SAM: A Modular Framework for Self-Adapting Web Menus

Camille Gobert, Kashyap Todi, Gilles Bailly, Antti Oulasvirta

SAM: A Modular Framework for Self-Adapting Web Menus

Camille Gobert[1] camille.gobert@ens.fr
Kashyap Todi[1] kashyap.todi@gmail.com
Gilles Bailly[2] gilles.bailly@upmc.fr
Antti Oulasvirta[1] antti.oulasvirta@aalto.fi
[1]Department of Communication and Networking, Aalto University [2]Sorbonne Université, CNRS, ISIR

ABSTRACT

This paper presents SAM, a modular and extensible JavaScript framework for self-adapting menus on webpages. SAM allows control of two elementary aspects for adapting web menus: (1) the *target policy*, which assigns scores to menu items for adaptation, and (2) the *adaptation style*, which specifies how they are adapted on display. By decoupling them, SAM enables the exploration of different combinations independently. Several policies from literature are readily implemented, and paired with adaptation styles such as reordering and highlighting. The process—including user data logging—is local, offering privacy benefits and eliminating the need for server-side modifications. Researchers can use SAM to experiment adaptation policies and styles, and benchmark techniques in an ecological setting with real webpages. Practitioners can make websites self-adapting, and end-users can dynamically personalise typically static web menus.

CCS CONCEPTS

• **Human-centered computing** → **Human computer interaction (HCI)**; **User interface design**;

1 INTRODUCTION

Several decades of work on adaptive menus has shown empirical evidence that they can improve the usability of complex interfaces [1, 4, 5, 16, 20]. However, most of these techniques have not been deployed or even released [1]. Technical and practical reasons can be identified. First, to improve adaptive interfaces, it is crucial to isolate and understand the different aspects of the adaptation process [7]. Second, implementing adaptive menus that work outside prototype systems remains a challenging task. Most graphical toolkits (*e.g.* Java Swing, Qt) only provide limited support for customisation [1]. Our goal is to support effective (re)use of theoretical and technical knowledge in adaptive menus to facilitate the transfer of this technology [17].

UI toolkits and frameworks have facilitated the implementation of software interfaces [13]. Similarly, we aim to support implementation and adoption of adaptive menus and interfaces. To this end, we present SAM, an open-source, modular and extendable JavaScript framework for the research and deployment of self-adapting menus on regular web pages and applications. SAM offers logging capability for modern browsers and permits full control of two elementary aspects of an adaptive menu: (1) a **target policy**, which determines relative importance of items and groups found within a menu, based on a user's interaction history, and (2) an **adaptation style**, which specifies the visual changes made while adapting menu items (*i.e.* look and feel). By decoupling the two, it is possible to independently explore different combinations. To facilitate this, SAM includes several readily implemented adapted menus from the literature (*e.g.* [4, 6, 12, 20]). Finally, a key aspect of SAM's design is to ensure privacy: the adaptation process—including user data logging and computation—is entirely local, and requires no server-side storage or modifications.

SAM targets researchers, practitioners, and end-users. Researchers can use SAM to experiment adaptation policies and styles, and compare with previously published ones, in ecological settings (with real webpages). Practitioners can make their websites self-adapting with minor modifications. Finally, end-users can dynamically personalise both the policy and the style of web menus which normally are static.

Our main contribution is the design and implementation of SAM, an open-source framework promoting ecological validity, replicability, and transfer of research on adaptive menus. To demonstrate the capabilities and usage of the framework, we implemented six policies and four styles issued from literature, leading to 24 different adaptive menus. We describe the code required to create a new design and integrate it in an existing web page.

2 RELATED WORK

This work is situated within two specific areas: (1) menu adaptation techniques and (2) automatic adaptation of webpages. We provide a brief overview of each as a precursor and motivation to SAM's design.

Menu Adaptation

Several menu adaptation styles have been proposed [1, 3, 22]. For instance, frequency-ordered or folded menus move the most frequent items to the top of the menu [1, 9]. However this approach does not maintain any visual consistency. To avoid this problem, split menus [16] duplicate the 2–5 most frequent items on the top of the menu so that the bottom part of the menu remains unchanged. This solution has been transposed from a single menu to the whole menu system [11]. Other approaches increase the saliency of some items by manipulating the size [4], the transparency [1], the background colour [20], or the delay of apparition [5, 9] of the items. Except [7], which studied three different menu adaptation styles and two policies, previous works generally rely on one "simple" target policy (*i.e.* item frequency) and do not study the combination of one or several styles. Moreover, these implementations and studies are restricted to specifically-developed software applications. This reduces the potential for adoption and replicability.

Adaptation of Webpages

AI techniques have been used to mine user logs for webpage adaptation [14]. PageGather [15] uses this to automatically create index pages by identifying candidate link sets on websites. However, all data logging and computation is done on the server-side, and the adaptation is constricted to creating new index pages. PageTailor [2] supports client-side customisation of webpages by the user, and saves the results for revisits. Similarly, [21] supports end-user link adaptation on webpages for better information discovery in long menus. These works lack policies for automatic adaptation. [8] automatically adapts pages based on styling issues related to accessibility, but does not continually adapt them based on usage and policies. [18] discusses automatic client-side web page layout adaptation to make them familiar to users based on usage histories. The adaptation is however limited to changes in positions. [10] develops an adaptive engine that took user's touch input to adapt element style based on usage. Similarly to our work, this client-side engine applies a usage-based policy to adapt styles using web technologies. However, the system does not decouple the different aspects of the adaptation, and thus hinders the exploration of different combinations of policies and styles, or further expandability—which are the key goals of our framework.

3 SAM: THE FRAMEWORK

SAM is a client-side framework (Figure 1). By separating out the underlying modules, we enable flexibility and control over the adaptation process. In this section, we provide an overview of each module. The implementation is described in the following section.

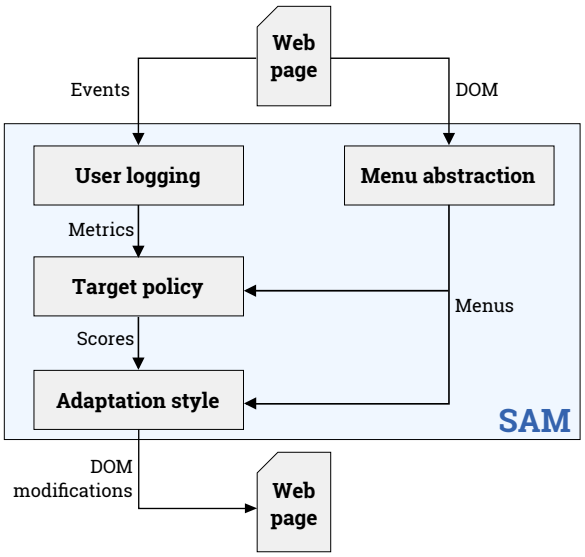

Figure 1: Overview of the interactions between the core modules of the SAM framework.

Menu Abstraction

Webpage menus are defined as sets of elements, whose types and hierarchy tend to differ from one website to another. To address this diversity, SAM builds a single common abstraction of the structure of a menu. This representation is necessary to consistently capture menu usage and adapt them across very different websites.

User Logging and Metrics

SAM captures a user's interaction history with menus, and stores this locally in a database. Currently, SAM records mouse clicks and time spent on each page. To extend logging capabilities, any event that can be captured on a webpage (*e.g.* cursor movements, eye tracking), can be logged by SAM, and used for further computations. SAM then computes usage metrics (*e.g.* click frequency or page visits) from user's interaction history to update the selected target policy. While this history may include erroneous events (such as clicking the wrong item), they eventually become neglected as the system captures more legitimate events and self-corrects itself. Figure 2a shows some logged data for an example scenario.

Target Policies

Scores are assigned to all menu items (and/or groups) based on a target policy. While new policies can be easily integrated into SAM, we include a readily-available set of policies, adapted from literature:

(1) **Item clicks frequency:** Items are assigned normalised scores based on the number of clicks [12, 16].

(2) **Page visits duration:** Items are assigned normalised scores based on the total time spent on the page they link to.
(3) **Page visits frequency:** Items linking to pages visited more frequently are assigned higher scores [18].
(4) **Page visits recency:** Items linking to pages visited more recently are assigned higher scores [18].
(5) **Serial-position curve:** Items are assigned normalised scores combining frequency, recency, and primacy [18].
(6) **AccessRank:** Items are scored according to the AccessRank algorithm [6], which includes recency, frequency, temporal clustering, and time of day as factors as well as a component for stability.

Adaptation Styles

SAM uses the score of each item (target policy) to apply the corresponding adaptation style to items. The number (N) of items to be adapted can either be specified as a fixed number, or as a function of the menu size. Currently, SAM includes the following styles:

(1) **Highlighting:** N items are highlighted using a contrasting background colour[1] [19].
(2) **Item reordering:** Selected N items are moved to the top of the (sub-)menu [12].
(3) **Group reordering:** Selected N entire groups are moved to the top of the menu.
(4) **Folding:** Items with low scores are truncated from the menu [1].

SAM can include additional styles (*e.g.* transparency, size, font, borders, positions, etc.) and can combine them to form composite styles.

4 IMPLEMENTATION

SAM is implemented in Typescript[2], a typed scripting language which is compiled to plain JavaScript. The only external dependency is jQuery[3], which is used to facilitate Document Object Model[4] (DOM) manipulations. To make SAM extendible, it has been split into multiples modules with different responsibilities (Figure 1).

Menu abstraction. Each element of adaptive menus (*e.g.* item, group) is associated with a node in the DOM. For a menu to adapt, a set of jQuery selectors must be provided to SAM, in order to fetch the nodes which form the structure of the menu in the webpage. Each element is then given a unique identifier, used to track them across pages and sessions. The identifier is determined by the node tag, `id` attribute, position among its siblings, and those of its ancestors.

Interaction Logging. The logging of all user interactions has been split between three modules to keep the code easily understandable and simple to extend: The *data logger* logs any event they catch in the database. It currently logs all `click` events fired on menu items, and all page visits (using `beforeunload` events). The *database* serialises its content into the Local Storage of the browser, and un-serializes it on each page load. This choice allows to easily switch to another form of persistent storage (*e.g.* IndexedDB), or to send data to a remote server (*e.g.* for an online study). The *data analyser* improves performances, by only recomputing metrics if the database content has been updated since the last computation, and using a cached version otherwise.

Target Policy. The target policy uses the output of the data analyser and abstract menus content (Figure 1) to compute the scores for each item or/and group and sort them.

Adaptation Style. The adaptation style takes the output of the aforementioned policy and a list of abstracted menus to modify the DOM and apply the desired effect to the target items or/and groups. To be compatible with SAM, each style must implement two methods: one to apply the effect of the style, and one to cancel it.

Privacy. User data, target policy, and adaptation styles are stored, computed, and applied locally, on the user's browser.

Scalability. SAM can adapt web menus (desktop or mobile) of all sizes, with or without groups, with hundreds of items. The main threshold is the complexity of the target policy and the adaptation style—which can run arbitrarily costly computations. However, modern web browsers heavily optimise JavaScript code they run, and any combination of currently implemented policy and styles can smoothly adapt menus with dozens of groups and hundreds of items in a few milliseconds[5]. Furthermore, although the typical Local Storage of a browser has limited capacity (~ 5–10 MB per domain), the database of SAM can still store 2,000 to 10,000 visits and clicks before exceeding the available space. This limit could be easily bypassed by using IndexedDB instead of Local Storage.

5 USAGE AND EXTENSION

We release SAM as an open-source framework at https://github.com/aalto-ui/sam. It can be adopted and/or extended by different categories of users, based on their objectives.

Developers. Developers can edit the sources of a website to include SAM, and turn their static menus into adaptive menus. They first include the JavaScript library and CSS file in the HTML sources of any page with menus to adapt:

[1] Or any other effect which can be applied with CSS.
[2] https://www.typescriptlang.org
[3] https://jquery.com; the *slim* version suffices for SAM.
[4] https://www.w3.org/DO/
[5] In our tests, SAM computations on wikipedia.org took 0.077s (average over 10 page loads).

Item label	Nb. clicks	Duration (s)
Main page	2	74
Contents	0	0
Featured contents	6	29
Current events	0	0
Random article	10	–
Donate to Wikipedia	0	0
Wikipedia store	0	0

(a) Interaction history.

(b) Highlighting by Visit duration.

(c) Highlighting by Click frequency.

(d) Item reordering by Click frequency.

Figure 2: Various adaptive menus produced by SAM on Wikipedia, given a single interaction history (a). Changing the target policy (b → c) or the adaptation style (c → d) both result in different adaptive menus.

```
<link rel="stylesheet" href="css/sam.css"></link>
<script type="text/javascript" src="js/sam.js"></script>
```

They then initialise the framework by calling the static builder method `fromSelectors` on the global `SAM` object—which is exposed in the global scope by `sam.js`. This step requires the DOM to be fully loaded:

```
$(document).ready(() => {
  SAM.fromSelectors(".menu", ".group", ".item");
});
```

Researchers. Researchers can extend SAM with new policies and styles. To do so, they must implement the `TargetPolicy` or `AdaptationStyle` interfaces, add the new class to the right internal array, and recompile the framework (by running `grunt`[6]). For example, one could add a `Magnify` style which increases the font size of the 3 elements with the highest scores. By adding a CSS rule which increases the font size of all elements with class `sam-magnified`, the following snippet is sufficient to implement this style:

```
public class Magnify implements AdaptationStyle {
  readonly name = "Magnify";
  readonly    N = 3; // Number of items to select

  apply(menuManager, policy, dataManager) {
    // Select top N items ranked by the policy
    let items = policy.getSortedItems(menuManager, dataManager)
                      .slice(0, N);
    for (let item of items) {
      item.node.addClass("sam-magnified");
  } }

  cancel() {
    $(".sam-magnified").removeClass("sam-magnified");
} }
```

For convenience, abstract classes with partial implementations are provided. The following snippet illustrates how this allows to add a new target policy to SAM by implementing only one method:

```
public class ExamplePolicy extends DefaultTargetPolicy {
  readonly name = "ExamplePolicy";

  getSortedItemsWithScores(menuManager, dataManager) {
    let items = menuManager.getAllItems();
    let itemsWithScores = items.map((item) => {
      return {item: item, score: 0};
    });

    // --- Code to compute item scores ---

    return itemsWithScores;
} }
```

End-users. End-users can use SAM to adapt any web menu. This is achieved by injecting the SAM JavaScript library, the CSS file, and an initialisation script, in selected webpages, using a freely-available browser extension[7]. Moreover, by building a public repository of initialisation scripts (*e.g.* one per domain), end-users would not even have to specify the selectors to use by themselves. This could make menu adaptation on the web seamless and accessible to anyone, without requiring any technical knowledge.

6 CONCLUSION AND OUTLOOK

This paper has presented SAM, a JavaScript framework for developing and deploying adaptive menus on the web. It allows to compose policies and styles to easily explore new types of adaptive techniques. In the future, we aim to provide additional technical resources for inquiring user experience via studies which include adaptive menus supported by SAM. Furthermore, we intend to extend the framework with more policies and styles, which better cover existing literature.

[6] https://gruntjs.com/

[7] We used Code Injector: https://addons.mozilla.org/en-US/firefox/addon/codeinjector/

ACKNOWLEDGMENTS

This work has been funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement 637991).

REFERENCES

[1] Gilles Bailly, Eric Lecolinet, and Laurence Nigay. 2017. Visual menu techniques. *ACM Computing Surveys (CSUR)* 49, 4 (2017), 60.

[2] Nilton Bila, Troy Ronda, Iqbal Mohomed, Khai N. Truong, and Eyal de Lara. 2007. PageTailor: Reusable End-user Customization for the Mobile Web. In *Proceedings of the 5th International Conference on Mobile Systems, Applications and Services (MobiSys '07)*. ACM, New York, NY, USA, 16–29. https://doi.org/10.1145/1247660.1247666

[3] Sara Bouzit, Gaëlle Calvary, Denis Chêne, and Jean Vanderdonckt. 2016. A design space for engineering graphical adaptive menus. In *Proceedings of the 8th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*. ACM, 239–244.

[4] Andy Cockburn, Carl Gutwin, and Saul Greenberg. 2007. A Predictive Model of Menu Performance. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '07)*. ACM, New York, NY, USA, 627–636. https://doi.org/10.1145/1240624.1240723

[5] Leah Findlater, Karyn Moffatt, Joanna McGrenere, and Jessica Dawson. 2009. Ephemeral adaptation: The use of gradual onset to improve menu selection performance. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1655–1664.

[6] Stephen Fitchett and Andy Cockburn. 2012. Accessrank: predicting what users will do next. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2239–2242.

[7] Krzysztof Z Gajos, Mary Czerwinski, Desney S Tan, and Daniel S Weld. 2006. Exploring the design space for adaptive graphical user interfaces. In *Proceedings of the working conference on Advanced visual interfaces*. ACM, 201–208.

[8] S. H. Kurniawan, A. King, D. G. Evans, and P. L. Blenkhorn. 2006. Personalising web page presentation for older people. *Interacting with Computers* 18, 3 (May 2006), 457–477. https://doi.org/10.1016/j.intcom.2005.11.006

[9] Dong-Seok Lee and Wan Chul Yoon. 2004. Quantitative results assessing design issues of selection-supportive menus. *International Journal of Industrial Ergonomics* 33, 1 (2004), 41–52.

[10] Luis A. Leiva. 2011. Restyling website design via touch-based interactions. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services - MobileHCI '11*. ACM Press, Stockholm, Sweden, 599. https://doi.org/10.1145/2037373.2037467

[11] Wanyu Liu, Olivier Rioul, Joanna Mcgrenere, Wendy E Mackay, and Michel Beaudouin-Lafon. 2018. BIGFile: Bayesian Information Gain for Fast File Retrieval. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, 385.

[12] Jeffrey Mitchell and Ben Shneiderman. 1989. Dynamic versus static menus: an exploratory comparison. *ACM SigCHI Bulletin* 20, 4 (1989), 33–37.

[13] Brad A. Myers. 1995. User Interface Software Tools. *ACM Trans. Comput.-Hum. Interact.* 2, 1 (March 1995), 64–103. https://doi.org/10.1145/200968.200971

[14] Mike Perkowitz and Oren Etzioni. 1997. Adaptive Web Sites: an AI Challenge. In *IJCAI*.

[15] Mike Perkowitz and Oren Etzioni. 1999. Towards Adaptive Web Sites: Conceptual Framework and Case Study. In *Proceedings of the Eighth International Conference on World Wide Web (WWW '99)*. Elsevier North-Holland, Inc., New York, NY, USA, 1245–1258. http://dl.acm.org/citation.cfm?id=313234.313022

[16] Andrew Sears and Ben Shneiderman. 1994. Split menus: effectively using selection frequency to organize menus. *ACM Transactions on Computer-Human Interaction (TOCHI)* 1, 1 (1994), 27–51.

[17] Alistair Sutcliffe. 2000. On the Effective Use and Reuse of HCI Knowledge. *ACM Trans. Comput.-Hum. Interact.* 7, 2 (June 2000), 197–221. https://doi.org/10.1145/353485.353488

[18] Kashyap Todi, Jussi Jokinen, Kris Luyten, and Antti Oulasvirta. 2018. Familiarisation: Restructuring Layouts with Visual Learning Models. In *23rd International Conference on Intelligent User Interfaces (IUI '18)*. ACM, New York, NY, USA, 547–558. https://doi.org/10.1145/3172944.3172949

[19] Theophanis Tsandilas et al. 2005. An empirical assessment of adaptation techniques. In *CHI'05 Extended Abstracts on Human Factors in Computing Systems*. ACM, 2009–2012.

[20] Theophanis Tsandilas et al. 2007. Bubbling menus: a selective mechanism for accessing hierarchical drop-down menus. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 1195–1204.

[21] Theophanis Tsandilas and m. c. schraefel. 2003. User-controlled Link Adaptation. In *Proceedings of the Fourteenth ACM Conference on Hypertext and Hypermedia (HYPERTEXT '03)*. ACM, New York, NY, USA, 152–160. https://doi.org/10.1145/900051.900086

[22] Jean Vanderdonckt, Sarah Bouzit, Gaëlle Calvary, and Denis Chene. 2018. Cloud Menus, a Circular Adaptive Menu for Small Screens. In *IUI'18: 23rd International Conference on Intelligent User Interfaces Proceedings*.