Restyling Website Design via Touch-based Interactions

Luis A. Leiva
ITI – Institut Tecnològic d'Informàtica
Universitat Politècnica de València
Camí de Vera, s/n - 46022 (Spain)
llt@acm.org

Abstract
This paper introduces an ongoing technique for dynamically updating presentational attributes of UI elements. Starting from an existing web layout, the webmaster specifies what elements are candidates to be modified. Then, touch-based events are used as implicit inputs to an adaptive engine that automatically modifies, rearranges, and restyles the interacted items according to browsing usage. In this way, the UI is capable of (incrementally) adapting itself to the abilities of individual users at run-time.

Keywords
Adaptive interfaces, touch tracking, implicit interaction

ACM Classification Keywords
H.5.2 [User Interfaces]: User-centered design; H.5.4 [Hypertext/Hypermedia]: Navigation

General Terms
Experimentation, Design, Human Factors, Algorithms

Introduction
Websites are supposed to satisfy heterogeneous needs of many users. However, traditionally websites have been unable to support that goal [4]. The one-size-fits-all design is a suboptimal solution when no clues about the target population are available, or usage specifications are too general. It is clear that

MobileHCI 2011, Aug 30–Sept 2, 2011, Stockholm, Sweden.
ACM 978-1-4503-0541-9/11/08-09.

manually designing interfaces for each user is impractical and not scalable. Also, continuously performing usability tests to assess new changes on the UI is very time-consuming.

On the other hand, most web pages are visually-oriented and assume that users do not have functional impairments of special requirements. These problems are more aggravating when moving to the mobile domain, where the range of screen sizes and the rendering possibilities are exceedingly large. Although at present the boundaries between the desktop and mobile devices are blurring, mobile is still about poor connections, one-handed use, glancing, interruptions, and (lately) touch screens [11].

Browsers can personalize the presentation of web pages (e.g., modifying font size or applying some accessibility guidelines), but unfortunately the changes they perform operate from a global perspective. We believe that web pages can better adapt themselves to a particular user's needs. Our proposal is to dynamically incorporate slight modifications either to individual widgets (e.g., the first item of a list) or to a specific typology of HTML elements (e.g., all links), based on how users browse websites. This technique is specifically suitable to mobile devices, but it also generalizes to the desktop and traditional browsing.

Background and Related Work

The idea of adapting websites according to user interactions is not new (see, e.g. [13].) However, practical examples have been too scarce so far. Despite considerable debate, automatic adaptation of UIs remains a contentious area [8]. Commonly cited issues with adaptive interfaces include lack of control, predictability, transparency, privacy, and trust [7].

Most approaches employ knowledge that exist prior to interaction, such as physical abilities or familiarity with web browsing. Sometimes such knowledge is acquired by asking the users, e.g., through online surveys. This is known as *explicit feedback*, and may not be as reliable as is often presumed [6]. Other approaches, however, rely on information that users exchange unconsciously, either because of their way of interacting (e.g., scrolling, bookmarking) or because of the information embedded on their browsers (e.g., preferred language, the user-agent string). This is known as *implicit feedback*, and for some tasks it has been proved to be useful — e.g., see [5]. Our approach is based on this kind of feedback. On the one hand, it allows to gather much usage data without burdening the user. On the other hand, though, collected data is potentially noisy and prone to more errors than explicit feedback. For that reason, we let the webmaster to gain control over what elements are going to be modified and *how* (see Framework Formulation).

Designing Alternatives

Probably the major advances in the field of automatic presentational adaptation of UIs are the ones carried out by Gajos and co-authors [8], where adaptation is treated as an optimization problem. However, their experiments were implemented on form-based layouts, by modeling widget constraints, and choosing the best alternatives from a defined set of UI elements (e.g., sliders, combo boxes, radio buttons, etc.). Web layouts are nevertheless a completely different matter. Their dynamic nature per se makes the automatic adaptation a non-trivial problem.

Approaches to Adaptive Design of Websites

Ivory et al. [9] employed learned statistical profiles of good sites to suggest improvements to existing designs; however, changes would be manually

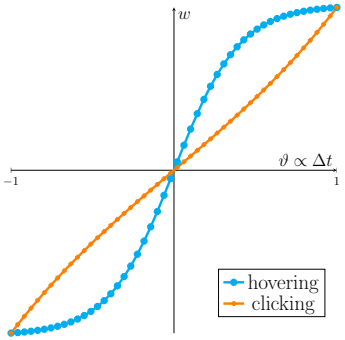

Figure 1: Weighting touch interaction. Hovering is modeled as $w = \tanh(\lambda\vartheta)$, while clicking is modeled as $w = \sinh(\lambda\vartheta)$. The parameter λ allows to tune the slope of both curves. In this case, we used $\lambda = 5$ for hovering and $\lambda = 1$ for clicking.

implemented. Tsandilas and Schraefel [12] introduced an adaptive link annotation technique, although it required the user to perform direct manipulation of a middleware application. Notable mobile-oriented approaches in this direction include the work of Baudisch et al. [2] and Bila et al. [3], where the user must actively modify the layout contents. Kurniawan and co-workers [10] proposed to override the visual layer of a web page with custom Cascading Style Sheets (CSS), but unfortunately updates had to be performed by hand. Now that web standards allow to cleanly decouple presentation and content, we foresee this approach as an alternative to automate the adaptation of web page design.

Touch-based Interactions

We decided to focus on touch interaction for a number of reasons. First, it is the most direct form of interaction —not only on mobile devices— since the finger and the information are only separated by a physical display. Second, it is a user-friendly medium, i.e., there is no need of any mastery to use it. Third, touch-based interactions are more robust than free-moving input devices such as the mouse [1]. Nevertheless, our methodology can be completely transposed to other pointing devices (e.g., styli or touchpads). Finally, the scheme presented here is also fully extensible to other probability distributions and/or different input signals. For instance, the amount of entered text and typing speed could be used to modify elements that have received text focus, such as form fields.

Contributions of this Work

We developed a straightforward method to incorporate information from user interactions to the presentational properties of HTML elements. The novelty of this approach is two-fold: *1)* to let the webmaster decide what elements are going to be adapted; and *2)* to automatically apply slight modifications to the CSS of UI elements based on how the user has interacted with them. In this way, we try to invisibly improve the user-perceived performance towards a page.

Framework Formulation

An HTML element i is susceptible to adapt (the value of) k of its visual properties based on the style function

$$\text{CSS}(k) = \begin{cases} k\,(w_{ik} + 1) & \text{if } w \geq 0, \\ k\,w_{ik} & \text{if } w < 0, \end{cases} \tag{1}$$

where $w_{ik} : M \mapsto \mathbb{R} \in [-1, 1]$ is the weight for the CSS property k belonging to HTML element i, which will be automatically computed by the adaptive engine, based on the contribution of the user interactions ϑ, i.e.,

$$w_k = f(\vartheta_k). \tag{2}$$

$f(\vartheta_k)$ considers the influence of touch hovering and clicking over time increments in a non-linear correlation (Figure 1). To sum up, in this framework:

- Elements are DOM entities (e.g., `div`).
- Properties are CSS features (e.g., `font-size`).
- Weights are values that modify the properties of (interacted) elements. See also System Workflow.

Implementation

The system assigns a unique identifier to the user device, computed by time-stamping a random seed, and stores that value in a cookie. Then the user activity (movements, taps/clicks) is registered in background via DOM events, as well as the interacted HTML elements. The resulting assets are sequences of coordinates, and two rankings of UI elements (hovered and clicked items) ordered by percentage of

(a) Original page **(b)** Automatically adapted design

Figure 2: An example of website design modifications. Changed parts are marked with numbered balloons in Figure 2b. (1) headline text: `font-size`, `padding-top`; (2) navigation menu: `font-size`; (3) welcome paragraphs: `font-size`; (4) 'read more' links: `color`; (5) 'online booking' heading: `color`; (6) submit button: `font-weight`; and (7) 'special menu' div: `margin-top`.

browsing time. Each item in both ranked lists is stored as a CSS selector string, computed by recursively traversing the DOM tree from the interacted element to its nearest parent node having an ID or class attribute. In this way, a compact representation is ensured while facilitating at the same time a ready-to-use CSS reference (see Figure 3). In the worst case, i.e., no node with ID or class is found, the application stores its full DOM position. When a `unload` event is detected, the gathered data are asynchronously transmitted to the adaptive engine, which was coded in PHP, and finally

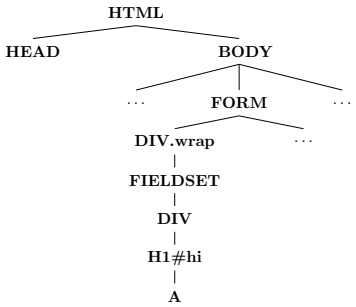

Figure 3: DOM tree traversal example. When the user clicks in the anchor (A) element, the event is propagated to the root (HTML) node. Our system logs this interaction as `[H1#hi>A]`, instead of storing the full path `[HTML>BODY>FORM>DIV.wrap>FIELDSET>DIV>H1#hi>A]`; since the selector `[H1#hi>A]` is enough for unequivocally accessing that element via CSS.

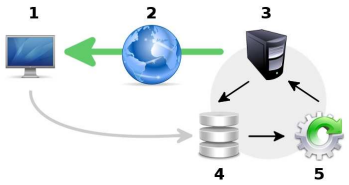

Figure 4: Interaction data are gathered on client side (1) and inserted into a database (4) for later processing by the adaptive engine (5), which, after an iterative process, will notify the web server (3) to deliver the new design via Internet (2).

stored into a MySQL database for better indexing and later querying. Figure 4 shows the system architecture.

System Workflow

When a new user accesses a web page, and therefore no previous interaction data are available, UI elements are rendered as they were originally designed (or, similarly, weights virtually have a null value). Then, while browsing, user interactions are buffered until she leaves the page. At that point data are transmitted back to the web server, and the adaptive engine will assign new values to the weights of the corresponding interacted elements that match those specified by the webmaster. For example, a weight of 0.2 for a `background-color` property will increase the contrast of the affected element by a 20%. Inversely, a weight of -0.046 for a `width` property will decrease the width of the corresponding element by a 4.6%. Since weights are proportional to the temporal variations of user interactions, it is not possible to alter the CSS properties significantly, ensuring thus that changes are incrementally applied. Finally, non-matched elements allows the adaptive engine to have a 'user interaction history', which is useful when the webmaster decides to track new elements (the *cold start* problem.)

Usage Scheme

We needed to name our system to allow web developers accessing its API through the top-level namespace of all browsers, i.e., the `window` object. We found ACE (Adaptive CSS-oriented Engine) to be a consistent, short, and descriptive title. To get the system working, a web page must include it by placing a single JavaScript file either in the `HEAD` or `BODY`. Then, the webmaster should register those elements that will be automatically adapted through the function call `ACE.track([obj1,...,objN])`, where the argument is an array of JavaScript objects, each one having two properties: `selector` (a CSS selector string) and `property` (an array of CSS attributes). For instance, if we want the system to alter the font size and font color of the links in a div element with ID "chef", and the dimensions of the input elements with class "text", we would write this code:

```
var items = [{
    selector: "#chef p a",
    property: ["font-size", "color"]
},{
    selector: "form input.text",
    property: ["width", "height"]
}];
ACE.track(items);
```

Evaluation, Discussion, and Future Work

A preliminary informal study with 12 mobile users revealed that the perceived acceptability regarding the automatically restyled design was positive. Participants were told to freely browse a mockup site (Figure 2) with the ACE system on an HTC Desire. Nine of them reported that they did not notice the introduced changes, and none found distracting such an automatic modifications. We concluded that this framework can be a promising approach for adapting the design of websites, specially when browsing with mobile devices. However, we believe that much work remains to be done.

First of all, we feel that evaluating this kind of adaptation strategy is quite challenging, since no objective metrics can be consistently computed; e.g., in the absence of labeled samples, we cannot use the well-known precision and recall measures; and having to interrupt the normal navigation flow of mobile users to ask them to vote is certainly not an option. We hypothesize, though, that touch interactions inherently encode performance. Thus, if

an adapted design works better than a previous iteration, it should be reflected somehow in the traces of movements, gestures, etc. Nevertheless, one needs to be cautious with this hypothesis, since learnability and familiarity with the UI could be introducing a serious bias. Therefore, our next move will be carrying out a formal in-lab evaluation methodology.

Second, since content is automatically generated, it is likely to be of less quality than human-generated content. Thus, we believe that it would be interesting to assess the influence of such variations in layout design, or use different evaluation viewpoints (e.g., measure the reduction of user effort, compare to other adaptive systems, etc.)

Third, the prototype for the adaptive engine was written in PHP, in order to delegate the computational load to the web server. However, it could be coded entirely in JavaScript, allowing thus to be released as a browser plugin. The user could therefore take control and specify what UI candidates could be modified, or, on the other hand, let the system decide alternatives in a semi-supervised way.

Fourth, it is clear that some CSS properties cannot be adapted based on this framework (e.g., `font-family` or `text-align`). Concretely, re-adaptation will take effect on those properties that vary in a numerical range, e.g., `font-size`, `color`, `height`, or `margin`). We are studying the possibility of mapping semantic properties to a real-valued scales, in order to cope with this limitation.

Finally, redesign decisions are (by now) based on modifications of shape, position, and/or color attributes. We plan to develop more advanced strategies such as inserting related content based on the nature of similar users' interactions.

Acknowledgements

Work partially supported by the Spanish MEC/MICINN under the project MIPRCV (CSD2007-00018).

References

[1] Albinsson, P.-A. and Zhai, S. High precision touch screen interaction. In *Proc. CHI*, pp. 105–112, 2003.

[2] Baudisch, P., Xie, X., Wang, C., and Ma, W.-Y. Collapse-to-zoom: viewing web pages on small screen devices by interactively removing irrelevant content. In *Proc. UIST*, pp. 91–94, 2004.

[3] Bila, N., Ronda, T., Mohomed, I., Truong, K. N., and de Lara, E. Pagetailor: reusable end-user customization for the mobile web. In *Proc. MobySys*, pp. 16–29, 2007.

[4] Brusilovsky, P. and Maybury, M. T. From adaptive hypermedia to the adaptive web. *Commun. ACM*, 45(5):30–33, 2002.

[5] Buscher, G., Dengel, A., and van Elst, L. Eye movements as implicit relevance feedback. In *Proc. EA CHI*, pp. 2991–2996, 2009.

[6] Claypool, M., Le, P., Wased, M., and Brown, D. Implicit interest indicators. In *Proc. IUI*, pp. 33–40, 2001.

[7] Findlater, L. and McGrenere, J. Impact of screen size on performance, awareness, and user satisfaction with adaptive graphical user interfaces. In *Proc. CHI*, pp. 1247–1256, 2008.

[8] Gajos, K. Z., Everitt, K., Tan, D. S., Czerwinski, M., and Weld, D. S. Predictability and accuracy in adaptive user interfaces. In *Proc. CHI*, pp. 1271–1274, 2008.

[9] Ivory, M. Y. and Hearst, M. A. Statistical profiles of highly-rated web sites. In *Proc. CHI*, pp. 367–374, 2002.

[10] Kurniawan, S., King, A., Evans, D., and Blenkhorn, P. Personalising web page presentation for older people. *Interacting with Computers*, 18(3):457–477, 2006.

[11] Rieger, S. and Rieger, B. It's about people, not devices. Available at uxbooth.com. Retrieved March 1, 2011.

[12] Tsandilas, T. and Schraefel, M. C. User-controlled link adaptation. In *Proc. HT*, pp. 152–160, 2003.

[13] Zhang, D. Web content adaptation for mobile handheld devices. *Commun. ACM*, 50(2):75–79, 2007.