Towards Asynchronous Adaptive Hypermedia: An Unobtrusive Generic Help System

Andreas Putzinger
Johannes Kepler University of Linz
A-4040, Linz, Austria
putzinger@fim.uni-linz.ac.at

Abstract

First, this paper introduces the concept and the upcoming features of Asynchronous Adaptive Hypermedia Systems (AAHS). The design of a concrete system will show how the new principles can successfully be applied to build a generic adaptive help module which can be put on top of existing adaptive or non-adaptive web application without the need of refactoring.

1 Introduction

It is widely acknowledged that Adaptive Hypermedia Systems (AHS) can successfully be applied to several different application domains. The first summarizing taxonomy was published in [Brusilovsky, 1996] and later updated in [Brusilovsky, 2001]. Although several research communities with special focus on adaptivity reacted to the upcoming web by establishing AHS, technologies, commonly referred to as "web 2.0" (cf.[O'Reilly, 2005]) have yet to make their mark on the scene.

The author's current focus of research lies in the concept of out-of-band communication in AHS. In this context the term "asynchronous" is used quite often, whereas "asynchronicity" refers to the actual transmission of data, which takes place independently of the main HTTP request-response cycle. The term "asynchronicity" is used in this paper to refer to the concept of out-of-band communication.

So far it seems that to date very few research groups have published results yet, which would specifically focus on the impact of an out-of-band communication on AHS, related privacy issues or the range of upcoming features. [Barla, 2006] uses asynchronous techniques to get more precise information from the client's context. [Boddu *et al.*, 2007] show how the AHA! framework (cf.[de Bra *et al.*, 2002]) could be enhanced by applying asynchronous concepts.

2 Introduction to the Concept of AAHS

2.1 "Asynchronous Web" in General

Techniques for realizing out-of-band communication in the web are widespread and provide the substantial base for many so–called Rich Internet Applications (RIA, first published in [Allaire, 2002]). A rather well known acronym in this context is AJAX (Asynchronous JavaScript and XML), which denotes a set of technologies often used together. The term itself was first introduced in [Garrett, 2005]. One main goal is to bridge the gap between desktop and web applications in several aspects, mainly in communication and latency matters.

The question may arise as to how far this asynchronous technology can actually provide new possibilities and features. Since asynchronously transferred data could theoretically be also transmitted "synchronously" by transparently bundling it with the next page–request; therefore, all messages have to be collected and cached locally at the client and "piggybacked" on the next HTTP request. Yet, on closer examination this technique is not of equal potential as asynchronous transmissions. If the user, for instance, manually closes the browser window, all accumulated data from the point of entering the page until the point of leaving are lost. In addition, and this represents one of the main drawback, the advantage of a communication with only a short latency is lost. Furthermore, it is still not possible for the server to initiate a call to the client. Thus piggybacking data is not always an alternative to an out-of-band communication. Nevertheless, specific privacy issues mentioned in [Putzinger, 2007] as well as security issues discussed in [Sonntag, 2006; Di Paola, 2006] have to be considered.

2.2 Specific Application in AHS

The combination of AHS on the one side and out-of-band communication techniques on the other opens a great variety of new possibilities in the adaptive hypermedia field and will, at least in the opinion of the author, start a new era of AHS. Enhanced adaptive technologies empowered by stable bidirectional channels between browser and server will, for instance, not only ease the provision of feedback for users[1] and enable advanced usage of subsymbolic data from the client side (cf.[Hofmann *et al.*, 2006; Farzan and Brusilovsky, 2005]), but will also form the base for new kinds of adaptations already known from more traditional desktop adaptive systems.

The upcoming concept of asynchronicity in AHS inherently changes many modules involved in such a system. First, the possibilities for retrieving raw data from the user's context is broadened as far as latency and quantity are concerned. The information about the user's current actions within the browser can, for example, be transmitted almost in realtime. This has a direct influence on the point in time when the process of modeling can take place. Instead of traditionally triggering the modeling process on an incoming page request, data is continously being retrieved and can therefore continuously be processed. Furthermore, triggering adaptations can be done at any time. In traditional AHS the adaptation takes place once when the page is created. Also people involved in the process of evalua-

[1] Amazon, for instance, uses out-of-band calls when users perform product ratings.

tion can heavily benefit from the new quantity and quality of data. This is particularly true for meta adaptive systems, which need to perform self evaluation as part of the standard adaptive behaviour.

The following paragraphs show some examples for new low-level techniques together with their high-level impact in adaptivity. More detailed information can also be found in [Putzinger, 2007]:

Monitoring the user's mouse In some cases it could be helpful to a system to get realtime information about the mouse activity on the clientside. Specifically, this could be the current position of the mouse cursor, the object, text or picture which is currently under the cursor. Also mouse movements or miscellaneous timing data are probably interesting, such as the speed of movements, latencies between (double-)clicks, etc. These data obviously represent valuable raw material for applying methods of subsymbolic user behaviour inference. [Atterer *et al.*, 2006], for instance, suggest to record mouse actions for website usability evaluation.

Monitoring Key Strokes A second category of usage data contains raw key strokes, which could be transmitted either key-by-key in realtime or also grouped. Thus, the application not only gets the finally submitted form data, but also the intermediary states, the involved timings, etc. The user's typing speed together with some other aspects can in some cases be regarded as a good indicator for the user's overall computer skills. A second example refers the possibility of introducing adaptive text completion or recommendation. Adapted to the user model the system suggests words or even complete paragraphs, which fit in with the context and presumably the user's current needs.

The two just mentioned categories of events can individually or combined improve results in plan recognition (cf.[Carberry, 2001; Kristina *et al.*, 1996]) by modeling a clearer picture of the user's current activities.

"Still Active" Messages Receiving the information about key strokes or mouse events out-of-band is an implicit and quite reliable indicator for deducing the binary state if the user is still working with the application. If neither of this data is asynchronously transmitted, explicit "still active" messages could be introduced in order to inform the system about the user's activity state. In particular, this could be used, for example, in e-learning environments, where users gets lots of material to locally read and learn. Although there is no synchronous interaction with the server in regular intervals, the learning platform could use the data about the activity to deduce further information.

On–Demand Data Retrieval Due to the bidirectional communication channel it is also possible for the server to (at least logically) initiate a communication and to push data to the browser without a prior client request. Whenever specific information are needed from the client, the server can simply ask for it. The communication itself is done out-of-band therefore probably even without the user's awareness.

Instant Adaptation The author has developed facilities which allow page fragments to be dynamically exchanged according to results of the underlying adaptive system. The chosen name for this technique is "*instant adaptation*". The effects of changes to the user model, which in turn cause (visual) changes on the user's current page, can instantly be pushed to the client as fragments. Thus, the possibilities for adaptation are becoming much richer. In traditional AH the actual adaptation takes place once when the page is generated. From this point onwards, the page keeps static in respect of adaptation, because it is sent back to the client and no further adaptation takes place until the next complete page is generated.

The technique of instant adaptation enables AHS to exchange selected parts within pages on the fly, although are already shown in the browser, by pushing the new fragment to the client. Some trivial client logics dynamically replaces, adds or modifies the specified part. This technology in context of AHS is new and seems to be quite powerful. Nevertheless, the designer of such a system has to be very careful in using these methods. Studies have shown, that a dynamically changed user interface often confuses people and therefore does not always have a positive impact on the overall user experience.

3 Technical Means of Transport

HTTP is still the default transport protocol in the web. It is good for serving traditional sites and applications but shows some shortcomings since web 2.0 features are required. The concept does not foresee, for instance, to load or send data in the background, least of all on a bidirectional channel. The author has investigated several techniques to overcome some of these limitations. In the rest of this section, some solutions are briefly discussed.

XmlHttpRequest is the most often used technique to implement asynchronous web systems. It is well supported by browser implementations, even on mobile devices, and inherently part of many current application frameworks. It's specification provides support for out-of-band calls from the client to the server, but not vice versa. This can be overcome, for example, by periodically polling the server or more elegantly by using a so called continuous http connection aka. "Streaming AJAX" (cf.[Alinone, 2005]) or "Comet". Comet is the technique finally selected for implementing the system described below.

Besides XmlHttpRequest, several other techniques exist which often require special objects to be included in web pages. These objects are used as intermediaries for the actual communication, as shown in fig.1. Javascript within the webpage sends data to a well-defined interface (①) of the proxy. The object performs the actual communication with the server (②) which takes place independently of the browser's main communication cycles (③). The kind of communication (HTTP, web service call, socket connection, etc.) depends on the kind of embedded object as well as the applying sandbox restriction policies. If the server wants to push data to the page, it is sent to the proxy object, which again informs the page about the received data by calling a specified javascript method. The author has by now successfully tested both Java Applets and Flash movies as technical intermediaries. Furthermore, there is no need to display the embedded objects on the page but they can, in fact, be hidden or even be placed within an (invisible) frame.

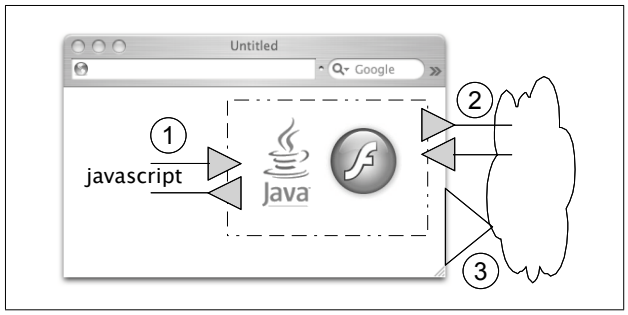

Figure 1: Proxy concept for out-of-band communication

4 Conceptual Design of an AAHS for Help Provision

An adaptive system $\mathcal{S}$ should be designed, which can generically be used within especially input–based web applications or sites without prior information about users. $\mathcal{S}$ should be plugged on top of an existing application $\mathcal{A}$. From a technical point of view, $\mathcal{S}$ and $\mathcal{A}$ should only be losely coupled. A kind of intermediary on any side of the communication adds some lines of javascript code to the pages provided by $\mathcal{A}$ before the pages gets delivered to the user. The additional code instruments the pages to interact with $\mathcal{S}$ autonomously. Furthermore, $\mathcal{S}$ completely relies on asynchronously transmitted data to continuously observe the users' behaviour on the site.

The following features must be carried out by $\mathcal{S}$:

- Generically determine situations in $\mathcal{A}$ when the user needs help in context of the currently performed action. Offer help in these cases.
- Generically determine when the user needs help in using the system, independent of the current context. Offer general help in these situations.
- $\mathcal{S}$ must behave as unobtrusive as possible. This concerns data collection as well as the adaptation processes (providing help).
- $\mathcal{S}$ may not depend on any existing user models or information about users, but works autonomously.

4.1 Data Collection

The following data are asynchronously and independently of $\mathcal{A}$ transmitted to the server, whereas every message contains additional information to uniquely identify the user and the current page:

- Mouse movements
- Keystrokes
- Changes of the currently focused input element
- Global events, such as focusing and blurring the window, scroll actions, etc.

4.2 Modeling and Inference

Every adaptive system needs some specific aspects to be modeled. The subsequent sections show the most important ones for $\mathcal{S}$, whereas the following axioms and definitions hereby apply:

- Let $\mathcal{P}$ globally be the *current selected page* in $\mathcal{S}$.
- Let $\mathbf{U}$ be the set of *all existing users* in $\mathcal{S}$ who have already visited $\mathcal{P}$.

$$\mathbf{U} = \{\mathcal{U}_i | \, \text{is\_system\_user}(\mathcal{U}_i, \mathcal{S}) \text{ and } \text{has\_visited}(\mathcal{U}_i, \mathcal{P})\}$$

- Let $\mathcal{U}$ be *any (single) user* $\in \mathbf{U}$.
- Let IE be the set of all input elements on $\mathcal{P}$ in $\mathcal{S}$.

$$\mathsf{IE} = \{\varepsilon_i | \, \text{is\_input\_element}(\varepsilon_i, \mathcal{P})\}$$

- Let $\text{char\_count}(\varepsilon, \mathcal{U}_i)$ be a function which returns the number of filled–in characters in an input element ε for a user $\mathcal{U}_i$ on $\mathcal{P}$ in $\mathcal{S}$.

User Idle Time

The point in time of the latest user interaction is a very useful information to deduce further data, such as, for example, "time spent reading" in e–Learning systems (cf.[Farzan and Brusilovsky, 2005; Hofmann *et al.*, 2006]). Equ.1 shows the calculation of $\mathcal{U}$'s idle time which is basically the delta of the current time now and the latest point in time last when $\mathcal{U}$ performed an interaction.

$$\text{it}(\mathcal{U}) = \text{now}() - \text{last}(\mathcal{U}) \quad (1)$$

$\mathcal{S}$ currently recognizes mouse interactions (movements), key interactions (releasing a key) and changes to the current window state (focusing, blurring and scrolling). Furthermore, variants of it are defined which only take individual kinds of interactions into account. Therefore, it_{mouse} considers mouse-events only, it_{key} key-events and $\text{it}_{\text{window}}$ only changes to the current window state.

Locus of Attention[2]

Another aspect to model concerns $\mathcal{U}$'s attentional focus, formally expressed as $\text{focus}(\mathcal{U})$. A large portion of research on human attention in digital environments is based on the findings of cognitive psychology. "For example [Raskin, 2000] analyses how single locus of attention, and habit formation have important consequences on human ability to interact with computers. [...] Attention therefore refers to the set of processes by which we select information" ([Roda and Thomas, 2005]). Several sensory-based mechanisms for the detection of users' attention have been employed, including gaze tracking, gesture tracking, head pose and acoustic tracking (cf.[Stiefelhagen *et al.*, 2001]). [Horvitz *et al.*, 2003] propose that sensory-based mechanisms could be integrated with other cues about $\text{focus}(\mathcal{U})$. [Horvitz *et al.*, 2003] also strengthen the theory of uncertainty and suggest to turn to models that can be harnessed to reason about a users attention and about the ideal attention-sensitive actions to take under uncertainty. [Horvitz *et al.*, 2003] think that such models and reasoning can unleash new functionalities and user experiences, which completely aligns with the author's opinion. [Owen, 2006] presents first results in tracking the user's attention by tracking the mouse position in browsers.

In the current preliminary version of $\mathcal{S}$, the possible loci of attention on $\mathcal{P}$ are a priori restricted to input elements $\varepsilon \in \mathsf{IE}$. $\mathcal{U}$'s possible focus is determined by simply taking the currently activated, technically spoken "focused", element on $\mathcal{P}$, which is formally represented by ε_{sel}. If no input element is activated, i.e. ε_{sel} is undefined, $\mathcal{U}$'s attentional focus can not be determined by $\mathcal{S}$ in the first place. It is planned to enhance $\mathcal{S}$ to support more abstract kinds of loci, such high level controls placed on $\mathcal{P}$.

The single information about ε_{sel} is a very unsure hint to determine $\text{focus}(\mathcal{U})$. Therefore, it seems necessary to

[2] In this paper the terms "locus" and "focus" of attention are used synonymously, whereas some authors differentiate more strictly, cf.[Raskin, 2000]

quantify the probability of the correctness, which is expressed as $\mathrm{p_{cor}}$ (see equ.2). The higher the value of the function $\mathrm{p_{cor}}$ is, the higher the chance can be assessed that $\mathrm{focus}(\mathcal{U}) = \varepsilon_{sel}$.

The used assessment function is based on the idea that at moments when $\mathcal{U}$ is typing text into ε_{sel}, the locus of attention is known quite precisely, i.e. the activated form field itself, because $\mathcal{U}$ is obviously just concentrated on writing. The higher, however, the value for $\mathrm{it_{key}}$ gets the lower the probability is that ε_{sel} still represents $\mathrm{focus}(\mathcal{U})$. A reciprocal exponential function has been chosen as the base form to express the probability of correctness, as shown in equ.2. The factor τ is customizable and determines the time span (in seconds) after which $\mathrm{p_{cor}}$ results in 50% probability. τ, therefore, parameterizes the aspect ratio of the curve. Fig.2 shows $\mathrm{p_{cor}}$ for $\tau = 20$ seconds.

$$\mathrm{p_{cor}}(\mathrm{it_{key}}(\mathcal{U}), \tau) = \frac{1}{1 + (\frac{\mathrm{it_{key}}(\mathcal{U})}{\tau})^2} \tag{2}$$

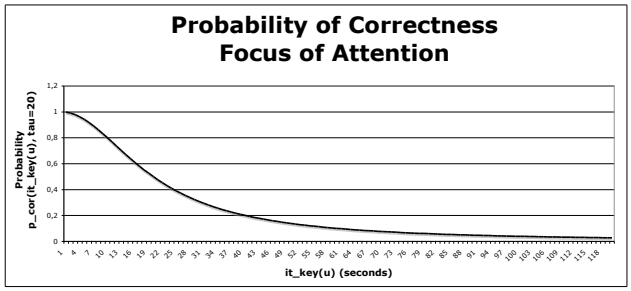

Figure 2: $\mathrm{p_{cor}}(\mathrm{it_{key}}(\mathcal{U}), \tau = 20)$

To finally decide whether $\mathcal{U}$ pays attention to ε_{sel} or not a parameter p_{limit} is introduced which represents the threshold of probability. Equ.3 finally shows the determination of the focus of attention against $\mathcal{U}$ and τ.

$$\mathrm{focus}(\mathcal{U}, \tau) = \begin{cases} \varepsilon_{sel} = \text{undefined} & : \text{undefined} \\ \mathrm{p_{cor}}(\mathrm{it_{key}}(\mathcal{U}), \tau) > p_{limit} & : \varepsilon_{sel} \\ \mathrm{p_{cor}}(\mathrm{it_{key}}(\mathcal{U}), \tau) \leq p_{limit} & : \text{undefined} \end{cases} \tag{3}$$

$\mathcal{S}$ has to determine after how many seconds p_{limit} is reached. For this purpose the inverse function to $\mathrm{p_{cor}}$, $\mathrm{inv_{p_{cor}}}$, is used, which is shown in equ.4.

$$\mathrm{inv_{p_{cor}}}(\mathrm{p_{cor}}, \tau) = \sqrt{\tau^2(\frac{1 - \mathrm{p_{cor}}}{\mathrm{p_{cor}}})} \tag{4}$$

Decision for Context Sensitive Help

Another aspect to model is the actual probability that $\mathcal{U}$ needs context sensitive help for ε_{sel}. This kind of help is offered if $\mathcal{U}$ is spending at least $threshold_{as}$ percent longer attention to ε_{sel} than the average of the other users do. The number of seconds $\mathcal{U}$'s attentional focus lies on ε is called "attention span" and is formally expressed by the function $\mathrm{as}(\mathcal{U}, \varepsilon)$. Equ.5 defines the average attention span $\mathrm{as_{avg}}$ for a specified ε against all users, whereas equ.6 exlicitely disregards $\mathcal{U}$.

$$\mathrm{as_{avg}}(e) = \frac{\sum\limits_{\mathcal{U}_i \in \mathbf{U}} \mathrm{as}(\mathcal{U}_i, \varepsilon)}{|\mathbf{U}|} \tag{5}$$

$$\mathrm{as_{avg}}(\mathcal{U}, \varepsilon) = \frac{\sum\limits_{\mathcal{U}_i \in \mathbf{U}, \mathcal{U}_i \neq \mathcal{U}} \mathrm{as}(\mathcal{U}_i, \varepsilon)}{|\mathbf{U}| - 1} \tag{6}$$

This inference obviously needs training. Thus, it would be an option to e.g.use a default value for the first $minusers$ instead of $\mathrm{as_{avg}}(\varepsilon)$. In this preliminary version of $\mathcal{S}$, $\mathcal{U}$'s individual speed factor is not taken into account. The personal speed, depending, for instance, on overall computer skills, etc., could also contribute to improve $\mathcal{S}$'s performance and to minimize the just mentioned bootstrap problem, which is based on the lack of user data and therefore experience when starting $\mathcal{S}$ for the first time or for new pages. Furthermore, in more advanced versions of $\mathcal{S}$, possible disabilities of $\mathcal{U}$ should be recognized and specially taken into account.

Example

- Let $\mathrm{as_{avg}}$ for ε_{sel} be 18 seconds.
- Let $\mathcal{S}$ be configured to offer $\mathcal{U}$ help in cases where $\mathrm{as}(\mathcal{U}, \varepsilon_{sel})$ is at least 33% higher than the average value. Therefore, $threshold_{as} = 33\%$.
- Let p_{limit} be 60%. This means that $\mathcal{S}$ must be up to 60% sure that $\mathcal{U}$'s focus of attention can be correctly determined by $\mathcal{S}$.
- Let $\mathrm{p_{cor}}$ be parameterized a way that it returns 50% after 12 seconds; therefore, $\tau = 12$.

Effect

- If $as(\mathcal{U}, \varepsilon_{sel})$ gets larger than 24 seconds, $\mathcal{S}$ triggers context sensitive help.
- If e.g.$as(\mathcal{U}, \varepsilon_{sel})$ = 6 seconds and $\mathcal{U}$ stops typing, $\mathrm{p_{cor}}$ after 24 seconds (at this time $\mathrm{it_{key}}(\mathcal{U}) = 18$ seconds) is calculated as shown in equ.7. The resulting value is less than the required 60%, which results in not offering help.

$$\begin{aligned} &\mathrm{p_{cor}}(\mathrm{it_{key}}(\mathcal{U}) = 18, \tau = 12) = \\ &\quad \approx 30,77\% < p_{limit} = 60\% \end{aligned} \tag{7}$$

  The limit for this configuration can be determined by $\mathrm{inv_{p_{cor}}}(\mathrm{p_{cor}} = 60, \tau = 12) =\approx 9,8$. Therefore, the first 10 seconds of $\mathrm{it_{key}}$ are added to the current value of $as(\mathcal{U}, \varepsilon_{sel})$, so that the new value for $as(\mathcal{U}, \varepsilon)$ is 16.
- If $\mathcal{U}$ restarts typing (after whatever time span) and does not blur ε_{sel} in the next 8 seconds, help will be offered after 8 seconds, when $as(\mathcal{U}, \varepsilon_{sel}) = 24$. This happens even if the user only presses one single key and immediatly stops typing again, as shown in equ.8.

$$\begin{aligned} &\mathrm{p_{cor}}(\mathrm{it_{key}}(\mathcal{U}) = 8, \tau = 12) = \\ &\quad \approx 69,23\% \geq p_{limit} = 60\% \end{aligned} \tag{8}$$

$\mathcal{U}$'s progress on $\mathcal{P}$

To determine if $\mathcal{U}$ probably needs context insensitive help $\mathcal{U}$'s overall progress on $\mathcal{P}$ will be taken into account and has therefore to be calculated in a first step. The following simple model is applied:

To determine $\mathcal{U}$'s overall progress on $\mathcal{P}$ the number of characters in each field $\varepsilon \in \mathbb{E}$ is compared with the average number of characters of that field of other users $\mathcal{U}_i \in \mathbf{U}, \mathcal{U}_i \neq \mathcal{U}$. If the factor is higher than 100% ($\mathcal{U}$ has more typed more text than average), the value is defined to be 100%. Equ.9 shows the calculation of the progress factor for a single ε, equ.10 for the whole page $\mathcal{P}$. In later versions of $\mathcal{S}$ more high-level controls will also be taken into account besides text input fields.

$$\mathrm{prog}_{\varepsilon}(\varepsilon, \mathcal{U}) = \frac{\sum\limits_{\mathcal{U}_i \in \mathbf{U}, \mathcal{U}_i \neq \mathcal{U}} \mathrm{char\_count}(\varepsilon, \mathcal{U}_i)}{|\mathbf{U}| - 1} \quad (9)$$

$$\mathrm{prog}(\mathcal{U}) = \frac{\sum\limits_{\varepsilon \in \mathbb{E}} \mathrm{prog}_{\varepsilon}(\varepsilon, \mathcal{U}) \leq 1 \begin{cases} \text{yes} & : \quad \mathrm{prog}_{\varepsilon}(\varepsilon, \mathcal{U}) \\ \text{no} & : \quad 1 \end{cases}}{|\mathbb{E}|} \quad (10)$$

Decision for Context Insensitive Help

Here, the term "context insensitive help" is used in a sense which designates general help about the usage of the overall system, in contrast to specific, context sensitive help for single input elements on $\mathcal{P}$. Generally, context insensitive help should be offered in cases when $\mathcal{S}$ determines that $\mathcal{U}$ may be generally confused with the usage of $\mathcal{S}$.

The concept which is used here to determine the "confusion probability" is not only based on the actual time span $\mathcal{U}$ is spending on $\mathcal{P}$ but mainly on $\mathcal{U}$'s individual progress on $\mathcal{P}$. The total number of $\mathcal{U}$'s interactions on $\mathcal{P}$ is summed up and set in relation to $\mathcal{U}$'s current progress. If the resulting factor differs too much from average, $\mathcal{S}$ triggers the offer to provide general help.

Following functions support the actual calculation:

- Let $\mathrm{t}_{\mathrm{start}}(\mathcal{U})$ be the time from the point of loading $\mathcal{P}$ until $\mathcal{U}$ starts working, whereas "working" is restricted to typing.
- Let $\mathrm{avg}_{\mathrm{t}_{\mathrm{start}}}()$ be a function which returns the average of $\mathrm{t}_{\mathrm{start}}$ for all $\mathcal{U}_i \in \mathbf{U}$.
- Let $\mathrm{num}_{\mathrm{interaction}}(\mathcal{U})$ be the total number of interactions $\mathcal{U}$ performed on $\mathcal{P}$ to reach the current progress.
- Let $\mathrm{avg}_{\mathrm{num}_{\mathrm{interaction}}}(progress)$ be a function which returns the average number of interactions for all $\mathcal{U}_i \in \mathbf{U}, \mathcal{U}_i \neq \mathcal{U}$ to reach the specified $progress$.

Furthermore, the fact has to be taken into account that when entering the page (identified by low progress value) the chance of needing help is much higher than later on. Additionally, the fact that $\mathcal{U}$ needs longer than average to start working is regarded as additional hint for the fact that $\mathcal{U}$ maybe needs help. Because many users orientate first when entering a page, a certain minimum level of deviation may be granted from the beginning.

Equ.11 shows a reciprocal exponential function similar to $\mathrm{p}_{\mathrm{cor}}$ in equ.2. Possible values for the parameter $progress$ lie between 0 ($progress = 0\%$) and 1,0 ($progress = 100\%$). The corresponding function results of $\mathrm{dev}(progress)$ are 1,0 for $progress = 0\%$, i.e. the user has not filled in anything, and $\approx 0,01$ for $progress = 100\%$.

$$\mathrm{dev}(progress) = \frac{1}{1 + 100 progress^2} \quad (11)$$

Several different options exist to consider dev within the help calculation. In $\mathcal{S}$'s implementation dev is used to dynamically determine the threshold for $\mathrm{num}_{\mathrm{interaction}}$ $\mathcal{U}$ may differ from $\mathrm{avg}_{\mathrm{num}_{\mathrm{interaction}}}$. The less the value of $progress$ is the less the threshold is. At the beginning when $\mathcal{U}$ enters $\mathcal{P}$, $\mathcal{S}$ reacts more strict to anormative behaviour and therefore offers help much faster than in phases of advanced progress.

To configure the maximum deviation allowed in case of $progress = 0\%$ and $progress = 100\%$, min_{dev} and max_{dev} are introduced. max_{dev} specifies the maximum percentage $\mathcal{U}$'s behaviour may deviate compared to the average in case of $progress = 100\%$, min_{dev} the maximum percentage $\mathcal{U}$'s behaviour may deviate at the beginning in case of $progress = 0\%$, whereas $0 \leq min_{dev} \leq max_{dev}$. Equ.12 shows the calculation of the general threshold factor. Fig.3 shows the corresponding graph for $min_{dev} = 35\%$ and $max_{dev} = 100\%$. Equ.13 shows the actual threshold value against $\mathcal{U}$ and $progress$.

$$\mathrm{thr}_{\mathrm{factor}}(progress) = (min_{dev} + (max_{dev} - min_{dev})(1 - \mathrm{dev}(progress))) \quad (12)$$

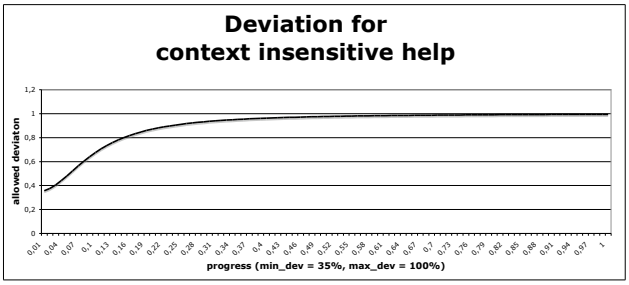

Figure 3: $min_{dev} = 35\%$, $max_{dev} = 100\%$

$$\mathrm{thr}_{\mathrm{value}}(\mathcal{U}, progress) = \mathrm{thr}_{\mathrm{factor}}(progress) \cdot \mathrm{num}_{\mathrm{interaction}}(\mathcal{U}, progress) \quad (13)$$

4.3 Model Application

Due to the genericness of $\mathcal{S}$ mappings between the input elements and the corresponding help texts have to defined. If $\mathcal{S}$ determines that $\mathcal{U}$ probably needs help in filling in ε_{sel}, $\mathcal{S}$ has to look up the mapping record for ε_{sel} to get the corresponding help text. This is afterwards sent to the client, which reacts with dynamically showing an unobtrusive question mark next to ε. If $\mathcal{U}$ clicks on it, the browser shows the received help text.

The content of the context insensitive help is by default the same for all $\mathcal{P}$. If $\mathcal{S}$ determines that $\mathcal{U}$ maybe needs this general kind of help, it simply sends the hint to the client to show a link to the static help pages. This link could be shown with absolute positioning so that it is e.g. always shown in the right upper corner independent of the window's current scroll state. All the user notifications could furthermore be combined with decent audio jingles if $\mathcal{U}$ shows a preference for that.

5 Future Work and Conclusions

This paper presented certain parts of the results in context of the author's ongoing PhD thesis. First, the author gave a brief introduction to the concept of AAHS. Afterwards, a preliminary version of a generic module was designed

which aims to offer both context sensitive and insensitive help by using out-of-band techniques.

Currently, the validation of the shown concepts are prepared. This concerns AAHS in general as well as the presented help system in particular. The evaluation study consists of two independent parts. In a technical evaluation the general feasibility and certain aspects like scalability, latencies and browser-behaviour are investigated and evaluated in order to build AAHS upon a stable and reliable technical basis. The number of simultaneous network connections as well as timings can be modelled quite well "offline" and therefore determined in advance without experiments.

The user-oriented evaluation shows if continuously updated user models in combination with the proposed technique of "instant adaptation" have further impact on the overall quality of AHS. In particular, the presented algorithms for determining $\mathcal{U}$'s locus of attention and the probability of needing help are tested in empirical user studies.

The upcoming field of AAHS aims to bridge the gap between adaptive desktop and hypermedia applications. Therefore, it has to be investigated how traditional and well-established adaptive techniques from the desktop can be applied to web applications by using out-of-band techniques. Due to the won connection directly to the user's desktop the investigation of interpretation of subsymbolic user behaviour offers new interesting challenges. Many new features, challenges and research topics are expected – thus, it's high time, let's launch Adaptive Hypermedia 2.0!

6 Acknowledgements

The work reported in this paper has been partially funded by the Socrates Minerva Adaptive Learning Spaces (ALS) project (229714-CP-1-2006-1-NL-MPP). For information on the ALS project please refer to the projects web site: `http://www.als-project.org`.

References

[Alinone, 2005] Alessandro Alinone. Changing the web paradigm - moving from traditional web applications to streaming-ajax. *published online.*, 2005.

[Allaire, 2002] Jeremy Allaire. Macromedia flash mx - a next generation rich client. *published online.*, 2002.

[Atterer *et al.*, 2006] Richard Atterer, Monika Wnuk, and Albrecht Schmidt. Knowing the user's every move: user activity tracking for website usability evaluation and implicit interaction. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 203–212, New York, NY, USA, 2006. ACM Press.

[Barla, 2006] Michal Barla. Interception of user's interests on the web. In *Proceedings of the International Conference on Adaptive Hypermedia (AH)*, volume 4018 of *Lecture Notes in Computer Science*, pages 435–439. Springer, 6 2006.

[Boddu *et al.*, 2007] Raja Sarat Kumar Boddu, Surendra Prasad Babu Maddali, and Siva Prakasa Rao Alti. Ajax interaction in adaptive hypermedia. In *A3H: Fifth International Workshop on Authoring of Adaptive and Adaptable Hypermedia, UM 2007, 11th International Conference on User Modeling, Corfu, Greece*, 2007.

[Brusilovsky, 1996] Peter Brusilovsky. Methods and techniques of adaptive hypermedia. *User Modeling and User-Adapted Interaction*, 6(2-3):87–129, 1996.

[Brusilovsky, 2001] Peter Brusilovsky. Adaptive hypermedia. *User Modeling and User-Adapted Interaction*, 11(1-2):87–110, 2001.

[Carberry, 2001] Sandra Carberry. Techniques for plan recognition. *User Modeling and User-Adapted Interaction*, 11(1-2):31–48, 2001.

[de Bra *et al.*, 2002] Paul de Bra, Ad Aerts, David Smits, and Natalia Stash. Aha! meets aham, 2002.

[Di Paola, 2006] Stefano Di Paola. Subverting ajax. In *Proceedings of 23rd CCC Conference, Berlin*, 2006.

[Farzan and Brusilovsky, 2005] R. Farzan and P. Brusilovsky. Social navigation support in e-learning: What are the real footprints? In *3rd WS on Intell. Techniques for Web Personalization (ITWP '05). 19th IJC on AI*, 2005.

[Garrett, 2005] Jesse James Garrett. Ajax: A new approach to web applications. *published online.*, 02 2005.

[Hofmann *et al.*, 2006] Katja Hofmann, Catherine Reed, and Hilary Holz. Unobtrusive data collection for web-based social navigation. In *Workshop on the Social Navigation and Community based Adaptation Technologies*, 2006.

[Horvitz *et al.*, 2003] Eric Horvitz, Carl Kadie, Tim Paek, and David Hovel. Models of attention in computing and communication: from principles to applications. *Commun. ACM*, 46(3):52–59, 2003.

[Kristina *et al.*, 1996] H. Kristina, J. Karlgren, A. Waern, N. Dahlback, C. Jansson, K. Karlgren, and B. Lemaire. A glass box approach to adaptive hypermedia, 1996.

[O'Reilly, 2005] Tim O'Reilly. What is web 2.0? *published online.*, 09 2005.

[Owen, 2006] Robert S. Owen. *Tracking Attention through Browser Mouse Tracking*, pages 615–621. Idea Group Reference, Hersey, London, Melbourne, Singapore, 2006. ISBN: 1-59140-562-9 (hardcover) / 1-59140-798-2 (ebook).

[Putzinger, 2007] Andreas Putzinger. Upcoming privacy issues in asynchronous adaptive hypermedia. In Christoph Hoyer, editor, *Proceedings of IDIMT 2007 (forthcoming)*, 2007.

[Raskin, 2000] Jef Raskin. *The Humane Interface: New Directions for Designing Interactive Systems*. Addison Wesley, 2000.

[Roda and Thomas, 2005] C. Roda and J. Thomas. *Encyclopaedia of HCI*, chapter Attention Aware Systems, pages 38–44. IDEA Group, 2005.

[Sonntag, 2006] Michael Sonntag. Ajax security in groupware. In *32nd EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO'06)*, 2006.

[Stiefelhagen *et al.*, 2001] R. Stiefelhagen, J. Yang, and A. Waibel. Estimating focus of attention based on gaze and sound, 2001.