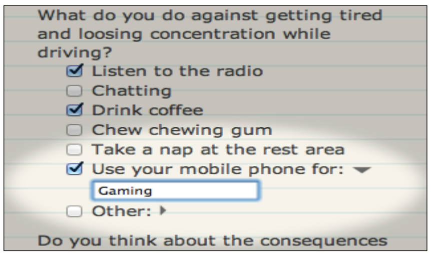
Tab Order Adaptation: The *tabindex* attribute of HTML allows the definition of an order, in which input fields are entered when the user presses the tabulator key. Evaluating the *hasFocusFollower* annotations of the elements on a page, a context-aware tabulator order is derived. In the case of cycles, the cycle element occurring first in the web page DOM is selected as starting point.

Priority Based Filtering: To reduce the information overload on small devices, at first page view elements which are unlikely to be relevant to the user in his current context can be hidden. With *SemS* annotations, this adaptation can be implemented by evaluating the elements' *hasPriority* annotations in combination with their contextual confidences in the request context.

[7] Many participants only filled out the first survey page, which explains the first drop in generated annotations. The last page only included the option to enter an email address to win a small gift, so only a single input field was presented and tracked.

Content Folding: Another approach is to hide details only and to show them upon request, known as *folding*: A small element acts as representative for a longer passage, which is folded out upon explicit user request. The *isSummaryOf* annotation allows to identify such representatives (usually a headline, but also any other element like an image can be assigned this annotation).

Paginate: Using the *isMemberOf* annotation, it is possible to separate a web page in logical subgroups. This adaptation scans the page for such groups (e.g. separated news articles), which are marked using the *hasRole* annotation as *content*. To reduce the amount of elements presented to the mobile user, only one of those groups is shown at a time.



What do you do against getting tired and loosing concentration while driving?

- [✓] Listen to the radio
- [ ] Chatting
- [✓] Drink coffee
- [ ] Chew chewing gum
- [ ] Take a nap at the rest area
- [✓] Use your mobile phone for: Gaming
- [ ] Other:

Do you think about the consequences

Figure 3. A detail of the survey, adapted for the visualization on a mobile device with limited screen size using the *Content Folding* technique.

Auto Fill: Many browsers assist the user by providing selectable options on free text input fields or pre-fill such fields with values used before on this page. Since this auto-fill algorithm only works on a per-user basis, the user cannot take advantage of values entered by other users in a similar context. In addition, most pre-filling algorithms do not evaluate the values already filled in by the user on the same page. Assuming such values have been persisted using the *induces* annotations, the adaptation dynamically extends the user's input to the most probably value in the current context.

Annotation Embedding: To prepare further client-based adaptations, this "adaptation" introduces new attributes on annotated elements in the HTML page. This allows performing the context awareness on the proxy side while leaving the concrete page adaptation to a client-based algorithm.

5. RELATED WORK

Annotation of web pages and its elements with metadata has been described in the last decade by various researchers. Handschuh and Staab present in [10] a system to annotate existing and newly created web pages with metadata. The described CREAM system focuses on using annotations to describe semantically, what is being represented by the HTML elements (named "contentual annotations" in the *SemS* concept), i.e. they do not focus on structural annotations. In addition, they do not model any context dimension for their annotations. Also the SHOE language [11], Ontobroker [7] and SemTag [5] do not model context-awareness.

Hori et al. sketch in [12] a system to optimize web pages for mobile use, driven by RDF-based annotations using XPath and XPointer expressions in RDF subjects. In this work, only the requesting device's capabilities are used as context parameter. To the best of our knowledge, the *Semantic Shadow* concept [3] is the first approach to model context-awareness alongside with semantic annotations on web page elements.

While the idea of generate those annotations based on user interaction has not been implemented before, other approaches to automatically create annotations based on a document text or DOM analysis have been presented: In [5], Dill et al. present *SemTag*, an application to tag automatically large text inputs. They intend to bootstrap the process of generating data for the Semantic Web by providing 434 million semantic tags generated from the analysis of 264 million web pages. In three passes (Spotting, Learning, Tagging) the algorithm seeks for web page sequences to label, resolves ambiguities using a "Taxonomy-Based Disambiguation (TBD)" and stores them into a tag database. While the analysis is fully automated, only the text of the web sites is used as input for the seeking engine.

Going one step further, the approach of Mukherjee et al. [13] also considers spatial locality and consistence in presentation of the elements in question: A "semantic structure" of the document is derived from its presentational definition and its content is then set into correspondence with domain ontologies and lexical databases. This "semantic partition tree" can then be used to enrich the original document with semantic annotations revealing the observed concepts to external systems. The authors refined their original approach to use statistical analysis for automated concept identification.

In their work [4] analyzed the HTML document structure to adapt those web pages automatically for devices with limited visualization capabilities. Similar to the approach of Mukherjee et al., they only focused on the nested grouping of HTML elements as a semantic structure: The page is iteratively segmented into smaller content blocks, until they are small enough and thus suitable for an extracted view on a mobile device. The only "context" considered by the algorithm is the limited device display.

An overview of current content adaptation techniques is provided by Adzic et al. in [1].

6. SUMMARY AND FURTHER WORK

This paper elaborated the idea of deriving context-aware structural semantics on web elements from page usage analysis. By this, the implicit knowledge a user has about a web page and expresses through his or her interaction with the web page in a certain context can be made explicit, without requiring a manual data input by the page editor. The paper distinguished four different levels of gathering web page usage data and compared them with respect to annotation derivation: *request log file inspection* (L1), *request parameter inspection* (L2), *browser event tracking* (L3), and *eye tracking* (L4), while the first two can be executed solely on server side and the other two require support from the client. Usage data from L2 and L3 has been found most valuable to generate the context-aware annotations from the *Semantic Shadow* base set, for which this paper presented basic derivation algorithms. To demonstrate the applicability of the approach, usage data on user interaction with an online survey has been collected. From this data, *Semantic Shadow* annotations have been derived and compared to the survey design by an expert. This comparison revealed, that the generated annotations recreate well the structural patterns expressed implicitly in the survey forms. As an exemplary application, adaptation and optimization of the web pages for mobile use has been show,

In a further work, the current linear approach of user tracking data analysis can be replaced by more elaborated algorithms, e.g. from machine learning. Also, the determination of the request context associated with the generated annotation can be further optimized, e.g. by automatically identifying grouping parameters and by that simplifying the context space traversed during annotation evaluation.

REFERENCES

Adzic, V.; Kalva, H.; Furth, B. (2011), *A survey of multimedia content adaptation for mobile devices*, Multimedia Tools and Applications 51(1), pp. 379-396

Atterer, R.; Wnuk, M. & Schmidt, A. (2006), Knowing the User's Every Move - User Activity Tracking for Website Usability Evaluation and Implicit Interaction, in *Proc. of WWW2006*.

Bihler, P. & Cremers, A. B. (2011), The Semantic Shadow: Structuring the Web for Adaptations, *Electronic Communications of the ECEASST 37*.

Chen, Y.; Ma, W.-Y. & Zhang, H.-J. (2003), Detecting Web Page Structure for Adaptive Viewing on Small Form Factor Devices, in *Proc. of the WWW '03*, ACM, New York, NY, USA, pp. 225-233.

Dill, S.; Eiron, N.; Gibson, D.; Gruhl, D.; Guha, R.; Jhingran, A.; Kanungo, T.; Rajagopalan, S.; Tomkins, A.; Tomlin, J. A. & Zien, J. Y. (2003), SemTag and Seeker: Bootstrapping the Semantic Web via Automated Semantic Annotation, in *Proc. of the WWW '03*, ACM, New York, NY, USA, pp. 178-186.

Duchowski, A.T. (2007), *Eye Tracking Methodology: Theory and Practice*, Springer-Verlag, Secaucus, NJ, USA.

Fensel, D.; Decker, S.; Erdmann, M. & Studer, R. (1998), Ontobroker: Or How to Enable Intelligent Access to the WWW, in *Proc. of the KAW '98 workshop*, AAAI Press, .

Granka, L. A.; Joachims, T. & Gay, G. (2004), Eye-Tracking Analysis of User Behavior in WWW-Search, in *Proc. of the SIGIR '04*, ACM, New York, NY, USA, pp. 478-479.

Grifantini, K. (2011), Eye Tracking for Mobile Control, *Technology Review*.

Handschuh, S. & Staab, S. (2002), Authoring and Annotation of Web Pages in CREAM, in *Proc. of the WWW '02*, ACM, New York, NY, USA, pp. 462-473.

Heflin, J.; Hendler, J. A. & Luke, S. (2003), SHOE: A Blueprint for the Semantic Web, in *Spinning the Semantic Web*, pp. 29-63.

Hori, M.; Kondoh, G.; Ono, K.; Hirose, S.-i. & Singhal, S. (2000), *Annotation-based Web content transcoding*, Computer Networks 33(1-6), pp. 197-211.

Mukherjee, S.; Ramakrishnan, I. V. & Singh, A. (2005), Bootstrapping Semantic Annotation for Content-Rich HTML Documents, in *Proc. of the ICDE '05*, IEEE Computer Society, Washington, DC, USA, pp. 583-593.

Plumbaum, T.; Stelter, T. & Korth, A. (2009), Semantic Web Usage Mining: Using Semantics to Understand User Intentions, in *Proc. of the UMAP '09*, Springer-Verlag, pp. 391-396.

Poole, A. & Ball, L. J. (2005), Eye Tracking in Human-Computer Interaction and Usability Research: Current Status and Future Prospects, in Claude Ghaoui, ed., *Encyclopedia of Human Computer Interaction*, IGI Global, .

Stiefelhagen, R.; Finke, M.; Yang, J. & Waibel, A. (1998), From Gaze to Focus of Attention, in *Proc. of the PUI '98 workshop*, pp. 25-30.