# AB4Web: An On-Line A/B Tester for Comparing User Interface Design Alternatives



JEAN VANDERDONCKT, Université catholique de Louvain, Belgium
MATHIEU ZEN, Université catholique de Louvain, Belgium
RADU-DANIEL VATAVU, MintViz Lab, MANSiD, University Ştefan cel Mare of Suceava, Romania

We introduce AB4Web, a web-based engine that implements a balanced randomized version of the multivariate A/B testing, specifically designed for practitioners to readily compare end-users' preferences for user interface alternatives, such as menu layouts, widgets, controls, forms, or visual input commands. AB4Web automatically generates a balanced set of randomized pairs from a pool of user interface design alternatives, presents them to participants, collects their preferences, and reports results from the perspective of four quantitative measures: the number of presentations, the preference percentage, the latent score of preference, and the matrix of preferences. In this paper, we exemplify the AB4Web tester with a user study for which $N$=108 participants expressed their preferences regarding the visual design of 49 distinct graphical adaptive menus, with a total number of 5,400 preference votes. We compare the results obtained from our quantitative measures with four alternative methods: Condorcet, de Borda count starting at one and zero, and the Dowdall scoring system. We plan to release AB4Web as a public tool for practitioners to create their own A/B testing experiments.

Additional Key Words and Phrases: A/B testing; Design by exploration; Experiment; Graphical adaptive menus; Multivariate testing; Overall Evaluation Criterion; Randomized procedure; User interface variants.

**ACM Reference Format:**
Jean Vanderdonckt, Mathieu Zen, and Radu-Daniel Vatavu. 2019. AB4Web: An On-Line A/B Tester for Comparing User Interface Design Alternatives. *Proc. ACM Hum.-Comput. Interact.* 3, EICS, Article 18 (June 2019), 28 pages. https://doi.org/101145.3331160

## 1 INTRODUCTION AND TERMINOLOGY

Designing User Interfaces (UI) is an iterative process, where the original prototype is being continuously updated and improved towards the final release. More often than not, designers or stakeholders come up with several UI alternatives as a result of "designing by exploration," a method that consists in designing multiple UI alternatives for an interactive application, followed by selecting the most suitable one according to predefined criteria, such as aesthetics, performance, preference [27]. Once the UI design alternatives are available and testable, they are submitted to a panel of representative end-users to collect their preferences and feedback. This process represents an instance of User Acceptance Testing (UAT), which consists in verifying that a part of the software product, such as its user interface, is acceptable for end-users, as opposed to System Testing.

Authors' addresses: Jean Vanderdonckt, Université catholique de Louvain, LouRIM Institute, Place des Doyens, 1, 1348, Louvain-la-Neuve, Belgium, jean.vanderdonckt@uclouvain.be; Mathieu Zen, Université catholique de Louvain, Exploitation des produits et services du système d'information (SIPS), Pierre & Marie Curie, rue du Compas 1, 1348, Louvain-la-Neuve, Belgium, mathieu.zen@uclouvain.be; Radu-Daniel Vatavu, MintViz Lab, MANSiD, University Ştefan cel Mare of Suceava, 13 Universitatii, Suceava, State, 720229, Romania, radu.vatavu@usm.ro.


2573-0142/2019/6-ART18 $15.00
https://doi.org/101145.3331160

### 1.1 Bivalued Split Testing

One way to conduct this process is represented by *A/B testing*, also referred to as "split testing," a between-subjects experimental design that presents participants with pairs of two UI design alternatives, referred to as *variant A* and *variant B*, to determine which variant is superior according to some *Overall Evaluation Criteria* (OEC). These criteria are defined in the form of qualitative or quantitative measures, usually referred to as *dependent variables* or *user's responses* [31]. The OEC could actually cover various aspects, such as effectiveness, goodness-of-fit, overall performance, users' preferences, or social acceptance. When an OEC addresses a qualitative aspect, such as preference, it can still be transformed into a quantitative measure by means of ratings, rankings, and scales. A/B testing is known for its simple application with very good results in practice.

A *factor* expresses a controllable experimental variable that is expected to affect positively or negatively the OEC. The factor presents a *stimulus* with different *values*. In the original A/B testing, there is only one factor with two values (*monovariate bivalued*), thus producing two variants, A and B. A *variant* consists of an alternative being tested by assigning values to the factors: it is either the *control* or one of the *treatments* [31]. The control designates the existing variant being compared against the new treatments. When there is no control, all variants are simply considered as treatments. The two variants are simultaneously presented to isolate the factor's effect on the Overall Evaluation Criteria.

For example, Fig. 1 depicts an original split testing where the search box of a web site (the stimulus) could be located (the factor) either at the top right (first value in variant A) or in the middle of the screen (second value in variant B). The population sample is then equally divided into two parts, each using a particular variant. In this example, 50% of the sample will be directed to variant A here considered as the control, while the rest will visit the variant B, here considered as the treatment challenging the control. One computes the conversion rate (the OEC) which equals the ratio between the amount of search box usages and the amount of web page visits. In our example, 46% of participants use the search engine located on the top right while 32% of participants use it when located in the middle. Thus, the variant A is the winner, B is the looser.

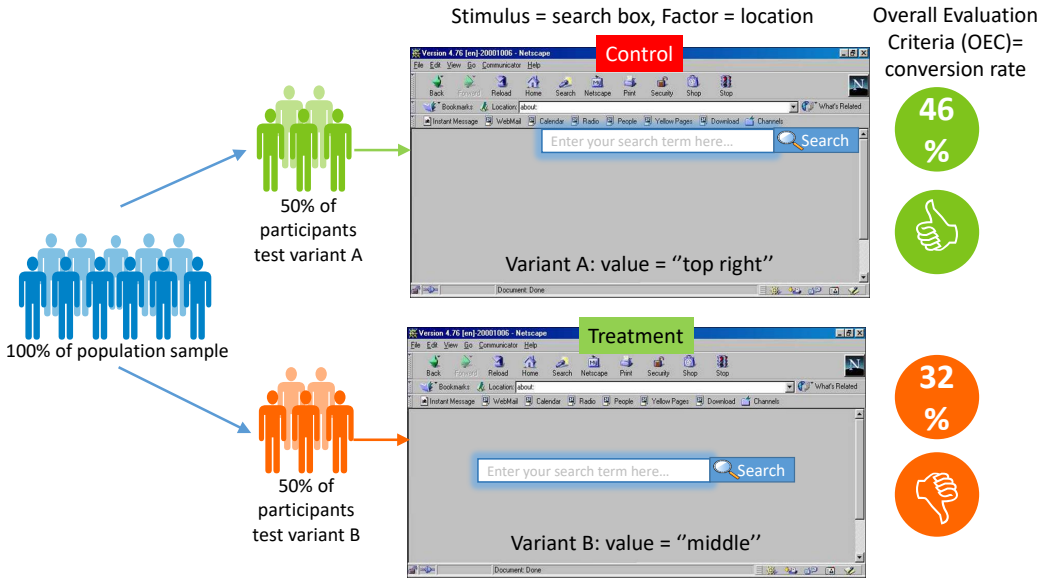


Fig. 1. The original Split Testing.

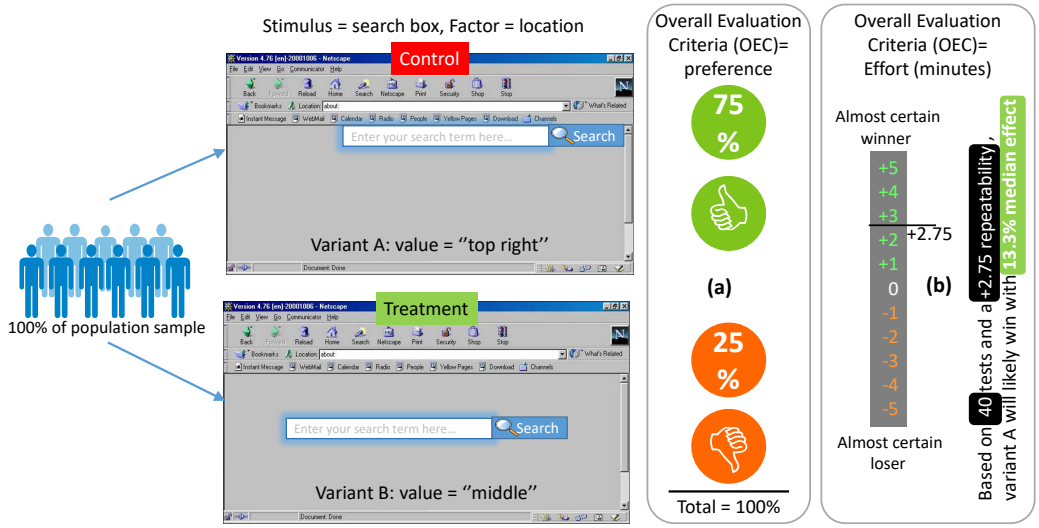


Fig. 2. The original Split Testing with full population sample: (a) for preference, (b) by pattern.

Instead of dividing the sample into the amount of variants (two groups of 50% of participants in Fig. 1), all participants could be engaged directly in the full A/B testing, see Fig. 2: 100% of the population sample is confronted to both variants, participants are asked to express their overall preference for one of these two possible variants (Fig. 2a). The stimulus, the factor, and the values remain the same while the OEC considers the participants' preference: 75% of participants prefer the variant A while 25% prefer the variant B, thus totaling 100%. Fig. 2b represents the same testing with another OEC: the effort of using the stimulus. The results are expressed as a design pattern emerging from statistical data inspired by GooDUI[1]: *the variant A will likely win over variant B with* 13.3% *median effect for a* +2.75 *repeatability based on 40 A/B tests performed so far.*

## 1.2 Multivalued Split Testing

When a factor is assigned to more than two values, the split testing becomes *multivalued*. In our example, the search box could be located in many other positions in the web page layout and we do not know which location is the most appreciated by end users. Fig. 3 depicts this situation where the web page layout is discretized into a grid of seven rows and eight columns, therefore generating fifty-six potential locations. Of course, certain locations are more likely to be accepted or rejected. If a split test should be rigorously conducted, 56 treatments should be produced, the location expected by participants could be graphically depicted as a heat map.

[1]See http://www.goodui.org

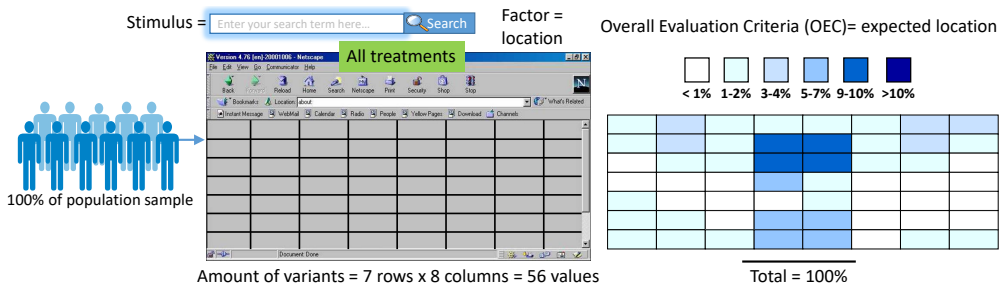


Fig. 3. A multivalued Split Testing with full population sample

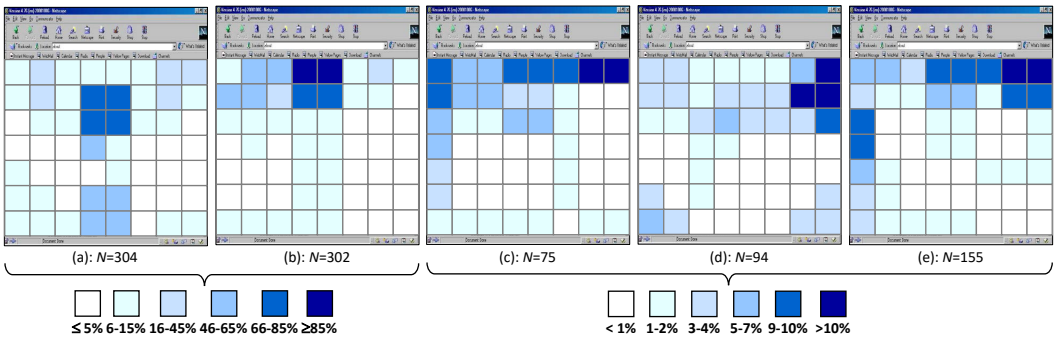


Fig. 4. The expected location of the search box for a web application, represented in the form of a heatmap based on percentage: (a) results from the initial study of Bernard [9], (b) the revision for electronic commerce [10], (c) the replicated study [42], for participants of Southeast Asian culture [4], and (e) for Indian users [43].

Bernard [9] addressed this particular problem in a study to determine suitable locations for the search box with results represented using a heatmap of expected locations (not necessarily preferences); see Fig. 4a. Results showed that participants shared a mental model of the web page layout, revealed by the fact that they located the search box in the top center or at the bottom center of the web page. However, this expected location could dramatically change depending on several factors: *the type of web page* (see Fig. 4b) for the heatmap representative of electronic commerce web sites [10]), *time* (see Fig. 4c for the results of the same study, but replicated three years later [42]), *cultural background* (see Fig. 4d for results reflective of preferences of participants from Southeast Asian countries [4]), and *country* (see Fig. 4e which illustrates the locations expected by Indian users for information web sites [43]).

### 1.3 Multivariate Split Testing

The previous example shows that the original A/B testing has limits when more than two values need to be compared for the same factor. Instead of presenting all variants, which could be overwhelming when many values exist for a factor, one pair of selected variants could be presented at a time. Fig. 5a depicts a monovariate multivalued split testing where pairs of variants are presented to participants. The total number of pairs that can be formed is calculated as: $\frac{n\times(n-1)}{2} = \frac{56\times(56-1)}{2} = 1,540$ pairs. Let us imagine that two factors are considered simultaneously: the location with its 56 values and the style of the search box with 4 values, *i.e.*, edit box only, with text push button, with icon push button, or with text and icon push button (see Fig. 5b). Then, the total number of pairs becomes $\frac{n\times(n-1)}{2} = \frac{(56\times4)\times((56\times4)-1)}{2} = 24,976$.

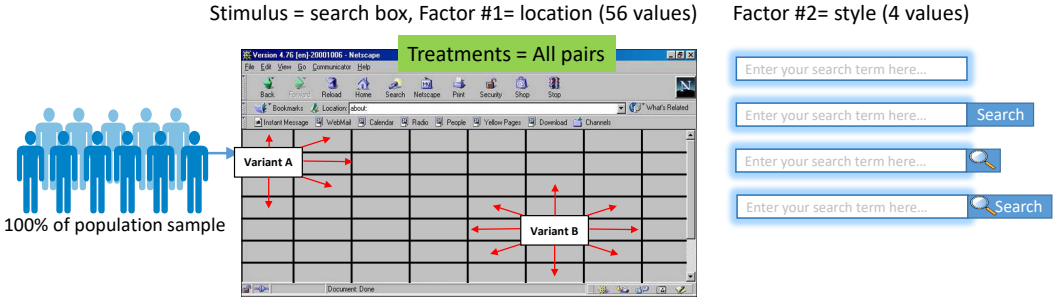


Fig. 5. A multivariate Split Testing by pairs: (a) with one factor, (b) with two factors.

On one hand, presenting variants that involve more than one factor (called *multivariate*) with more than two values per factor (called *multivalued*) may lead to user responses that are difficult to associate to individual changes in each factor and, thus, fall outside the simple application of the split testing method [19]. On the other hand, A/B testing needs collecting responses to a number of questions that depends quadratically on the pairs of variants. In the last example, administering a very large amount of questions will lead to boredom, fatigue, and/or deficits of attention or of commitment to completing the study, with direct impact on the quality of the collected feedback and the number of volunteers willing to participate in the study. Thus, simple A/B testing quickly becomes prohibitive, even when not all variants are practically feasible or meaningful.

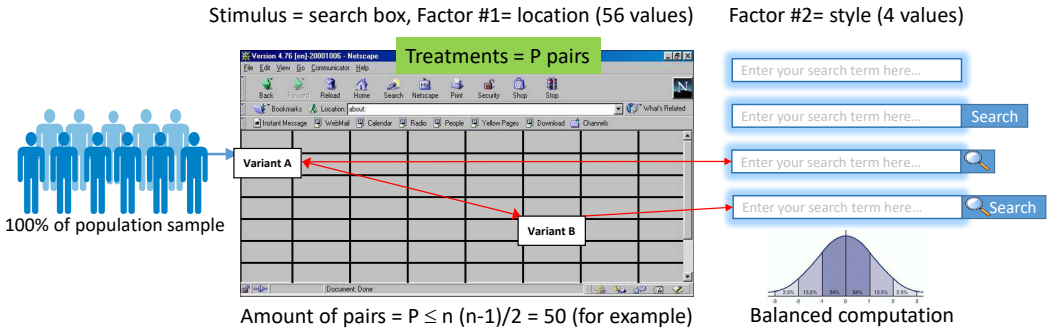


Fig. 6. A randomized multivariate split testing with balance.

To address these problems, we provide the following contributions:

(1) We introduce the *randomized split testing*, a new version of the multivariate and multivalued A/B testing [31], where participants are presented with a reasonable number of variants, while making sure that a sufficient number of responses are collected for each A/B pair over all participants, thanks to a balancing method computed (see Fig. 6).
(2) We introduce AB4Web, a web-based engine that implements an on-line instantiation of the randomized split testing, where participants are presented with a number of pairs determined by the designer and where they express their preference by selecting the variant of their choice.
(3) AB4Web is specifically tailored to manipulate UI design alternatives as variants, which are represented by static (*e.g.*, screen shots, mock-ups, wireframes) or dynamic UI design artifacts (*e.g.*, animations, prototypes, or a URL).
(4) AB4Web computes and reports four measures of user preference, *i.e.*, the number of presentations, the preference percentage, the latent score of preference, and the matrix of preferences.
(5) To illustrate AB4Web for a practical scenario, we conducted an on-line A/B testing to collect users' preferences regarding various designs of Graphical Adaptive Menus (GAMs). We chose this application domain because menu selection represents a fundamental task in virtually any modern interactive application, and the quality of the adaptation determines how much time users actually spend searching for and selecting items from the menu.

The remainder of this paper is structured as follows: Section 2 provides a comparative analysis of selected software applications for conducting A/B testing and highlights the unique features of AB4Web; Section 3 describes the AB4Web software architecture and system walkthrough, and defines the four measures of preference computed by AB4Web; Section 4 demonstrates AB4Web with a user study on GAMs; Section 5 discusses advantages and limitations intrinsic to the randomized version of A/B testing and suggests ways to address the limitations in future work.

| Name | Area | Stimulus | Factors | Values | OEC | Output measures | Method |
|---|---|---|---|---|---|---|---|
| HubSpot | Customer relationship man. | Electronic mail | Message style | Two values | Call-to-action | Action success rate | Monovariate simple A/B testing |
| Optimizely | Electronic commerce | Web page | One web page element (e.g., a widget, an image) | Two to many values for a property of the web page element or two different elements | User action performed on the web page element | Click through rate (not extensible) | Monovariate simple A/B testing |
| Pardot | Marketing | Web page | One web page element at a time (e.g., headers, headlines, landing pages, form fields). | Two values of one property | Usage data on element | Click through rate (not extensible) | Monovariate simple A/B testing |
| GoodUI | Web site patterns | GUI pattern | One to many GUI elements | One to many widget properties | Own empirical data (no software) | Winner/looser Median effect (not extensible) | Multivariate simple A/B testing |
| MoocLet | Distance learning | Adaptive web pages | Production rules for questions/answers | Two to many conclusions for production rules | User's answer to questions | Correct answer rate | Multivariate simple A/B testing |
| Assist-Ments | eLlearning | Web exercises | Exercise policy | Many policies based on learner's profile | User's answer to questions | Correct answer rate | Multivariate simple A/B testing |
| mTurk | Multimedia application | Multi-media contents | One to many multimedia contents | Many contents | Qualitative criteria (e.g., goodness of fit) | Criteria positive rate | Multivariate simple A/B testing |
| IPEAD | Games | Game screens | One to many game elements (e.g., command layout) | Many aesthetic changes | Qualitative criteria | Criteria positive rate | Multivariate simple A/B testing |
| AB4Web | UI design by explo-ration | Any UI artefact | One to many UI artefacts | One to many artefacts or elements | Qualitative ranking (e.g., user's preference) | Number of presentations, Preference percentage, Latent score of preference, Matrix of preferences (extensible) | Multivariate randomized A/B testing |

Table 1. Comparison of existing A/B testing software.

## 2 RELATED WORK

A wide variety of software for conducting A/B testing, both online and offline, is available to practitioners. For example, Capterra[2] compares product reviews and features (*e.g.*, audience targeting, campaign segmentation, heatmaps, multivariate testing, split testing) of several A/B testing software in the field of marketing. In this section, we conduct a comparative analysis of selected, representative software for running A/B testing according to the following criteria: applicability range, type of stimulus, factors and values, OEC, output measures, and the specific implementation of A/B testing; see Table 1:

(1) HubSpot[3] compares two styles of a promotion electronic mail in Customer Relationship Management (CRM) by recording the call-to-action that could be triggered from the email. Thus, it implements a monovariate bivalued A/B testing procedure.

(2) Optimizely[4] represents the standard software for A/B testing in electronic commerce. A single element of a web page is subject to two variants, for instance a "Submit" button with two different formats and styles. Optimizely computes the ratio between the number of times a specific button was clicked out of the number of times that button was presented to users. In this way, marketing practitioners can understand what design attracts customers more. Although the default setting is to compare just two conditions for one factor, Optimizely can accommodate more conditions, but each time only two variants are presented.

(3) Pardot[5] is a marketing automation software implementing A/B testing to evaluate two web page variants based on their header images, headlines, landing pages, form fields, etc. For example, a designer may introduce a new treatment by changing the landing page of an institutional web site. Pardot collects usage statistics, such as *"the click-through-rate has increased by 20% with respect to the control variant."* The designer uses this information to keep the current treatment as a new control and test against other treatments.

(4) GoodUI[6] provides a catalogue of patterns reporting the final results of several tests, such as *"Based on 4 tests comparing variant A with a coupon field in a shopping cart and variant B without, with a repeatability of* +2.75*, the B variant is likely to win with a 13.3% median effect,"* thus recommending to remove the coupon field from the shopping cart UI. While A/B testing patterns are interesting, GoodUI does not permit conducting A/B testing on-line. Thus, GoodUI acts more like a guide for applying existing A/B testing results rather than being a software for conducting A/B testing per se.

(5) MOOClet [48] consists of an engine for running A/B tests in the context of distance learning based on adaptation policies, such as production rules. Different questions, which can be considered as variants of A/B testing, are automatically selected and presented to participants. The answers and the adaptation policy determine which variants are presented next. The ratio of correct responses is computed for each student. Since multiple-answer questions are permitted, MOOClet is multivariate and multivalued.

(6) AssistMents [28, 49] is an eLearning system that offers personalized A/B testing to students depending on their profile. Instructors define the learning policy [48] that is used to automatically create an online A/B test for quickly deploying exercises. The system reports the ratio of correct responses.

[2] https://www.capterra.com/ab-testing-software/compare/
[3] http://www.hubppot.com
[4] http://www.optimizely.com
[5] http://www.pardot.com
[6] http://www.goodui.org

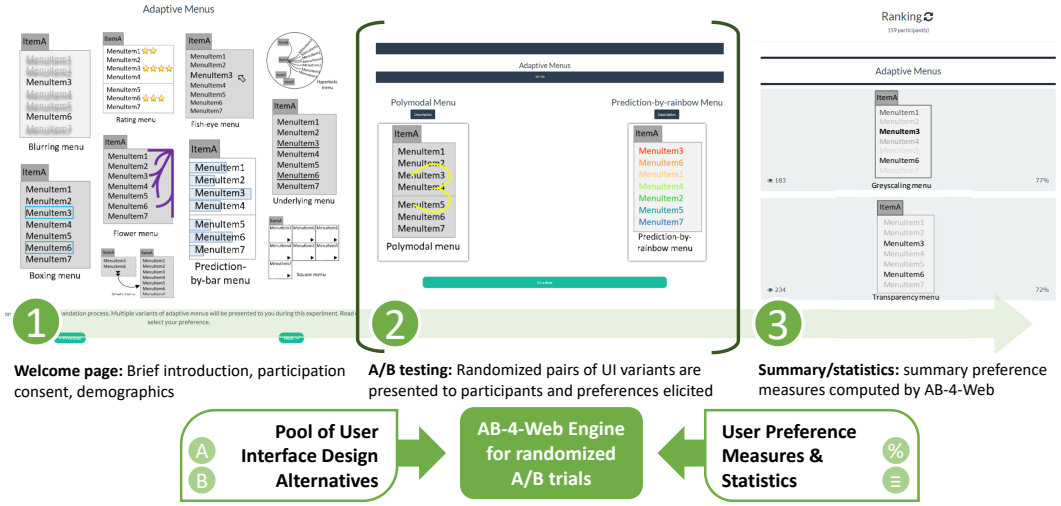


Fig. 7. The AB4Web architecture illustrated with snapshots collected during our study on users' preferences for Graphical Adaptive Menus.

(7) Amazon's crowdsourcing platform Mechanical Turk[7] can involve hundreds of participants to A/B test variants for a multimedia application within a remote virtual machine called HIT [47]. The software leaves the OEC open to define and reports the ratio between positive feedback for a given variant and the number of times that variant was presented.

(8) IPEAD (Ideation, Prioritization, Execution, Analysis, and Documentation) [44] is an A/B testing framework for investigating the impact of organizational changes by applying lean management and gamification. Large-scale A/B tests were conducted in the field of games: variants with different aesthetic improvements were compared together and the effect of each improvement on play time, progress, and the positive criteria rate was examined [2]).

In conclusion, most of the existing software applications implement the simple A/B testing procedure, which is *monovariate* and *bivalued*. Some applications are more flexible by enabling multiple values for the same factor or more than one factor. However, they all implement the traditional version of the A/B testing procedure by repeatedly presenting the variants to participants. In contrast, AB4Web implements a balanced, randomized version of A/B testing that saves considerable time during the study and, consequently, can be applied for studies where the number of potential comparisons of A/B variants is prohibitively large.

## 3 AB4WEB

We implemented AB4Web as an AngularJS[8] front-end in HTML5 with JavaScript and CSS3 code coupled to a FireBase[9] backend. Data communication is ensured by FireBase, running as a NoSQL database management system interfaced as a web service with REST (Representational State Transfer) calls. The FireBase Storage and Hosting are then exploited to make the database persistent and accessible through a console when there is a need to retrieve data directly. For example, if other database queries are needed, such as for specific calculations, the FireBase database can be exported into a JSON format and converted to CSV or XLS format thanks to any on-line converter[10].

[7] https://www.mturk.com
[8] https://angularjs.org
[9] https://firebase.google.com
[10] For example, see http://www.convertcsv.com/json-to-csv.htm.

### 3.1 System Walkthrough and Procedure

Before the experiment, the designer prepares the UI variants as images (*e.g.*, PNG files), animations (*e.g.*, animated GIF files), videos (*e.g.* MP4 files), or links to an external third-party application (*e.g.*, a prototyping tool). Regarding the latter, any sketching software or prototyping application can be used to export UI prototypes or to generate a direct URL to those prototypes. For example, the Gambit platform [40] for collaborative UI design by sketching produces screenshots, web pages, and sketches, which can then be linked from AB4Web. The artifacts can cover monovariate A/B testing, but also multivariate, bi-valued, and multivalued. In the monovariate multivalued case, the conditions should present variations of a single factor. In the multivariate multivalued case, the conditions could present variations of multiple factors, each of them with two or more values as indicated on the bottom row of Table 1.

All variants are then uploaded into a *pool*, materialized by a web directory created for each experiment. The designer specifies a configuration file containing the starting and ending dates of the study, the textual and/or graphical instructions for the participants, the text for the consent form, the number of comparisons per participant, and the demographic data to be captured. If the pool contains $n$ variants, the maximum amount of pairs is $\frac{n \cdot (n-1)}{2}$. Our randomized version does not require to present this full set of pairs, but rather a reasonable subset of $P \leq \frac{n \cdot (n-1)}{2}$. For example, an exhaustive A/B testing procedure for a pool containing 50 variants would require 1,225 comparisons. In contrast, our randomized version is shortened with $P = 50$ comparisons, provided that the number of participants is sufficient to cover all the possible comparisons repeatedly. To ensure the equal proportion of pairs to compare, AB4Web implements a balanced randomized procedure. Each time pair $(A, B)$ or $(B, A)$ is generated, a counter is updated to make sure the maximum number of presentations per pair is not exceeded. This way, the number of presentations of each pair is upper bounded.

AB4Web exposes two URLs: one for conducting the study which is sent to participants and one for access to the results. Once participants receive the URL, the study starts: (1) a welcome page is presented with instructions; (2) the the pairs of variants are presented and the participant selects one of the variants by applying the OEC criteria or selects "It's a draw" if undecided; (3) an ending page is displayed showing the participant their results.

### 3.2 Output Measures

AB4Web collects, stores, and uses participants' responses to compute preference measures. Each time a participant clicks on a variant or on the "It's a draw" push button, AB4Web records these selections and computes the following output measures:

(1) **The Number of Presentations (Num-Presentations)** represents the total amount of times that a particular UI variant was included in an A/B test and, thus, presented to participants. Considering the aforementioned example about the search box design with $6 \times 5$ variants and a total possible number of 435 A/B pairs, if the variant "search box at middle location in the browser with text only elements" is randomly selected by AB4Web to be presented as part of 37 distinct A/B-type pairs, then Num-Presentations is simply 37.

(2) **The Preference Percentage (Preference)** represents the ratio between the number of times that a particular UI variant was marked as preferred by participants during A/B-testing and Num-Presentations. In the previous example, if the "middle location, text only" search box was preferred by participants for 23 times over other variants, then Preference is $23/37 = 62.2\%$. Inversely, the dislike percentage is $(37 - 23)/37 = 37.8\%$.

(3) **The Latent Score of Preference (Latent-Preference)** is based on the calculation model of Bradley-Terry-Luce (BTL) [16], *i.e.*, participants' responses are entered into a preference vector with values +1 (preferred), −1 (not preferred), and 0 (undecided), from which the probability that a specific variant is preferred to others is computed as follows:

$$P_{\text{BTL}}(A \text{ preferred over } B) = \frac{p_A}{p_A + p_B} \tag{1}$$

where $p_A$ and $p_B$ are positive, real-valued scores assigned to variants A and B. Let's assume that option A was preferred 20 times and disliked 11 times compared to option B, while in 6 cases participants voted for a draw. Then, Latent-Preference is $20 - 11 = 9$ and $P_{\text{BTL}} = 9/37 = 24.3\%$.

(4) **The Matrix of Preferences (Preference-Matrix)** computes for each A/B pair a normalized score between $-b$ and $+b$, where $b$ is a bounding real number indicating the relative preference of variant A over B. When $b = N$, the number of responses, the values are normalized in $-1..1$. The matrix offers the practitioner an overall perspective of *what* was preferred and by *how much*, especially when it is visualized using color coding schemes.

## 4 USER STUDY ON GRAPHICAL ADAPTIVE MENUS

### 4.1 Motivations

In order to illustrate the functioning of AB4Web, we conducted an experiment for collecting end users' preferences for Graphical Adaptive Menus (GAMs), which are graphical menus subject to adaptation depending on different parameters such as selection history, user profile, etc. Despite the fact that GAMs have received extensive research [21, 25, 26, 37], user preference over their visual design has remained little evaluated in studies that focused mostly on obtrusiveness [37], predictability [26], subjective satisfaction [37], awareness [21], or disruption [30]. Comparing the performance of the many GAM alternatives proposed in the literature is virtually impossible: their implementation requires a substantive development effort and identifying one common quantitative OEC is illusory. Item selection time could serve to quantitatively compare GAMs of interest, but instrumenting code for properly capturing and computing this variable for all the GAMs would require an enormous development effort. Moreover, the deployment of the experiment would involve consequent resources that would not necessarily lead to conclusive results.

In contrast, understanding user preferences for these alternatives is doable and readily achievable using AB4Web since preference can be assessed as a common OEC for all of them, independently of their implementation.

A GAM should identify the best adaptation mechanism depending on the context of use, which means to automatically display menu items of immediate predicted use to the user while optimizing several variables: maximize perception of predicted items, maximize the ease of understanding, minimize navigation operations and selection time, maximize retention over time, etc. Since these variables are often conflicting and posing constraints, optimizing them all simultaneously is virtually impossible. Similarly, conducting a series of experiments to determine the best adaptation mechanism would require a substantive amount of resources without being sure to reach conclusive results. Interpreting menu selection as an optimization problem enables at least to consider a sub-set of constraints and optimize a sub-set of these variables, like in multi-criteria optimization. But the risk is real to obtain an under-constrained problem where too many variants are identified or to obtain an over-constrained situation where no variant finally emerges. Conducting a series of experiments by controlling these variables would lead to overwhelming complexity. In particular, determining one single variable that would be transversely and consistently used for comparing GAMs remains illusory. Therefore, we decided to adopt a randomized A/B testing.

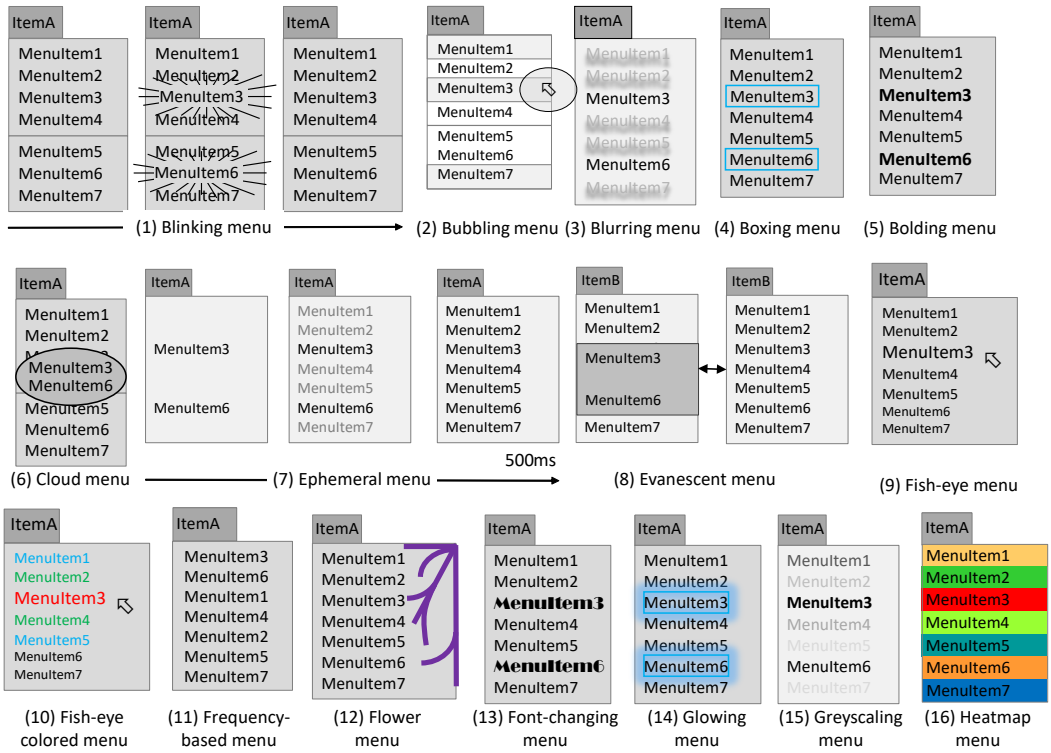


Fig. 8. Catalog of Graphical Adaptive Menus: part 1/4.

## 4.2 Menu Types

A forward and backward snowballing procedure was applied to the pioneer reference [41], which resulted into 122 references[11], which were browsed to come up with a pool of 49 unique GAMs. We consistently describe these GAMS (Figs. 8,9,10,11) in alphabetic order (for easy further reference) according to three criteria: a sentence describing the adaptation mechanism along with its original reference, a description of the selection mechanism, and the Bertin's visual variables that are affected based on [11]: position, size, shape, value, color, orientation, texture, and motion. When such a visual variable is affected, it will be denoted as `Variable+`. We refer to their respective references for details.

(1) **Blinking Menu** [36]: frequent items are blinking on and off for two times in a row (`Value+`, `Motion+`).
(2) **Bubbling Menu** [45]: frequent items are made more easily accessible by a bubbling cursor. This menu accelerates the selection of the frequently used items by directly jumping to them one by one by combining two techniques: the bubble cursor, whose size dynamically changes as the cursor moves and selects the target within the closest distance, and directional mouse-gesture techniques, which accelerate reaching predicted items (`Position+`,`Value+`).
(3) **Blurring Menu**[12]: infrequent items are progressively blurred (`Value+`).
(4) **Boxing Menu**: frequent items are displayed surrounded by a box (`Shape+`).
(5) **Bolding Menu** [37]: predicted items, such as frequent items, are boldfaced (`Value+`).

[11]Only 2D GAMs were considered, not 3D menus such as those used in virtual or augmented reality.
[12]See https://tympanus.net/codrops/2011/10/19/blur-menu-with-css3-transitions/.

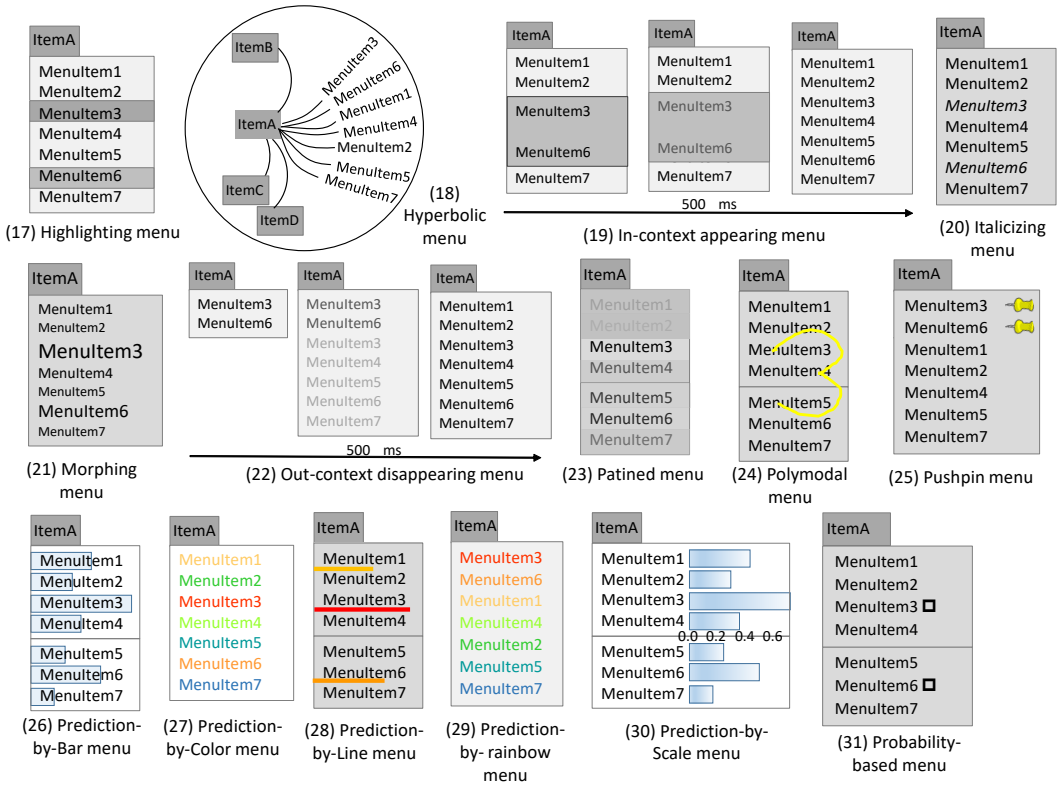


Fig. 9. Catalog of Graphical Adaptive Menus: part 2/4.

(6) **Cloud Menu** [46]: predicted menu items are arranged in a circular tag cloud superimposed to the full menu with a location consistent with their corresponding position in this menu and a font size determined by their prediction level (`Position+`, `Size+`, `Shape+`).
(7) **Ephemeral Menu** [23]: at opening the menu, user finds predicted items and after a delay of 500ms remaining items appear gradually (`Value+`, `Motion+`).
(8) **Evanescent Menu** [15]: a prediction window containing the predicted items is superimposed over the menu and progressively made transparent to reveal all the items, thus enabling the user to select a predicted item if it belongs to the prediction window and any other item from the full menu after (`Position+`, `Shape+`, `Motion+`).
(9) **Fish-Eye Menu** [8]: the font size of frequent items is increased and items are grouped. Initially, this menu displays items with a font size that increases or decreases depending on the distance with respect to cursor position: the closer, the larger, the further, the smaller. They could be made adaptive by assigning the frequency to the font size, as in morphing menus, but also have frequent items closer to the cursor position (`Position+`, `Size+`).
(10) **Fish-Eye Colored Menu**: frequent items are increased in text size, colored, and grouped (`Position+`, `Size+`, `Color+`).
(11) **Flower Menu** [5]: frequent items are arranged according to a layout where items are accessed by flower gestures (`Position+`).
(12) **Font-Changing Menu**: frequent items are displayed using salient fonts (`Value+`).
(13) **Frequency-based Menu**, also called **Dynamic Menus** [35]: items are sorted in decreasing order of their frequency of usage (`Position+`). A variant of this menu is the family of

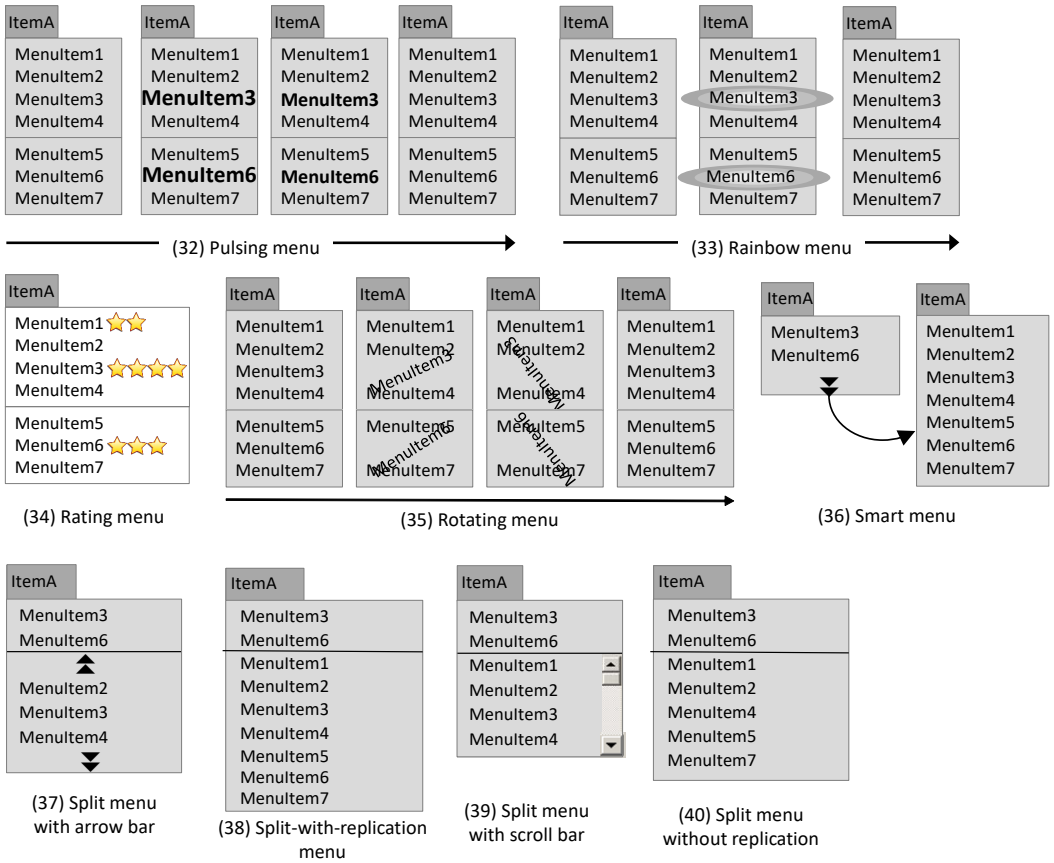


Fig. 10. Catalog of Graphical Adaptive Menus: part 3/4.

probability-based menus, where item are sorted in decreasing order of their probability of usage, which could be computed as the ratio of item selection per unit of time [6].

(14) **Glowing Menu** [7]: frequent items are glowed progressively (`Value+,Texture+`).
(15) **Greyscaling Menu**: an adaptive menu for which infrequent items are greyscaled (`Value+`).
(16) **Heatmap Menu**: item frequency is depicted using a heatmap color coding (`Color+, Texture+`).
(17) **Highlighting Menu** [37]: frequent items are emphasized by contrasting them with respect to normal items appearing in the menu (`Value+,Texture+`). For instance, Gajos et al. [25] highlight predicted items by colouring their background in pink.
(18) **Hyperbolic Menu** [32]: frequent items are arranged in related hyperbolic (sub-)trees. This menu consists of a "focus + context" technique for displaying and manipulating large hierarchies with parts of the hyperbolic view that are expanding and collapsing depending on the frequency (`Position+, Shape+, Orientation+`).
(19) **In-Context Appearing Menu** [14]: at opening the menu, a prediction window prompts three predicted items and disappears gradually to leave the room for the full menu (`Position+, Shape+, Motion+`).
(20) **Italicizing Menu**: frequent items are displayed using italic fonts (`Value+`).
(21) **Morphing Menu** [18]: the font size of each menu item is adapted depending on its prediction: the higher the prediction is, the larger the font size becomes, the lower the prediction is, the smaller the font size becomes (`Position+,Size+`).

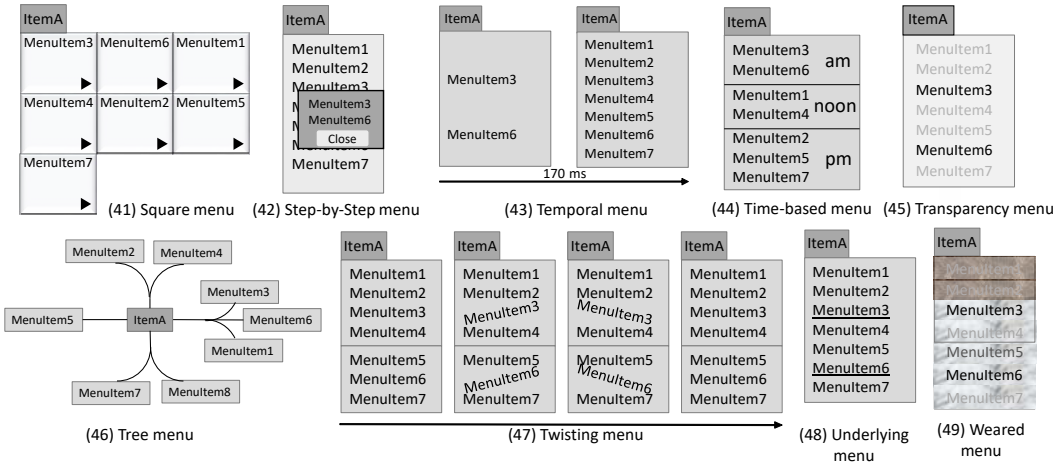


Fig. 11. Catalog of Graphical Adaptive Menus: part 4/4.

(22) **Out-Context Disappearing Menu** [14]: at opening menu, a prediction window is immediately displayed with predicted items; after 500 msec [23], the complete menu is gradually displayed from the back, replacing the prediction window (`Position+, Shape+, Motion+`).
(23) **Patined Menu** [34]: items are patined (`Texture+`).
(24) **Polymodal Menu** [13]: any menu item can be selected graphically (by pointing), vocally (by voice recognition), tactilely (by touching), gesturally (by issuing a gesture representing the menu), or any combination of them. Predicted menus are rendered in graphical or vocal prediction window (`Position+`).
(25) **Prediction-by-Bar Menu**: item frequency is depicted with a transparent horizontal bar (`Shape+, Value+`).
(26) **Prediction-by-Color Menu**: item frequency is rendered according to a color coding scheme (`Value+, Color+`).
(27) **Prediction-by-Line menu** [6]: item frequency is depicted by an underlining bar (`Shape+, Color+`).
(28) **Prediction-by-Rainbow Menu**: item frequency is shown using the rainbow color coding scheme (`Color+, Texture+`).
(29) **Prediction-by-Scale Menu**: item frequency is depicted by an horizontal histogram (`Shape+, Value+, Texture+`).
(30) **Probability-based Menu**: frequent items are marked by partially–filled squares (`Position+, Shape+`).
(31) **Pulsing Menu**: items selected more frequently are pulsating back and forth (`Position+, Size+, Motion+`).
(32) **Pushpin Menu** [29]: a pushpin retains the frequently used menus on the menu to make them salient and unmovable. When the menu button is selected by the user, a menu appears in a rectangular box containing a pushpin and several menu items. If the user clicks on the pushpin then the temporary menu box is converted into a permanent window which remains on the display regardless of other display operations. The user may click again on the pushpin to release the permanent placement of the menu on the screen (`Shape+, Color+`).
(33) **Rainbow Menu**: item frequency is depicted with a rainbow color coding (`Color+, Texture+`).
(34) **Rating Menu**: items used more frequently are marked with a number of stars (`Shape+`).
(35) **Rotating Menu**: frequent items are rotated two times (`Position+, Orientation+, Motion+`).

(36) **Smart Menu** [3]: based upon the usage pattern of the user, only the most commonly used items are displayed in a short menu mode, which can be expanded to long menu mode to display all items. The most frequent items are added to the short menu and the least frequent items are removed from it (`Position+, Size+, Shape+`).
(37) **Split-with-Arrowbar Menu** [8]: frequent items are duplicated into a box added on top of the initial menu transformed into a list with arrows at both ends (`Position+, Size+, Shape+`). This menu is suggested when the amount of items in the initial menu is large.
(38) **Split-with-Replication Menu** [25]: frequent items are duplicated into a box added on top of the initial menu (`Size+, Shape+`).
(39) **Split-with-Scrollbar Menu** [8]: frequent items are duplicated into a box added on top of the initial menu transformed into a fixed list box (`Position+, Size+, Shape+`).
(40) **Split-without-Replication Menu** [22, 41]: frequent, recent, or interest-based items are moved into a topmost box on top of the initial menu (`Position+, Size+`).
(41) **Square Menu** [1]: initially, items of the pull-down menus are reformatted into square regions in order to improve their selection performance. To make it adaptive, frequent items are displayed in a square proportional to frequency of usage (`Position+, Size+, Shape+`).
(42) **Step-by-Step Menu** [12]: displays at each level of the menu hierarchy a prediction window containing predicted items and offers to select the most likely menu item leading to the next level of the target path (`Position+, Size+, Shape+`).
(43) **Temporal Menu** [33]: at opening, the menu displays only frequent items and, after a delay of 170 ms, the rest of items which are assumed to be less frequent (`Position+, Motion+`).
(44) **Time-based menu** [3]: items are marked with a time value depending on their frequency during the day (`Position+, Shape+`).
(45) **Transparency Menu**: infrequently selected items are made more transparent (`Value+`).
(46) **Tree Menu**: frequent items are arranged in the form of a marking menu, *e.g.* in a clockwise manner in decreasing order of their frequency (`Position, Size+, Shape+`).
(47) **Twisting Menu** [39]: predicted items are emphasized by twisting them a limited amount of times (`Shape+, Orientation+, Motion+`).
(48) **Underlining Menu**: items used frequently have the text underlined (`Value+`).
(49) **Weared Menu**: item frequency and recency are rendered according to a wearing scheme (`Value+, Texture+`).

*4.2.1 Participants.* A total number of 163 participants were recruited using the authors' institutions mailing lists. Participants had various nationalities, originating from sixteen countries, and speaking eight languages, in alphabetical order: Arabian, Dutch, English, French, German, Portuguese, Romanian, and Spanish. The study took place online via AB4Web with no compensation offered. We removed 55 outliers for different reasons: the experiment was left unfinished (hence, data were incomplete), the same response was entered repeatedly from one pair to another (*e.g.* always the right variant or always "It's a draw"), the user selected the 'back' button in the browser (thus provoking the same pair to appear twice) or because too rapid decision making alerted by our system. The final data collected from our study is represented by 108 (participants: 36 female, 78 male) × 50 (randomized trials) = 5,400 responses over GAM pairs.

*4.2.2 Representativeness of participants.* Fig. 12 illustrates the age-gender demographic distribution of our participants as an age pyramid. Overall, participants' age varied between 22 and 73 years ($M$ = 39.19, $SD$ = 12.10 years). A acceptable age coverage for both gender groups: female participants between 25 and 73 years old and males between 22 and 73 years old. The age distributions were not normal (an observation confirmed by Shapiro-Wilk tests, $W$=.862, $p$<.001 for female and $W$=.948, $p$<.001 for male participants, respectively) with a higher representativeness for male and young

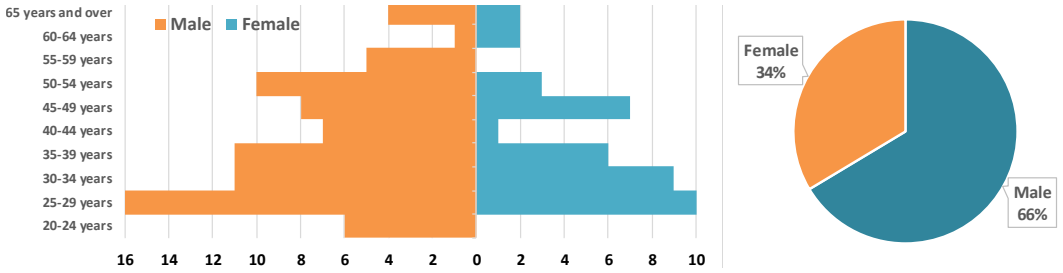


Fig. 12. Age pyramid of participants and distribution by gender.

people, around 35 years old (the mode and the median of the sample both equal 35), in our sample. Young people are expected to be more open and eager to use technologies, whereas older adults are expected to rely on common technologies. From this perspective, our sample fits the goal of our experiment, which is focused on understanding end users' preferences for GAMs. Moreover, the mean ages were close for the two gender groups ($M$=38.42, $Mdn$=34.50 years for female and $M$=39.38, $Mdn$=37.00 years for male participants, respectively, Wilcoxon's rank sum exact test $W$=1708, $p$=.226, $n.s.$) and the age distributions were significantly different (as indicated by a Kolmogorov-Smirnov test $D$=.111 < $CritD$ = .131, $p$=.018).

*4.2.3 Apparatus.* The study was conducted entirely online via AB4Web, and the URL[13] was communicated to participants by email. The measures are computed in real-time and accessible on-demand at the end of the experiment[14]. Vectorial images were produced for each GAM and converted to high-resolution GIF images. For GAMs with visual effects, animated GIFs were produced according to the temporal guidelines of the respective GAM, *e.g.*, 170 ms for the Temporal Menu [33], 500 ms for the Ephemeral Menu [23], etc. The animation was repeated three times in a row. Images were uploaded to AB4Web along with a short text description of each menu type.

*4.2.4 Procedure.* AB4Web generated 50 GAM pairs at random for each participant, presented them in a random order, and asked participants to select the variant they preferred by relying solely on visual aesthetics. The GAM selected from each A/B-type comparison accumulated one point, while the alternative was deducted one point. If participants were undecided, AB4Web offered the option "It is a draw," in which case no points were assigned to either variant. Since our catalog comprised $n$=49 GAMs, a complete A/B testing would require $n \times (n-1)/2 = 1,176$ pairs of variants to be presented to each participant, a prohibitive study to complete. Thus, AB4Web randomly generated a smaller number of 50 pairs (no duplicates) for each participant. The A/B pairs were randomized in terms of their order of presentation, and variants were randomized in terms of their order within each pair. No time constraint was imposed. The study lasted on average 15 minutes.

*4.2.5 Results and Discussion for the Preference Percentage.* Fig. 13 reproduces the ranking of the 49 GAMs in decreasing order of their averaged Preference, from 81% for the Grayscaling Menu to just 8% for the Rotating Menu. In this figure, GAMs are clustered into four groups depending on their Preference: *high preference* above 50% (ranging from the Greyscaling menu to the Frequency-based menu), *medium preference* between 30% and 50% (ranging from Prediction-by-line Menu to Heatmap Menu), *low preference* between 20% and 30% (ranging from polymodal menu to hyperbolic menu), and *very low preference* below 20%.

[13] https://mathieuzen.github.io/adaptive-menus-ranking/
[14] https://mathieuzen.github.io/adaptive-menus-ranking/#/stats

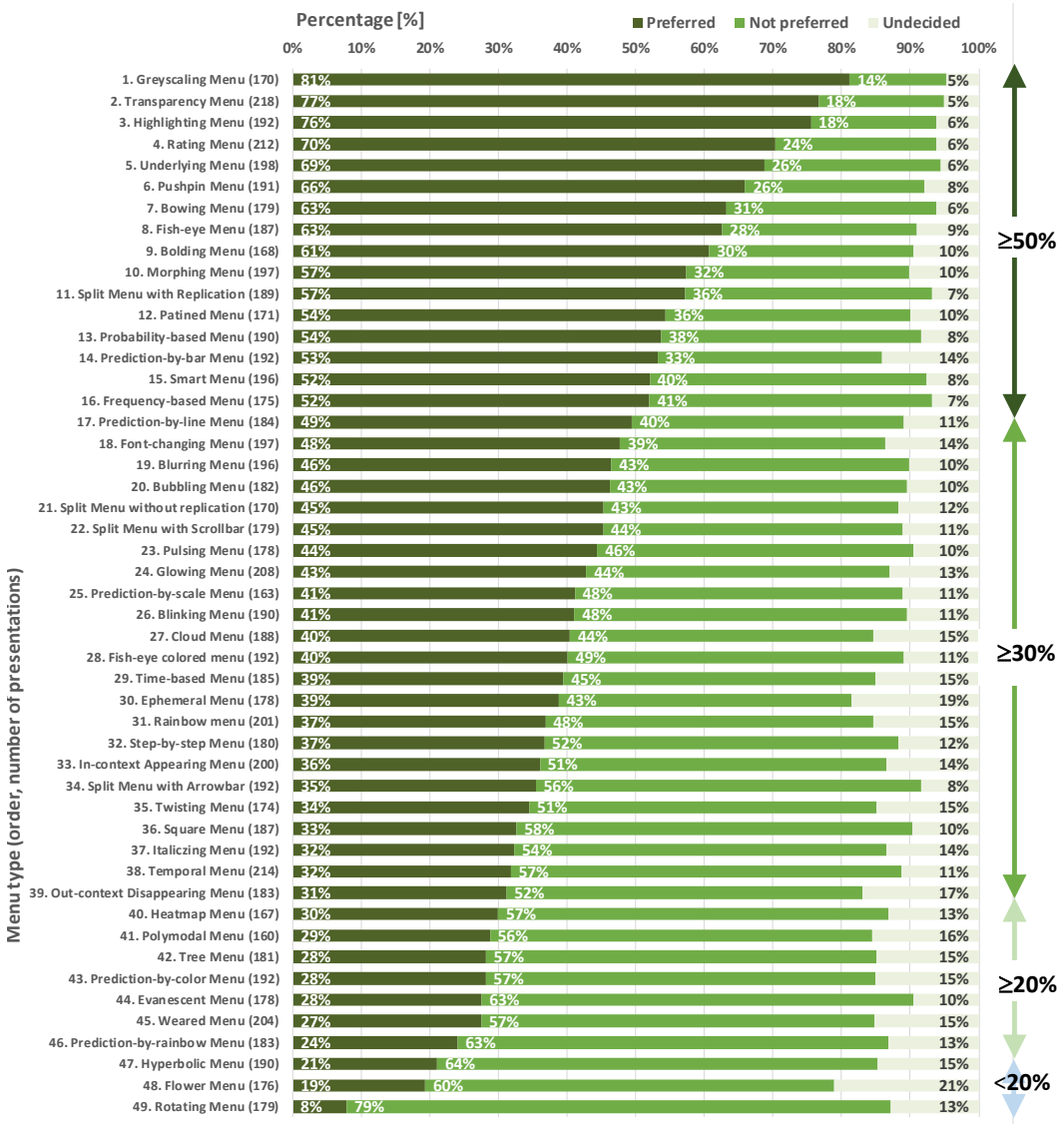


Fig. 13. Graphical Adaptive Menus in decreasing order of user preference. The label indicates the order and the name of the menu type followed by the average Num-Presentations. Each bar indicates respectively the Preference, the percentage of being not preferred, and the percentage of being undecided (figures rounded).

**Positive results.** A first observation reveals that the most preferred menus in general are those menus which only admit some change in the `Value` visual variable: among the first 10 menus in Fig. 13, six are value-changing at pro-eminent places: greyscaling (#1), transparency (#2), highlighting (#3), underlining (#5), boxing (#7), and bolding (#9). This suggests that participants preferred the menus that minimize not only the amount of visual change (only one variable), but also that among possible visual changes, they chose the one that affects the least the visual appearance of the original menu. Greyscaling, the most preferred menu, disrupts the least the overall appearance of the initial menu subject to adaptivity. The rating menu appeared in the 4th

position and affects size and shape only, but again leaves the original menu unaltered; it is also the first menu changing two variables for adaptivity at the same time. The Pushpin (#6) is in the same case, both menus are the only two designs that feature shape change, while the Fish-Eye Menu (#8) and the Morphing Menu (#10) are the only designs with size-changing capability. These results confirm a preference for preserving physical stability in terms of shape and size.

The morphing menu (#10) is the first one that tolerates a position-changing behaviour (`Position+`), along with a change in size. The next one, i.e. the split with replication (#11), affects also the same two variables, i.e. `Position` and `Size`, but represents the most appreciated split menu, probably the only that is acceptable since the prediction window is replicated on top of the menu. All other instances of the split menu come quite later on: split without replication (#21), split with scroll bar (#22), and finally split with arrow bar (#34). None of them are satisfying enough.

The patined menu (#12) made the first foray into the territory of texture-changing menus, long before other designs from the same category, such as the Weared Menu (#45), again probably because the visual change is minimized, even if it is textured. Surprisingly, the smart menu (#15), which has been much reviled [17], is the first menu accommodating with 3 visual variables changed simultaneously. The prediction-by-line (#17) is the first GAM accepting some color change, far before all other color-changing menus, but another member of this family, is still preferred: prediction-by-bar (#14). The frequency-based menu (#16) closes the first cluster with high preference ($\geq 50\%$).

**Negative results.** Among other motion-changing menus, the Pulsing menu (#23) belongs to the second cluster of medium preferences, followed by Glowing Menu (#24), the Blinking Menu (#26), and the Ephemeral Menu (#30), the twisting menu (#35). Other members of this category come long after, such as temporal menus (#38), ICD (#33) and its counterpart OCD (#39), or the evanescent menu (#44).

We also found out that color-changing menus were not appreciated at all in most menu designs, probably because participants could not easily map colors to item frequency, even with the Prediction-by-color (#43), Prediction-by-Rainbow (#46) or the Heatmap Menu (#40). Our intuition is that these color schemes seem more appropriate for the visualization of frequencies, rather than to assist item selection. The Cloud Menu (#27) [46] is the first GAM with a superimposed prediction window, as opposed to a tiled one, as in Split-with-Replication (#11). Menus with unusual shapes, such as the Square Menu (#36) or the Hyperbolic Menu (#47) were not appreciated either, probably because their respective advantages are elsewhere than in their visual design. Two menus belong to the fourth cluster of very low agreement, *i.e.* the flower menu (#48) and the rotating menu (#49), the first one because participants did not succeed to map gestures to the various items, especially after adaptation, the second one because rotating is the most distractive animation.

*4.2.6 Results and Discussion for the Latent Score of Preference.* Fig. 14 reproduces the ranking of the 49 GAMs in decreasing order of their averaged Latent-Preference, from a score of $BTL = 127$ for the Transparency Menu to just $BTL = -128$ for the Rotating Menu. This approach gives a clearer stance on the possible GAMS that could be kept and those that should be avoided: only eighteen GAMS, ranging from the transparency menu to the prediction-by-line menu, are revealed acceptable, with a positive BTL score. Those menus which display a close-to-zero or negative BTL score are meant to be definitely discarded. It is interesting to notice that in this set of 18 admissible GAMS, we find again exactly the same first GAMS which were discovered thanks to the Preference measure. The main difference is that the discrimination criteria appear more clearly through the Latent-Preference than through the Preference variable. Fundamentally, the ranking obtained by BTL does not change the main position and the main clusters: menus appear almost at the same position, sometimes with a slightly changed order. The first and last menus of each cluster always appear in the same position, which confirm the initially obtained results.

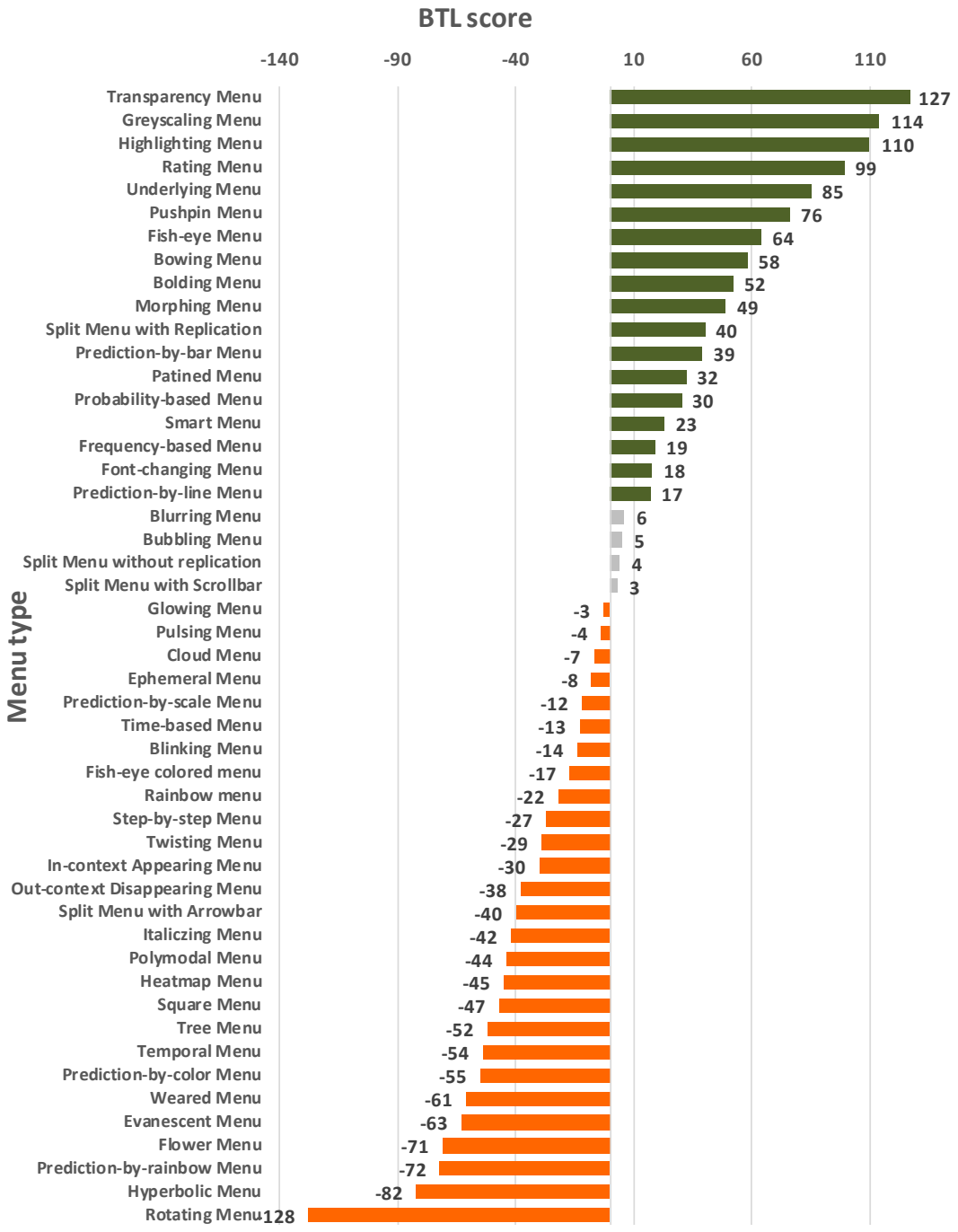


Fig. 14. Graphical Adaptive Menus in decreasing order of their Latent-Preference.

| Graphical Adaptive Menus | Blinking M | Blurring M | Bolding Me | Bowing Me | Bubbling M | Cloud Men | Ephemeral | Evanescent | Fish-eye M | Fish-eye co |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 Blinking Menu | 0 | -1 | -3 | -4 | 2 | -1 | -1 | 3 | -2 | 2 |
| 2 Blurring Menu | 1 | 0 | 0 | -4 | 1 | 1 | 1 | 3 | 1 | 0 |
| 3 Bolding Menu | 3 | 0 | 0 | 1 | 2 | 4 | 0 | 1 | -1 | 5 |
| 4 Bowing Menu | 4 | 4 | -1 | 0 | 1 | 4 | 1 | 3 | 1 | 1 |
| 5 Bubbling Menu | -2 | -1 | -2 | -1 | 0 | 2 | 1 | 2 | -3 | 0 |
| 6 Cloud Menu | 1 | -1 | -4 | -4 | -2 | 0 | 1 | 2 | -3 | 2 |
| 7 Ephemeral Menu | 1 | -1 | 0 | -1 | -1 | -1 | 0 | 1 | -2 | -1 |
| 8 Evanescent Menu | -3 | -3 | -1 | -3 | -2 | -2 | -1 | 0 | 0 | -2 |
| 9 Fish-eye Menu | 2 | -1 | 1 | -1 | 3 | 3 | 2 | 0 | 0 | 3 |
| 10 Fish-eye colored menu | -2 | 0 | -5 | -1 | 0 | -2 | 1 | 2 | -3 | 0 |

Fig. 15. An excerpt of the Normalized Matrix of Preferences.

*4.2.7 Results for the Matrix of Preferences.* Fig. 15 reproduces an excerpt of the matrix of preferences PREFERENCE-MATRIX normalized by the bounding value $b = 8$ after applying a conditional formatting based on this value. The full matrix of preferences is provided in the supplemental resources. This symmetric matrix gives an idea to what extent any particular GAM is preferred or not with respect to another as each cell contains the BTL score normalized between two GAMs of a pair: each cell $m_{ij}$ gives the relative preference of menu $m_i$ over $m_j$, between -8 and +8 in our case.

For example, the first column of Fig. 15 is interpreted as follows: the blinking menu is equivalent to itself (hence, the 0 score), the blurring menu is slightly preferred over the blinking menu (hence, the 1 score), the bolding and the bowing menus are much more preferred over the blinking menu (respectively, with a score of 3 and 4). Conversely, the bubbling menu is judged inferior to the blinking menu (with a negative score of -1), the evanescent menu is estimated quite more inferior to the blinking menu (with a score of -3). The cell $c_{3,10} = 5$ suggests that the bolding menu is much more preferred over the fish-eye menu with colors.

In general terms, $\forall i, j \in 1, \ldots, n : p_{i,j} = c_{i,j}/b$ where $b$ denotes a bounding value, typically here, the amount of pair comparisons (in our case, $P = 50$). This enables us to get the probability that a specific menu is preferred to the others: $m_i$ is preferred to $m_j \Leftrightarrow P(m_i > m_j) = p_{i,j} = \frac{p_{m_i}}{p_{m_i}+p_{m_j}} > P(m_i \leq m_j)$. For instance, the cell $M_{2,1} = 1$ expresses that the menu $M_2$=blurring menu is slightly preferred over the menu $M_1$=blinking menu. The cell $M_{8,1} = -3$ expresses that the menu $M_8$=evanescent menu is largely unpreferred with respect to menu $M_1$=blinking menu. The column #45 containing the results of comparing GAM with respect to the transparent menu is almost red everywhere, meaning that the transparent menu was almost always preferred to other menus compared, except for the greyscaling menu ($M_{15,45} = 5$) and the highlighting menu ($M_{17,45} = 4$), which were ranked before. In conclusion, instead of giving an *absolute* reference as in PREFERENCE, this matrix of preferences provides a *relative* reference where any pair of GAMs is compared. If a designer needs to choose between two GAMS or more before implementation, this matrix provides a clear answer.

### 4.3 Other Output Measures

Beyond the computation of our four measures defined in Section 3, we also wanted to test other measures on the same raw data to confirm or infirm the preferences. There are two related methods for consolidating votes from participants on candidates: the *Condorcet method* [38] and the *de Borda method* [20], the last one being often used when the first one is unable to identify a *Condorcet winner*. Each method may be used to rank candidates (here, rank GAMs) for a selection based on votes (here, based on end users' preferences) and to choose a winner (here, the most preferred GAMs). To choose a winner, the Condorcet method is based on one rule: select the GAM candidate (if it exists) that beats each other GAMs in exhaustive pairwise comparison. The de Borda method is based on a similar rule: select the GAM that on average stands highest in the participants' preferences. To rank the GAMs, the Condorcet method applies the rule: rank the GAMs in descending order

of their number of victories in exhaustive pairwise comparison with all the other GAMs. The de Borda method ranks the GAMs in descending order of their standing in the participants' rankings. The lowest score is commonly assigned to the least preferred candidate and the score increases with the ranking. Based on these rules, we define the following additional measures:

(1) **The de Borda score starting at 1 (BordaOne)** [20] represents the ratio between the double amount of times a particular UI variant was preferred plus the amount of times it has been assessed as undecided or unpreferred and Num-Presentations. Considering the aforementioned example about the search box design with $6 \times 5$ variants and a total possible number of 435 A/B pairs, if the variant "search box at middle location in the browser with text only elements" was presented 37 times, among which preferred for 23 times, then undecided for 10 times, then unpreferred for 7 times, then BordaOne is $(2 \times 23 + 10 + 7)/37 = 1.70$.

(2) **The de Borda score starting at 0 (BordaZero)** [20] represents the ratio between the amount of times a particular UI variant was preferred and Num-Presentations. Considering the same example again, BordaZero is $23/37 = .62$.

(3) **The Dowdall score (Dowdall)** [24] represents the ratio between the amount of times a particular UI variant was preferred plus half the amount of times it has been assessed as undecided plus one third of the mount of unpreferred and Num-Presentations. In the same example, Dowdall is $(23 + 10/2 + 7/3)/37 = .82$.

Fig. 16 graphically depicts the two first counting measures, respectively BordaOne and BordaZero in decreasing order of their value. Fig. 17 does the same job for the Dowdall measure. Before analyzing these results, we must first acknowledge that these measures are not primarily intended to express preferences, but to identify the $n$-best candidates in a pool of $m >= n$, most of the time only the best candidate should be identified (the winner takes it all).

The de Borda method starting at 1 or 0 as well as the Dowdall keep five frontrunners already found in the previous measures: the Transparency menu (#2 in Latent-Preference) which is now the winner in all categories (#1 in all counting methods, followed by the Rating menu (respectively, #4, #2, #2, and #2), the Pushpin menu (respectively #6, #3, #4, #3), the Highlighting menu (#3, #4, #3, and #4), and the Underlining menu (respectively, #5, #9, #6, #6). Surprisingly, the Greyscaling menu, which was selected as the most preferred menu in our previous measure, falls to the #16 in the BordaOne, #5 in the BordaZero, and #9 in the Dowdall measures. This may be explained by the fact that these measures considers all percentages, including the number of times for unpreferred and undecided, but with a lower coefficient, thus resulting into Transparency menu as first winner because its value is the largest in terms of preference and rejection combined, all measures considered. The ordering and the proportion above the average for these measures remain more or less the same since they compute a linear combination of Preference. The Dowdall counting exhibits a distribution analogous to BordaOne for the same reasons, apart the fact that the amount of undecided and unpreferred count for less importance in the linear combination.

## 5 BENEFITS AND LIMITATIONS

In this section, we report on observed benefits of the balanced randomized A/B testing as a method, the potential benefits of its implementation into AB4Web, and their corresponding limitations. In this way, a shortcoming that is intrinsic to the method will be obviously propagated to AB4Web, but the way the method is implemented in AB4Web could also lead to particular limitations. We therefore denote by $B$, respectively by $L$ a benefit, respectively a limitation and we add a suffix to denote whether it is applicable to the method ($M$) or the tool implementing it ($T$). For example, a tool limitation would be reported as $LT$, while a method benefit would be reported as $BM$.

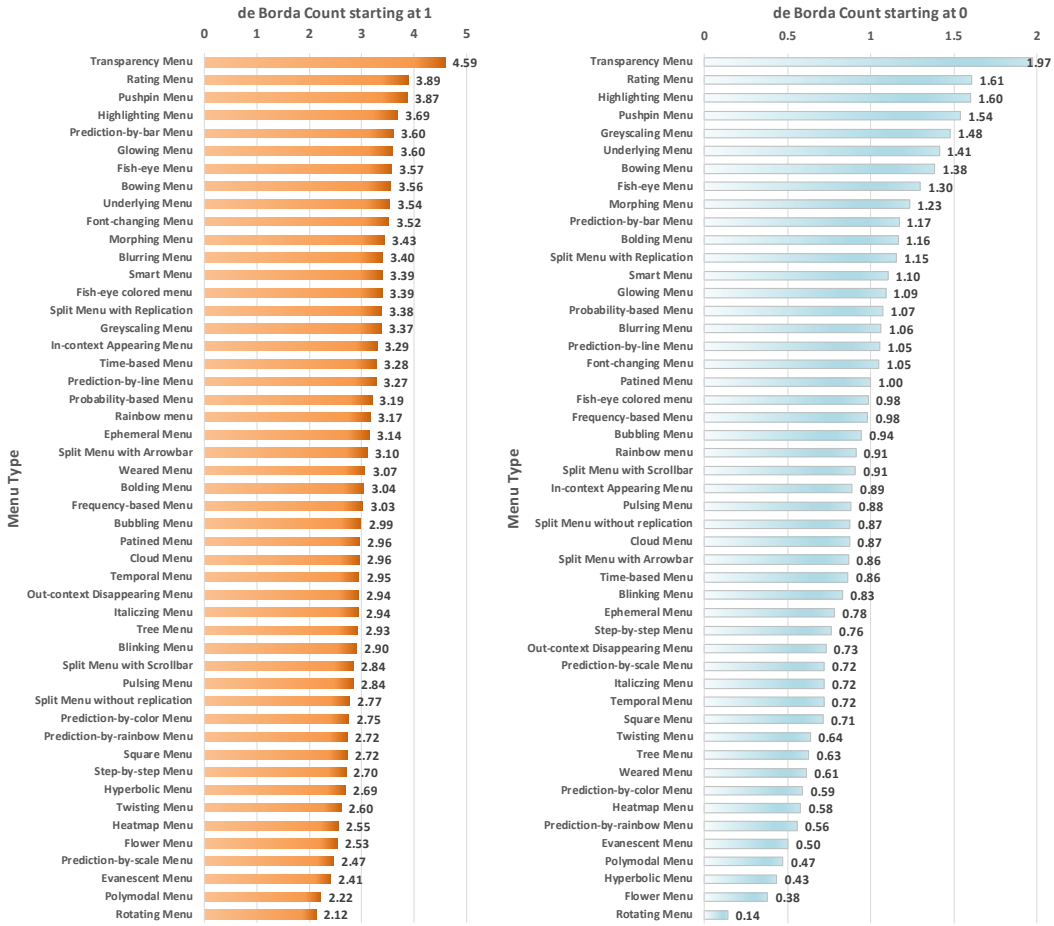


Fig. 16. Graphical Adaptive Menus in decreasing order of their count computed by the de Borda method starting at 1 and 0.

- **BM1: Low cost operationalization.** The total cost of conducting our balanced randomized A/B testing equals the cost for creating, storing the UI design alternatives plus the cost of experiment configuration and deployment. Therefore, the operationalization cost is considered low compared to other methods.
- **BM2: Flexibility of UI variants**. The UI variants could be static or dynamic, thus leaving ample flexibility to the designer to test real UI variants vs restricted versions (called represented UI) with real vs represented users. A new UI variant can be added at any point in time. When the experiment was launched, we stopped examining the existing literature about GAMs to keep the pool constant. In the meanwhile, we identified new GAMs which could be incorporated in AB4Web simply by uploading their representation into the pool, without requesting any other change in the software. AB4Web will then automatically consider new menus as any other instance for comparing pairs. Nevertheless, some time will be needed to reach a GAM distribution that is again balanced: if the Num-Presentations was around 187 until now in average ($SD$ = 13), there will be some time needed for the new menus to reach the same average, which is ensured by AB4Web.

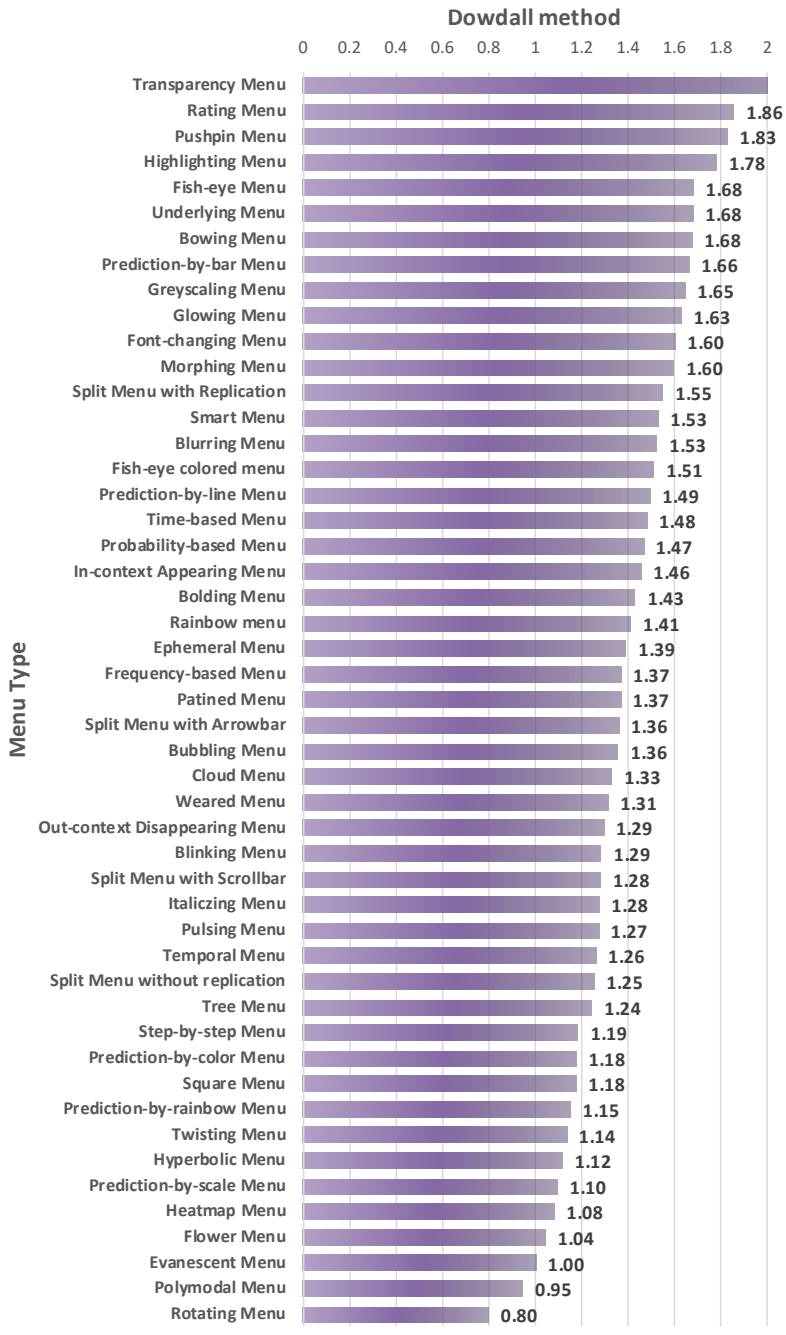


Fig. 17. Graphical Adaptive Menus in decreasing order of their count computed by the Dowdall method.

- **BM3: Continuity of experiment.** Any experiment can be launched and stopped at any time or left open to maintain the results evolving over time. AB4Web randomly selects UI variant based on www.random.org, which itself is a real-random service based on weather temperatures. Although it is not a pseudo-random generator, it does not ensure the perfect equal distribution about all UI variants (see PREFERENCE in parentheses in Fig. 13).

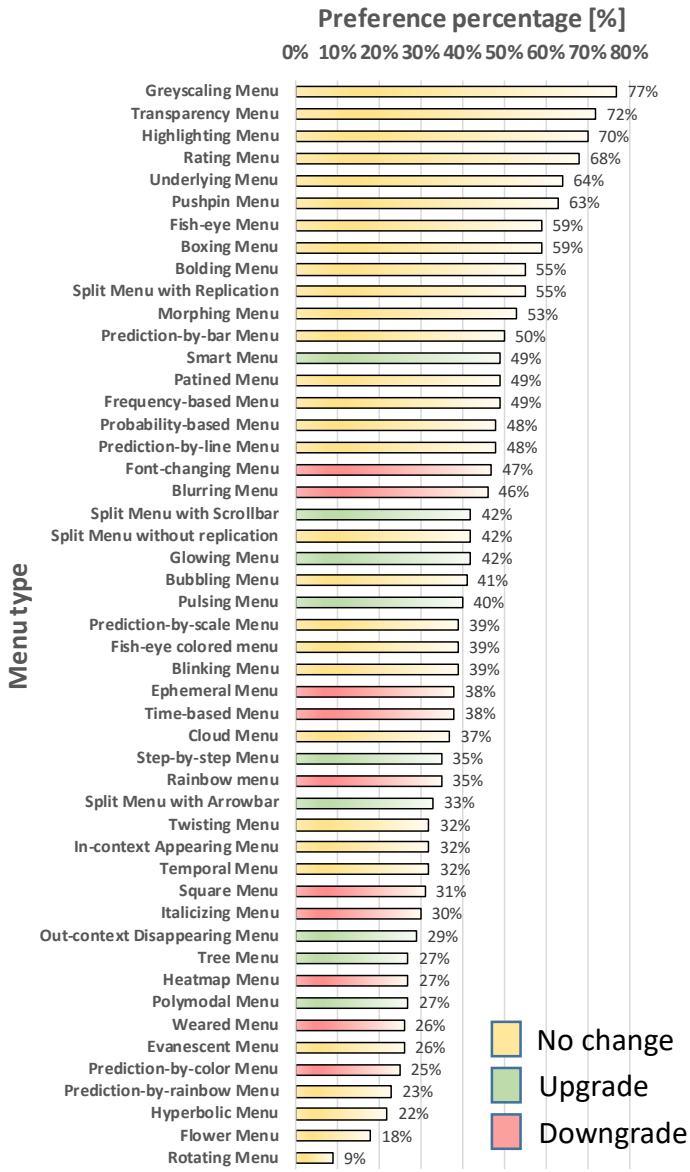


Fig. 18. Evolution of Preference between two points in time.

- **BM4: Comparison over time.** Multiple versions could be compared between two points in time or more, which is handy for analyzing trends in real time. Fig. 18 colors bars depending on the change of ranking between two snapshots that we took over time: one while the experiment was running (87 participants with outliers) until the second snapshot (163 participants reduced to 108 after outlier removal). Only 9 menus have seen their ranking based on preference being increased (in green), 10 menus have been depreciated (in red), and the remaining 30 ones remain in the same interval (in yellow). This reveals a relative convergence of results, even after a reasonable amount of participants. A further analysis would be welcome to determine the population sampling size beyond which a certain convergence

would be guaranteed. This raises the question of determining when the audience size is large enough to be representative for inputs. Similarly, the question of how long the experiment should be in order to produce reliable conclusion is equally important.

- **BT1: No technical expertise required.** Contrarily to many software for A/B testing used in marketing, eLearning (as discussed in Section 2), AB4Web does not require any technical expertise to be conducted, thus making it an affordable tool in the hands of designers. The A/B testing according to our randomized procedure is completely automated when launched. Anonymized data are automatically linked to any UI variant. At any time, it is possible to access raw data by querying the FireBase database, but this is reserved for advanced users.
- **BT2: Free format for UI variants.** No constraint is imposed to the building and representation of UI variants: the variant could be a real UI (e.g., a a running one) or a represented UI (*e.g.*, a screen shot, an image, a wireframe, a prototype) for real users or representative users. Once could imagine to couple AB4Web to any UI generator so that (semi-)automatically generated user interfaces could feed the pool of UI variants in a more efficient way.
- **LT1: Fixed presentation policy.** All GAMs uploaded into the pool are considered equiprobable (same probability for all variants), they are always presented by pairs and compared according to one single OEC. This presentation policy could be made more flexible by parameterizing the presentation policy: present UI variant according to their own probability (instead of all same), augment the amount of UI variants per comparison (not too much, though, probably not more than 3-4), support 1 against $n$ UI variants or $m < n$ at the same time. Such a presentation policy could dynamically change the presentation rules (*e.g.* based on production rules), such as changing the presentation so that the best or the worst UI variant is presented more or less often as data re collected. This requires discovering how to personalize the presentation policy and how these rules could be implemented. Two approaches are particularly appropriate: "Keep-the-Best", where the preferred UI variant would be included in the immediate next round and compared against a new one, and the "Tournament", where the selected UI variant would be added to a winners pool and would resurface in future comparisons when all other variants have been compared.
- **LT2: Single Overall Evaluation Criteria.** In the current version of AB4Web, only one OEC is assumed (in our experiment, preference). Instead of relying on only one such criteria, a multi-criteria approach could be offered, again provided that the amount of criteria does not reduce the easiness of the current approach. If a pair of two variants is presented for example, AB4Web could accommodate additional widgets to capture a rating or a ranking, *e.g.*, via a rating bar, a slider, a drag and drop list. Equally important is also how to apply the OEC. If several UI variants are proposed, a ranking should be offered to let participants rank the variants according to each criteria.
- **LT3: Four implemented measures.** AB4Web implements the four output measures defined in Section 3.2, but could implement the other ones, *e.g.* the Condorcet, the de Borda or the Dowdall measures as well. This wold be particularly meaningful when the designer would like to choose different measure depending on the desired goal: find the best UI variant according to one or multiple criteria, find the $p$-best candidates among a pool of $n$ UI variants, find the $p$-best candidates above a certain threshold, etc.
- **LT4: Segmentation and targeting.** A/B testing usually considers that all UI variants are equiprobable (*e.g.* with equal probability delivered to all participants). However, in some circumstances, responses to UI variants may depend on some variables, like age, gender, location or any other profile parameter. For instance, while UI variant A receives the highest Preference, UI variant B may receive a higher value within a specific segment of the sample.

## 6 CONCLUSION

We presented AB4Web, a web-based engine that implements an on-line balanced randomized version of the multivariate A/B testing, enabling practitioners to rapidly conduct an experiment. AB4Web is particularly handy for large design spaces, where the standard A/B testing is prohibitive because of the large amount of factors, values, and possible pairs of comparison. This feature represents a significant advantage for participants as the complexity of the experiment can be managed by the practitioner. We illustrated AB4Web for collecting preferences about the visual design of Graphical Adaptive Menus, for which we reported experimental results. Beyond the release of the AB4Web tool in the public domain, future work will investigate how to implement a more flexible presentation policy and multi-variate testing. Also, we plan further evaluations of our tool by applying it to other application areas, such as collecting users' preferences for the visual aesthetics of menus displayed on smartglasses, Augmented Reality applications, or user preferences for the visual appearance of the shapes of stroke-gesture commands.

## ACKNOWLEDGMENTS

R.-D. Vatavu acknowledges support from the project PN-III-P1-1.1-TE-2016-2173 (TE141/2018), a grant of the Ministry of Research and Innovation, CNCS-UEFISCDI, awarded within PNCDI III.

## REFERENCES

[1] David Ahlström, Andy Cockburn, Carl Gutwin, and Pourang Irani. 2010. Why It's Quick to Be Square: Modelling New and Existing Hierarchical Menu Designs. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '10)*. ACM, New York, NY, USA, 1371–1380. https://doi.org/10.1145/1753326.1753534

[2] Erik Andersen, Yun-En Liu, Rich Snider, Roy Szeto, and Zoran Popović. 2011. Placing a Value on Aesthetics in Online Casual Games. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. ACM, New York, NY, USA, 1275–1278. https://doi.org/10.1145/1978942.1979131

[3] Michael P. Arcuri, Thomas Scott Coon, Jeffrey J. Johnson, Alexis Warren, Jacob Manning, and Martijn Eldert van Tilburg. 2000. Adaptive Menus, Patent US6121968A, Microsoft. (Sept. 2000). Filed June 17th, 1998, Issued Sep. 19th., 2000.

[4] Aslina Baharum and Azizah Jaafar. 2013. Users' Expectation of Web Objects Location: Case Study of ASEAN Countries. In *Third International Visual Informatics Conference on Advances in Visual Informatics - Volume 8237 (IVIC 2013)*. Springer-Verlag, Berlin, Heidelberg, 383–395. https://doi.org/10.1007/978-3-319-02958-0_35

[5] Gilles Bailly, Eric Lecolinet, and Laurence Nigay. 2008. Flower Menus: A New Type of Marking Menu with Large Menu Breadth, Within Groups and Efficient Expert Mode Memorization. In *Proceedings of the Working Conference on Advanced Visual Interfaces (AVI '08)*. ACM, New York, NY, USA, 15–22. https://doi.org/10.1145/1385569.1385575

[6] Gilles Bailly, Eric Lecolinet, and Laurence Nigay. 2016. Visual Menu Techniques. *Comput. Surveys* 49, 4, Article 60 (Dec. 2016), 41 pages. https://doi.org/10.1145/3002171

[7] Patrick Baudisch, Desney Tan, Maxime Collomb, Dan Robbins, Ken Hinckley, Ken Hinckley, Maneesh Agrawala, Shengdong Zhao, and Gonzalo Ramos. 2006. Phosphor: Explaining Transitions in the User Interface Using Afterglow Effects. In *Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology (UIST '06)*. ACM, New York, NY, USA, 169–178. https://doi.org/10.1145/1166253.1166280

[8] Benjamin B. Bederson. 2000. Fisheye Menus. In *Proceedings of the 13th Annual ACM Symposium on User Interface Software and Technology (UIST '00)*. ACM, New York, NY, USA, 217–225. https://doi.org/10.1145/354401.354782

[9] Michael Bernard. 2001. User Expectations for the Location of Web Objects. In *CHI '01 Extended Abstracts on Human Factors in Computing Systems (CHI EA '01)*. ACM, New York, NY, USA, 171–172. https://doi.org/10.1145/634067.634171

[10] Michael L. Bernard. 2003. Examining User Expectations for the Location of Common E-Commerce Web Objects. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 47, 11 (2003), 1356–1360. https://doi.org/10.1177/154193120304701108 arXiv:https://doi.org/10.1177/154193120304701108

[11] 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 (EICS '16)*. ACM, New York, NY, USA, 239–244. https://doi.org/10.1145/2933242.2935874

[12] 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 (EICS '16)*. ACM, New York, NY, USA, 239–244. https://doi.org/10.1145/2933242.2935874

[13] Sara Bouzit, Gaëlle Calvary, Denis Chêne, and Jean Vanderdonckt. 2017. Polymodal Menus: A Model-based Approach for Designing Multimodal Adaptive Menus for Small Screens. *Proc. ACM Hum.-Comput. Interact.* 1, EICS, Article 15 (June 2017), 19 pages. https://doi.org/10.1145/3099585

[14] Sara Bouzit, Denis Chêne, and Gaëlle Calvary. 2014. From Appearing to Disappearing Ephemeral Adaptation for Small Screens. In *Proceedings of the 26th Australian Computer-Human Interaction Conference on Designing Futures: The Future of Design (OzCHI '14)*. ACM, New York, NY, USA, 41–48. https://doi.org/10.1145/2686612.2686619

[15] Sara Bouzit, Denis Chêne, and Gaëlle Calvary. 2015. Evanescent Adaptation on Small Screens. In *Proceedings of the Annual Meeting of the Australian Special Interest Group for Computer Human Interaction (OzCHI '15)*. ACM, New York, NY, USA, 62–68. https://doi.org/10.1145/2838739.2838749

[16] Ralph Allan Bradley and Milton E. Terry. 1952. Rank Analysis of Incomplete Block Designs: I. The Method of Paired Comparisons. *Biometrika* 39, 3/4 (1952), 324–345. http://www.jstor.org/stable/2334029

[17] Robert Bridle and Eric McCreath. 2006. Inducing Shortcuts on a Mobile Phone Interface. In *Proceedings of the 11th International Conference on Intelligent User Interfaces (IUI '06)*. ACM, New York, NY, USA, 327–329. https://doi.org/10.1145/1111449.1111526

[18] 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

[19] Ph. Courcoux and M. Semenou. 1997. Preference data analysis using a paired comparison model. *Food Quality and Preference* 8, 5 (1997), 353 – 358. https://doi.org/10.1016/S0950-3293(97)00004-9 Third Sensometrics Meeting.

[20] Peter Emerson. 2013. The original Borda count and partial voting. *Social Choice and Welfare* 40, 2 (01 Feb 2013), 353–358. https://doi.org/10.1007/s00355-011-0603-9

[21] Leah Findlater and Krzysztof Z. Gajos. 2009. Design Space and Evaluation Challenges of Adaptive Graphical User Interfaces. *AI Magazine* 30, 4 (2009), 68–73. http://www.aaai.org/ojs/index.php/aimagazine/article/view/2268

[22] Leah Findlater and Joanna McGrenere. 2004. A Comparison of Static, Adaptive, and Adaptable Menus. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '04)*. ACM, New York, NY, USA, 89–96. https://doi.org/10.1145/985692.985704

[23] 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 (CHI '09)*. ACM, New York, NY, USA, 1655–1664. https://doi.org/10.1145/1518701.1518956

[24] Jon Fraenkel and Bernard Grofman. 2014. The Borda Count and its real-world alternatives: Comparing scoring rules in Nauru and Slovenia. *Australian Journal of Political Science* 49, 2 (2014), 186–205. https://doi.org/10.1080/10361146.2014.900530 arXiv:https://doi.org/10.1080/10361146.2014.900530

[25] 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, AVI 2006, Venezia, Italy, May 23-26, 2006*, Augusto Celentano (Ed.). ACM Press, 201–208. https://doi.org/10.1145/1133265.1133306

[26] Krzysztof Z. Gajos, Katherine Everitt, Desney S. Tan, Mary Czerwinski, and Daniel S. Weld. 2008. Predictability and Accuracy in Adaptive User Interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08)*. ACM, New York, NY, USA, 1271–1274. https://doi.org/10.1145/1357054.1357252

[27] Björn Hartmann, Loren Yu, Abel Allison, Yeonsoo Yang, and Scott R. Klemmer. 2008. Design As Exploration: Creating Interface Alternatives Through Parallel Authoring and Runtime Tuning. In *Proc. of the 21st Annual ACM Symposium on User Interface Software and Technology (UIST '08)*. ACM, New York, NY, USA, 91–100. https://doi.org/10.1145/1449715.1449732

[28] Neil T. Heffernan and Cristina Lindquist Heffernan. 2014. The ASSISTments Ecosystem: Building a Platform that Brings Scientists and Teachers Together for Minimally Invasive Research on Human Learning and Teaching. *International Journal of Artificial Intelligence in Education* 24, 4 (01 Dec 2014), 470–497. https://doi.org/10.1007/s40593-014-0024-x

[29] Anthony Hoeber, Alan Mandler, and Norman Cox. 1992. Method and apparatus for selecting button functions and retaining selected options on a display, Patent US5243697, Oracle America Inc. (1992). https://patents.google.com/patent/US5243697 Filed March 15th, 1992, Granted July 9th, 1993.

[30] Bowen Hui, Grant A. Partridge, and Craig Boutilier. 2009. A probabilistic mental model for estimating disruption. In *Proc. of the 14th Int. Conf. on Intelligent User Interfaces (IUI '08)*. ACM, New York, NY, USA, 287–296. https://doi.org/10.1145/1502650.1502691

[31] Ron Kohavi and Roger Longbotham. 2017. *Online Controlled Experiments and A/B Testing*. Springer US, Boston, MA, 922–929. https://doi.org/10.1007/978-1-4899-7687-1_891

[32] John Lamping, Ramana Rao, and Peter Pirolli. 1995. A Focus+Context Technique Based on Hyperbolic Geometry for Visualizing Large Hierarchies. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '95)*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 401–408. https://doi.org/10.1145/223904.223956

[33] 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. https://doi.org/10.1016/j.ergon.2003.07.004

[34] Xiaoyang Mao, Yuji Hatanaka, Atsumi Imamiya, Yuki Kato, and Kentaro Go. 2000. Visualizing Computational Wear with Physical Wear. In *Proceedings of the 6th ERCIM Workshop "User Interfaces for All" (UI4All '00)*, Pier-Luigi Emiliani and Constantine Stephanidis (Eds.). CNR-IROE, Pisa, Italy, 12. http://ui4all.ics.forth.gr/UI4ALL-2000/files/Long_papers/Mao.pdf

[35] Jeffrey Mitchell and Ben Shneiderman. 1989. Dynamic Versus Static Menus: An Exploratory Comparison. *SIGCHI Bulletin* 20, 4 (April 1989), 33–37. https://doi.org/10.1145/67243.67247

[36] Kazuma Murao, Carson Reynolds, and Masatoshi Ishikawa. 2012. Blink Suppression Sensing and Classification. In *CHI '12 Extended Abstracts on Human Factors in Computing Systems (CHI EA '12)*. ACM, New York, NY, USA, 2255–2260. https://doi.org/10.1145/2212776.2223785

[37] Jungchul Park, Sung H. Han, Yong S. Park, and Youngseok Cho. 2007. Usability of Adaptable and Adaptive Menus. In *Usability and Internationalization. HCI and Culture*, Nuray Aykin (Ed.). Springer, Berlin, Heidelberg, 405–411. https://doi.org/10.1007/978-3-540-73287-7_49

[38] Marcus Pivato. 2015. Condorcet meets Bentham. *Journal of Mathematical Economics* 59 (2015), 58 – 65. https://doi.org/10.1016/j.jmateco.2015.04.006

[39] Antoine Ponsard, Kamyar Ardekani, Kailun Zhang, Frederic Ren, Matei Negulescu, and Joanna McGrenere. 2015. Twist and Pulse: Ephemeral Adaptation to Improve Icon Selection on Smartphones. In *Proceedings of the 41st Graphics Interface Conference (GI '15)*. Canadian Information Processing Society, Toronto, Ont., Canada, Canada, 219–222. http://dl.acm.org/citation.cfm?id=2788890.2788929

[40] Ugo Sangiorgi and Jean Vanderdonckt. 2012. GAMBIT: Addressing Multi-platform Collaborative Sketching with Html5. In *Proceedings of the 4th ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS '12)*. ACM, New York, NY, USA, 257–262. https://doi.org/10.1145/2305484.2305527

[41] Andrew Sears and Ben Shneiderman. 1994. Split Menus: Effectively Using Selection Frequency to Organize Menus. *ACM Trans. Comput.-Hum. Interact.* 1, 1 (March 1994), 27–51. https://doi.org/10.1145/174630.174632

[42] A. Dawn Shaikh, Barbara S. Chaparro, and Anirudha Joshi. 2006. Indian Users' Expectations for the Location of Web Objects on Informational Websites. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 50, 17 (2006), 1922–1926. https://doi.org/10.1177/154193120605001744 arXiv:https://doi.org/10.1177/154193120605001744

[43] A. Dawn Shaikh and K. Lenz. 2006. WhereâĂŹs the search? Re-examining user expectations of web objects. *Usability News* 8, 1 (2006), 1356–1360.

[44] Jorge Gabriel Siqueira and Melise M. V. de Paula. 2018. IPEAD A/B Test Execution Framework. In *Proceedings of the XIV Brazilian Symposium on Information Systems (SBSI'18)*. ACM, New York, NY, USA, Article 14, 8 pages. https://doi.org/10.1145/3229345.3229360

[45] Theophanis Tsandilas and m. c. schraefel. 2005. An Empirical Assessment of Adaptation Techniques. In *CHI '05 Extended Abstracts on Human Factors in Computing Systems (CHI EA '05)*. ACM, New York, NY, USA, 2009–2012. https://doi.org/10.1145/1056808.1057079

[46] Jean Vanderdonckt, Sara Bouzit, Gaëlle Calvary, and Denis Chêne. 2018. Cloud Menus: A Circular Adaptive Menu for Small Screens. In *23rd International Conference on Intelligent User Interfaces (IUI '18)*. ACM, New York, NY, USA, 317–328. https://doi.org/10.1145/3172944.3172975

[47] Raynor Vliegendhart, Eelco Dolstra, and Johan Pouwelse. 2012. Crowdsourced User Interface Testing for Multimedia Applications. In *Proceedings of the ACM Multimedia 2012 Workshop on Crowdsourcing for Multimedia (CrowdMM '12)*. ACM, New York, NY, USA, 21–22. https://doi.org/10.1145/2390803.2390813

[48] Joseph Williams and Neil Heffernan. 2015. A Methodology for Discovering How to Adaptively Personalize to Users Using Experimental Comparisons. *Social Science Research Network* (2015). https://doi.org/10.2139/ssrn.2660585

[49] Joseph Jay Williams, Anna N. Rafferty, Dustin Tingley, Andrew Ang, Walter S. Lasecki, and Juho Kim. 2018. Enhancing Online Problems Through Instructor-Centered Tools for Randomized Experiments. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM, New York, NY, USA, Article 207, 12 pages. https://doi.org/10.1145/3173574.3173781

Received February 21st, 2019; revised April 15th, 2019; revised April 24th, 2019; accepted April 26th, 2019