Todi, Kashyap; Jokinen, Jussi; Luyten, Kris; Oulasvirta, Antti

Familiarisation

Published in:
IUI 2018 - Proceedings of the 23rd International Conference on Intelligent User Interfaces

DOI:
10.1145/3172944.3172949

Published: 05/03/2018

Document Version
Publisher's PDF, also known as Version of record

Please cite the original version:
Todi, K., Jokinen, J., Luyten, K., & Oulasvirta, A. (2018). Familiarisation: Restructuring layouts with visual learning models. In *IUI 2018 - Proceedings of the 23rd International Conference on Intelligent User Interfaces* (Vol. Part F135193, pp. 547-558). ACM. https://doi.org/10.1145/3172944.3172949

Familiarisation: Restructuring Layouts with Visual Learning Models

Kashyap Todi[1,2] kashyap.todi@gmail.com
Jussi Jokinen[1] jussi.jokinen@aalto.fi
Kris Luyten[2] kris.luyten@uhasselt.be
Antti Oulasvirta[1] antti.oulasvirta@aalto.fi

[1]Department of Communications and Networking
Aalto University
Helsinki, Finland

[2]Hasselt University - tUL - Flanders Make
Diepenbeek, Belgium

ABSTRACT

In domains where users are exposed to large variations in visuo-spatial features among designs, they often spend excess time searching for common elements (*features*) in familiar locations. This paper contributes computational approaches to restructuring layouts such that features on a new, unvisited interface can be found quicker. We explore four concepts of *familiarisation*, inspired by the human visual system (HVS), to automatically generate a familiar design for each user. Given a history of previously visited interfaces, we restructure the spatial layout of the new (unseen) interface with the goal of making its elements more easily found. *Familiariser* is a browser-based implementation that automatically restructures webpage layouts based on the visual history of the user. Our evaluation with users provides first evidence favouring familiarisation.

Author Keywords

Visual search; Graphical layouts; Computational design; Adaptive user interfaces

INTRODUCTION

This paper addresses a common predicament in interaction with graphical user interfaces: from blogs to banking and mobile apps, users encounter a wide diversity of visual designs. Even when serving the same purpose, designs differ in terms of element positions, colouring, images, and widget types. While visual uniqueness provides for identity and brand recognition, it also implies that users are constantly confronted with learning, relearning, and accustoming to navigating new structures and different styles. For the visual system, this poses a challenge to constantly adapt the use of visual attention. By understanding how this happens, we could design and adapt interfaces automatically such that elements can be more easily found. This could make interface ecologies more usable and enjoyable for users who care less about brand identity.

Our work is motivated by the observation that the added effort experienced upon encountering unfamiliar or atypical designs depends on the cost related to visual search for new and unfamiliar visual layouts [8, 18, 35]. Consider taking a familiar design and moving an element to a different position. Users would first seek the element in its expected place and, upon failing, continue with some search strategy to locate the element that was moved, or simply give up. However, designs are likely to differ in more than one feature, requiring 'guessing' of element positions, and further frustration.

This paper investigates computational principles that restructure unfamiliar designs to designs the user considers to be familiar. More formally, our goal can be defined as follows: *Given a new design d, unfamiliar to the user, and a history of previously visited visual designs H ($d \notin H$), restructure d to minimize costs to visual search to the user.* By 'restructuring' we refer to manipulations to the visio-spatial layout, such as moving, resizing, and recolouring elements. In this paper, we call techniques that achieve this *familiarisation techniques*. These techniques exploit the capability of operating systems and browsers to change an interface design dynamically. Such computer-driven familiarisation complements efforts at design-time to ensure *consistency* and adherence to design standards and guidelines [26, 31]. At the user-end, familiarisation exploits the known processes of human visual system in visual search. This ensures that the familiarised layouts are more usable due to predictable element locations and layout structures. While consistency enforced at design time can only approximate what the user population has experienced, ensuring consistency through active familiarisation at runtime allows, in principle, 'perfect consistency' for an individual.

There are multiple competing principles on which familiarisation can be based, depending on how familiarity is defined. As these principles can lead to different techniques of familiarisation, it is important to explore their assumptions, implementations, and results. To this end, we define and formalise four principles of familiarisation, informed by theories of human visual system (HVS) and visual learning. These are implemented in *Familiariser*, a browser-based system that dynamically restructures interface layouts to make them easier to use. Results from our study indicate that familiarisation can reduce visual search times by over 10%, and also leads to over 23% lesser eye-gaze fixations.

IUI'18, March 7–11, 2018, Tokyo, Japan

 ISBN 978-1-4503-4945-1/18/03…$15.00

DOI: https://doi.org/10.1145/3172944.3172949

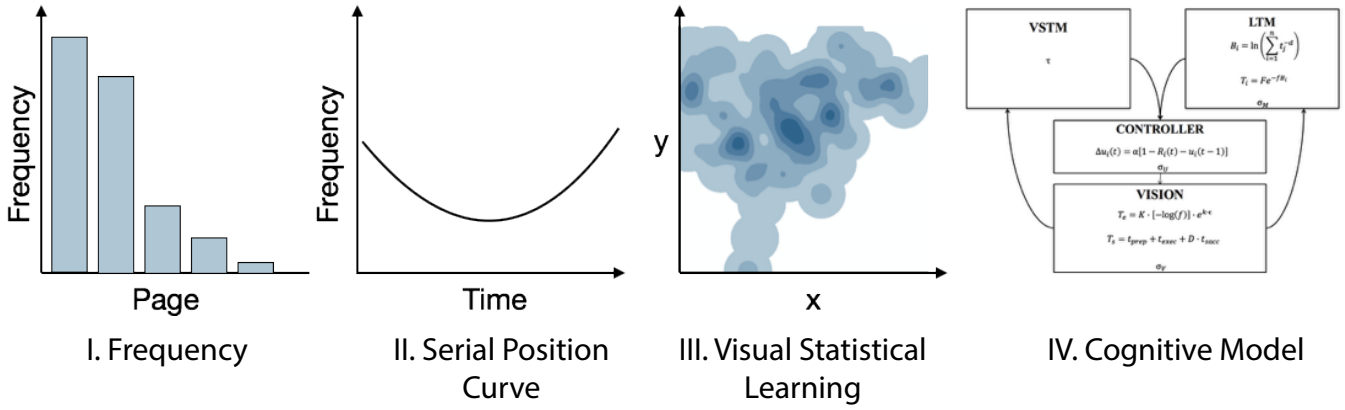


Figure 1. An illustration of the four principles for modelling familiarity. I selects the page with highest frequency; II considers frequency and time, selecting the page with the highest value on the curve; III considers positional (xy) information, and creates a probability distribution of features; IV uses a cognitive model of visual search to determine feature locations.

Overview: Four Familiarisation Principles

Familiarisation techniques can be generally defined as functions that take as input user history H and a new design d, and yield a restructured design d': $f(d,H) \rightarrow d'$. In this paper, we explore four principles with different theoretical underpinnings and consequences to how restructuring operates and how they influence results. Section *'Modelling Familiarity'* elaborates on each of these principles in greater detail.

I. Frequency: The most straightforward approach is to simply take the most *frequently used design* from H. Any new design within the same domain would be restructured such that elements appear in roughly the same places. While this design is likely to be recognizable, and simple to implement, the approach does not take into account the fact that most recent experiences are likely to dominate recall. Something that might have been frequently used in the past might be partially forgotten and overshadowed by the most recently encountered designs.

II. Serial Position Curve: The second principle implements the well-known serial position function of long-term recall [10]. Extensive empirical research on long-term and short-term memory has shown that the first and the most recently encountered objects are better recalled than ones in the middle. We implement a mathematical function that describes the relationship between order and recall probability. This is used to choose the most-likely-to-be-recalled design in H according to this function. A limitation of this and the first approach is that the visual features might not be adequately described by a single design.

III. Visual Statistical Learning: The third principle is visual statistical learning (VSL), according to which visual attention relies on a probabilistic internal model sensitive to the characteristics of the environments they have encountered before. We implement this hypothesis by building a statistical model of visual features in H. It is used to estimate the 'most probable' location of a given element. We then restructure a layout, element by element, considering this function, and finally align the elements so that they appear orderly.

IV. Cognitive Model: In our final technique, we train a cognitive model of layout learning with H and use its prediction on an empty canvas to predict the most likely positions of the elements of d. The model simulates learning of visual positions, and generates visual search patterns and times, given a layout and knowledge on that layout [18]. As an input it requires H and associated visitation durations, and optionally also considers element frequencies or relevance of elements. In this, it is similar to the previous approach. However, the model generates the actual eye movement patterns and visual search times, in addition to generating the memorability of location elements. It also simulates forgetting of layouts that have been visited only shortly and further back in time. This allows the model to be used for evaluating automatically how the familiarised layout will impact the user's behaviour, as well as how the user learns the new layout.

RELATED WORK

At its core, our work is situated within the domains of layout generation and restructuring, and user interface adaptation. We discuss prior literature that has dealt with these two themes, and draw comparisons to our work.

Layout Generation and Interface Restructuring

Automatic generation of layouts, and restructuring existing designs, have received significant attention by researchers and practitioners. One of the main rationales behind automatic generation is that it simplifies or eliminates the design task, making it possible for programmers or non-designers to create interfaces. Restructuring of designs allow for systematic improvement in usability and aesthetics of interfaces. Prior works have often used rules and heuristics to generate layouts that adhere to specific design guidelines [3, 6, 27, 28, 30], or example-based retargeting by mining existing designs [20, 21]. Our work situates itself in the area of model-based interface generation. Model-based techniques [5, 6, 30, 34] tackle the design problem by first abstracting the user interface as a set of models, which are then used to steer the generation or restructuring. There have been several prior systems, dating back to

the 1980s, such as [4, 17, 24, 29, 36], that apply model-based approaches to interface (re-)design.

The above techniques and implementations have largely considered the user population as a whole, and not individuals or groups of users. There has also been some significant work done to address different groups of users. Ability-based design [38] focuses on creating interfaces that take into account different requirements or restrictions, and optimise them towards specific abilities (e.g. [14, 33]). Culture-based design [19] took into account differences among different cultures, and used this as a basis for creating multiple designs of the same interface, each suited to a specific group of users. While all these works have considered a subset of the population, or specific preferences and requirements of groups of users, they generally do not address design on the per-user level, and also do not take into account individual users' past experiences or usage history. In contrast, with SUPPLE, Gajos et al. [11, 13] defined user interface generation as an optimisation problem that took a set of user interactions as input, and used this to generate an optimal solution. By doing so, the system could take into account specific user requirements and abilities, and create individualised interfaces for each user. HIGHLIGHT [23] enabled re-authoring of existing websites based on tracing how the user interacts with the site, and created mobile versions customised to the user tasks and mobile devices. UNIFORM [25] bears a lot of resemblance to our work, as it takes into account a user's history to automatically generated remote control interfaces. UNIFORM differs from our work since it does not restructure existing interfaces, yet focuses on generating consistent user interfaces that provide access to existing functionalities on new platforms. Additionally, the user history is limited to one source design, and the work focuses on the engineering aspects of restructuring an interface to match the source, and not on the systematic selection or generation of source designs from an extended history. The general approach adopted by previous works on per-user model-based interface generation has been to model the user as a one-time activity, before generating an optimised interface. Familiarisation, on the other hand, captures the complete user history, and each time a new interface is encountered, it generates a just-in-time redesigned interface based on an updated familiarity model. Interfaces can thus evolve over time and usage, and (re-)design becomes a continuous activity.

Run-time Adaptation of Interfaces

As highlighted above, familiarisation presents an approach to adapt and restructure interfaces at runtime. Such an approach falls under the category of 'adaptive user interfaces'—interfaces that can adapt or modify themselves while being used. There have been several works on adaptive UIs previously. Past systems have used different criteria for adapting interfaces. For example, the Walking User Interface in [38] adapted based on whether the user is stationary or walking. In the [12], Gajos presents UIs that adapt to users' current tasks. In contrast to use scenario or current task, Familiariser adapts an interface layout based on users' history, and exposure to previous interfaces belonging to the same domain. A common criticism and shortcoming of adaptive UIs is that unpredictability and cost of adaptation can negate the benefits provided [15, 22]. Since we focus on adapting newly visited interfaces that the user has not previously used or learnt, it does not suffer from these drawbacks. Thus, by incorporating model-based design generation, and just-in-time interface adaptation, our work attempts to leverage recall and visual learning, and provide users access to interfaces tailored to their expectations and mental models.

MODELLING FAMILIARITY

What we call 'familiarity' is a complex cognitive phenomenon influenced by several factors related to how people learn visual search. We approach the problem by exploring progressively sophisticated principles inspired by theories of HVS and human memory.

At the highest level, these principles break down to two groups: (a) page-wise familiarity; (b) feature-wise familiarity. In *page-wise familiarity*, an entire layout (e.g., webpage) is recognised as one unit. It is assumed that users learn and recall entire pages and their visual layouts. On the other hand, feature-wise familiarity considers high-level features on webpages as individual units. For example, logos, navigation menus, and search-boxes, can be considered as features, and users learn and recall these recurring features across different pages. Based on these two cases, we explore four principles:

Principle I: Frequency

This is the most straightforward *page-wise* approach to defining familiarity. It is informed by the frequency effect: frequently encountered items are more likely to be recalled than less frequently encountered. More specifically, the number of encounters directly affects both retrieval time and retention probability: more frequently practised items are recalled faster and easier [2]. The interface (webpage) that has been encountered the most number of times (i.e., most frequently visited), and for the longest durations, is assumed to be the most familiar to the user, and is thus used as the base design to which new pages are familiarised:

$$f_{page} = n_{visits} * t_{average} \tag{1}$$

where f_{page} is the familiarity score of a page, n_{visits} is number of visits, and $t_{average}$ is average duration of visits (in seconds). This results in a one-to-one retargeting of one page to another.

Principle II: Serial Position Curve

While frequency of usage can be a reasonable method to predict the most familiar interface in simple scenarios, as user histories get larger over time, this is susceptible to failure. Usage-based aspects such as first exposure to a page, recency of visits, and intervals between visits, are not considered by the first model. Firstly, not only frequency of rehearsals, but also their recency, affects recall time and probability: recently encountered and practised items are recalled faster and more probably [2]. In addition to recency, there is also an effect of primacy, where items that have been encountered first are better remembered than later items [16]. Together, the primacy and recency effects create a U-shaped curve (a *serial position curve*). It maps items encountered in the past to probability of recall: the first and last items are more likely to be recalled better.

This leads to a *page-wise* approach that considers:

1. *Frequency (v):* The frequency of visits to a given page, denoting how often a page is visited.

2. *Recency (r):* The recency of visit to a page, denoting when the page was last visited.

3. *Primacy (p):* The order of visits to different pages, or the sequence in which unique pages are encountered.

For each of these, scores are calculated for every page in the history by ranking them. By aggregating scores for all factors, for each page, we can calculate a familiarity score for every page, and hence determine the most familiar interface. Thus, we compute the following scores:

1. *Frequency Score (S_v):* This is the ratio of the visit count for the given page to the total visit count of all pages in the history.

$$S_v = n_{page}/n_{total} \tag{2}$$

where S_v is the frequency score, n_{page} is number of visits to page, and n_{total} is total number of visits to all pages.

2. *Recency Score (S_r):* This takes into account the decaying aspect of memory. The number of other pages that are visited since the user last visited the given page negatively influences the familiarity. The score is normalised (most recently visited page has a score of 1), and this reduces as recency increases

$$S_r = 1 - r_{page}/n_{total} \tag{3}$$

where S_r is the recency score, r_{page} is recency of a page, and n_{total} is total number of visits to all pages.

3. *Primacy Score (S_p):* Pages that are encountered first tend to be better recalled than later pages. This effect is known as primacy, and the *primacy score* takes this into account, where earlier pages have a higher score. The score is normalised (first visited page has a score of 1), and subsequently reduces for following pages.

$$S_p = 1 - ((p_{page} - 1)/n_{pages}) \tag{4}$$

where S_p is the primacy score, p_{page} is visit order number for the page, and n_{pages} is number of unique pages visited.

By aggregating these scores, we derive the familiarity score for a page:

$$F_{page} = \alpha * S_v + \beta * S_r + \gamma * S_p \tag{5}$$

where F_{page} is the familiarity score for the page, α, β, γ are weights for the three factors, and $\alpha + \beta + \gamma = 1$.

In our implementation, we assign equal weights to frequency and recency, and suggest that order has a lower effect on familiarity, thus assigning it a lower weight. Thus, the values we use are: $\alpha = \beta = 0.4$, $\gamma = 0.2$.

The page with the highest familiarity score ($max\ F_{page}$) is considered to be the familiar page for a user, and is selected as the basis for familiarisation.

Figure 2 presents a sample scenario, where a user visits four unique pages multiple times. As the user visits different pages, Figure 2(a) illustrates evolution in recorded usage metrics and scores, along with page selection results as per principles I and II.

Principle III: Visual Statistical Learning

This definition takes into account the statistical frequency with which different design features occurred in history H. This hypothesis is based on theories of visual statistical learning [7], according to which visual search strategies are sensitive to the statistical structure of the visual environment.

A *feature* is defined as a high-level semantic element, and can be composed of multiple low-level elements. For example, a website's logo is a feature, and could be composed of an image and a link element. By aggregating the commonly found positions of each feature among the visited pages in a user's history, we can determine the most familiar characteristics for each feature, and use this as the familiar design. Unlike in the previous principles, the resultant familiar design here is not a single page from the history, but a *feature mesh* consisting of familiar features from multiple pages in the user's history.

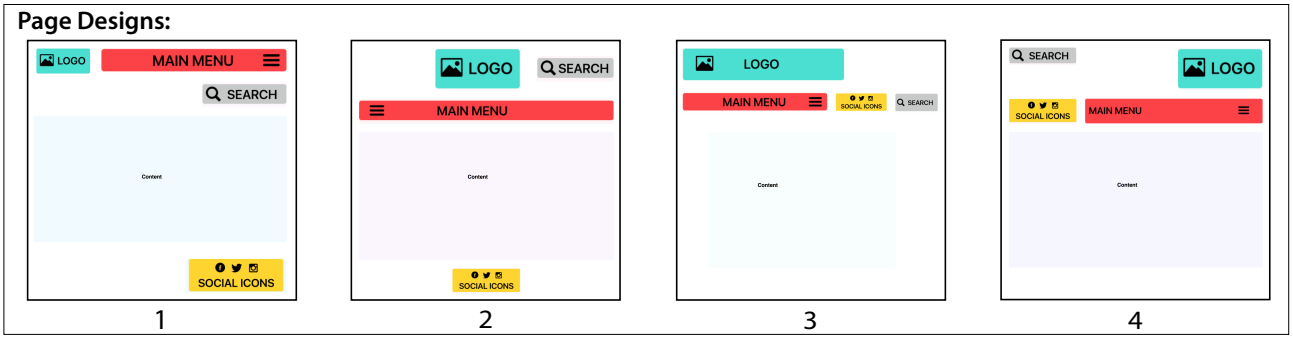
To generate this feature mesh, or familiar design, we first generate spatial probability distribution maps for each encountered feature. In these maps, for pages in the history, every for every pixel occupied by a particular feature, the spatial probability is incremented. Features are weighed according to familiarity scores (F_{page}) of each page (from #2). Thus, a feature appearing on a more familiar page has a higher influence on the probability distribution. For each feature, the value (position) with the highest probability is selected to create the familiar design consisting of all detected features. Figure 2(b) illustrates the computed probability distribution maps and selected positions for different features, for the given scenario.

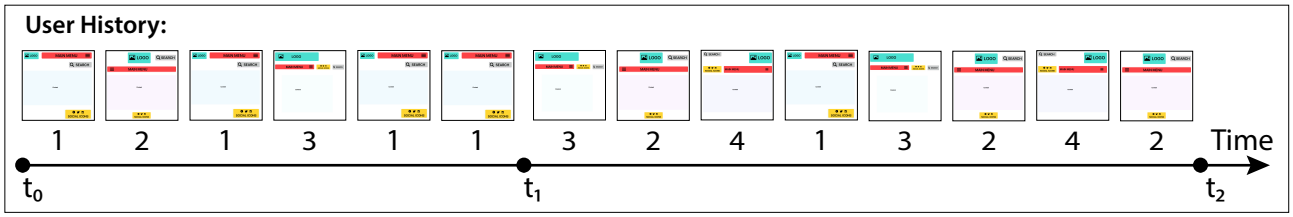
Principle IV: Visual Sampling Based on a Generative Cognitive Model

This definition uses a generative approach, where a model of visual search is used to generate gaze fixations as the simulated user is searching targets on a layout. The simulation integrates models of eye movements, visual short-term memory, and associative long-term memory, and proposes that visual search is the interaction of these three resources [18]. Given a user history with layouts, including locations of elements on the layouts, the model generates eye movement patterns as it simulates human-like visual search on the layouts. Using the model of eye movements [32], the simulation encodes elements on the layout until it has found the target. For each element, the encoding time is

$$T_e = K \cdot [-log(f)] \cdot e^{k \cdot \varepsilon}, \tag{6}$$

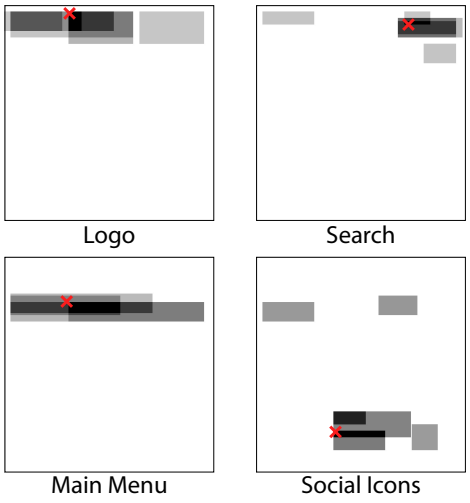
where f is the frequency of the object, either supplied externally to the model or assumed to be uniform, ε is the distance of the target from current eye fixation, and K and k are scaling constants, adapted from the literature [32]. T_e increases



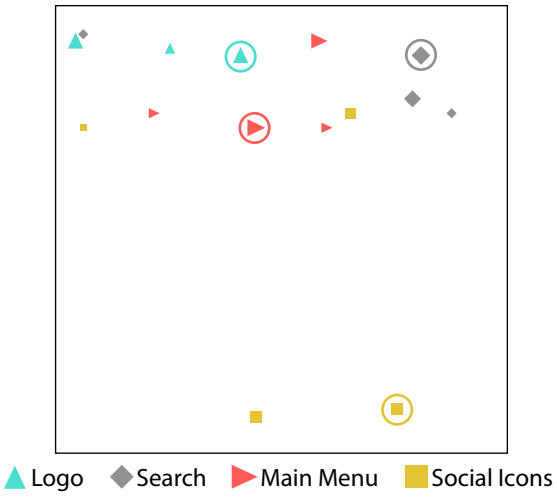


Timestamp	Page	Frequency (v)	Recency (r)	Primacy (p)	Frequency Score (S_v)	Recency Score (S_r)	Primacy Score (S_p)	Familiarity Score (F_{page})	
t_1	**1**	**3**	0	1	0.667	1.000	1.000	**0.867**	← Selected by I and II
	2	1	4	2	0.167	0.333	0.667	0.333	
	3	1	2	3	0.167	0.666	0.333	0.400	
	4	0	-	-	-	-	-	0	
t_2	**1**	**5**	4	1	0.357	0.714	1.000	**0.629**	← Selected by I
	2	4	0	2	0.286	1.000	0.750	**0.664**	← Selected by II
	3	3	3	3	0.214	0.786	0.500	0.500	
	4	2	2	4	0.143	0.857	0.250	0.479	

(a) Usage Metrics and Scores



(b) Probability Distribution Maps (for III)



(c) Activations (for IV)

Figure 2. Scenario with 4 different page designs. The user history shows a timeline of page visits (for simplicity, uniform visit durations are assumed across pages). (a) Evolution of metrics and scores at two timestamps (t_1, t_2), and page selection outcomes. (b) Computed probability distribution maps for key features at t_2. Red $\times$ symbols indicate the selected position for each feature. (c) Activations for features at t_2. Circled activations indicate the predicted positions for each feature.

exponentially as the target is further from the fixation, but the visual system may compensate this by initiating a fast eye movement to gaze closer to the target, with movement time of

$$T_s = t_{prep} + t_{exec} + D \cdot t_{sacc}, \quad (7)$$

where t_{prep}, t_{exec}, and t_{sacc} are constants from literature [32] and D is the movement amplitude.

After the eye movement, the target needs to be encoded (6). However, if the encoding time is less than t_{prep}, then the target is encoded without the eyes moving. This creates an eye movement model where the eyes move only when the target is too far to encode from the current gaze point.

In addition to visual search, the model simulates learning of layouts by using an activation-based associative memory model [1], its location is stored in the memory storage. An activation strength of an association can be calculated based on the number of times the target has been found:

$$B_i = \ln(\sum_{j=1}^{n} t_j^{-d}), \quad (8)$$

where t_j is the time since the j:th time of finding the target i, and d is a decay parameter, set from literature [18]. As the model learns the layout, it can use the associative memory to recall the position of the target without having to visually search for it. The model is able to recall the position of the target, if $B_i > 0$, with noise added from normal distribution [18]. Recall time is:

$$T_i = Fe^{-fB_i}, \quad (9)$$

where F and f are scaling constants, set based on literature [18].

Longer history with a layout results in higher associative activations, and faster, more expert-like performance in finding targets. From the activations, the model can predict where the user will gaze, given a layout and user history. Figure 2(c) illustrates activations, and predicted positions, for the given scenario.

FAMILIARISER: SYSTEM OVERVIEW

We implemented the above principles of familiarity in *Familiariser*, a browser-styled application that allows users to visit webpages and enables access to familiarised versions, adapted towards each user. As illustrated in Figure 3, when a user visits a page, the following pipeline is executed:

1. **Parse page:** The page source (HTML) is parsed by the system. Key elements, representing high-level features, are detected and labelled.

2. **Update history:** If the visited page does not exist in the history, it is added; if it was previously visited, the corresponding entry in the history is updated for the page. Additionally, familiarity scores, described in the previous section, are updated for every page in the user history.

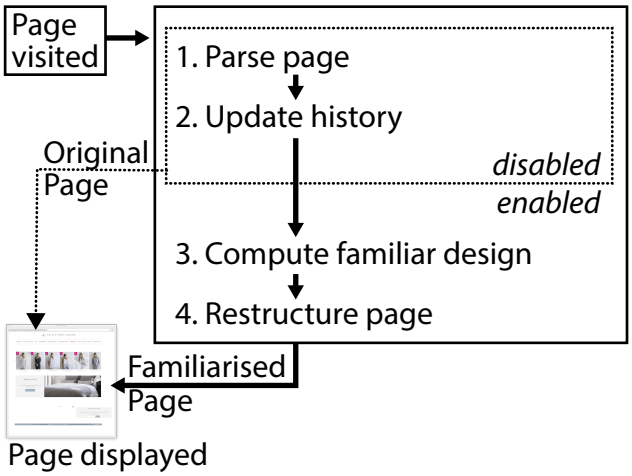


Figure 3. To compute a restructuring for a page, the history is updated for visited pages. Once familiarisation is enabled, the familiar design is computed using an HVS principle, and restructuring is instantiated via layout optimisation prior to rendering on a web browser.

While familiarisation is disabled, the pipeline ends at this point, and the original page is displayed as is. Once familiarisation is enabled, the following steps (3 and 4) are executed to familiarise the page.

3. **Compute familiar design:** Based on usage history, we compute the familiar page design, to be used as the basis for restructuring the new page.

4. **Restructure page:** The new page is restructured using positional values of matching features from the familiar design, and repositioning the corresponding elements on the visited page. Overlaps may occur in the restructured page. Familiariser resolves any such overlaps while attempting to best maintain relative alignments of elements on the page. This ensures a valid layout, without obscuring or omitting any contents. The familiarised page is finally displayed to the user.

Figure 4 illustrates two examples of results obtained by Familiariser, given a user history and original (unfamiliar) page, for each of the four principles of familiarity.

In the remainder of this section, we describe the various components of the system in greater detail.

Page Parsing

A *feature* can be defined as a task-level element [37]. They are actionable elements, with a well-defined purpose, such as the logo, search bar, buttons or icons to login and access a user account, or the shopping cart on e-commerce sites, etc. Features can be composed of several low-level HTML DOM elements. For familiarisation, it is necessary to first detect these features on a given page. Prior works have explored element detection when underlying sources were not available. Prefab [9] presented a pixel-based approach to reverse-engineering the GUI. Sikuli [39] allowed users to take screenshots of widgets and elements, and use these for search and automation of visual interfaces. For webpages, however, the underlying source files are openly accessible.

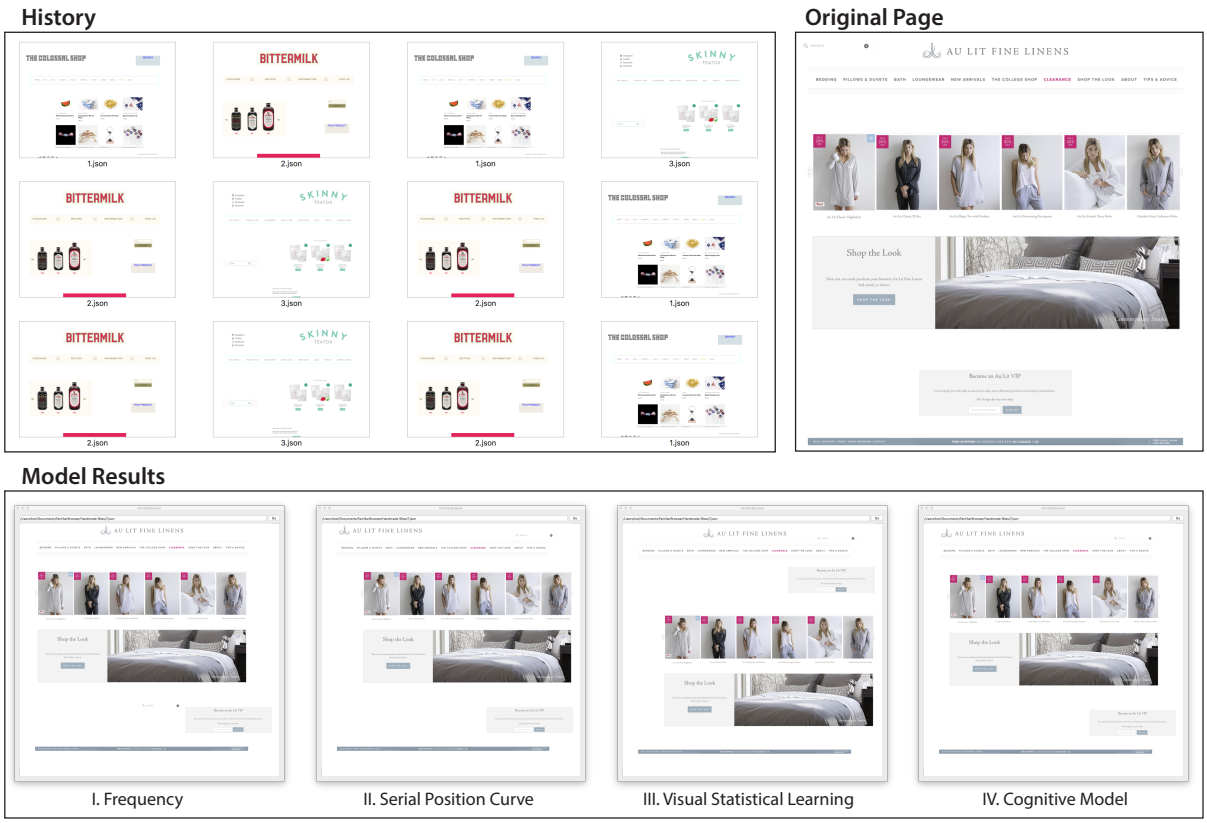


Figure 4. A visual comparison of the results for the four presented familiarity models, for two independent cases. For a given user history, when a new (unfamiliar) page is visited, each of the models may produce varying results.

Previously, Webzeitgeist [20] used CSS selectors to detect common features. Familiariser parses the underlying DOM tree of a page, and analyses DOM tags, identifiers, and linked CSS classes, of elements to automatically detect features on the page. We use partial string matching to detect commonly-used names, and map them to corresponding features. A feature can either be a leaf node in the DOM tree, or a compound element consisting of several smaller elements. The detection of features relies on appropriate naming and tagging of underlying HTML elements, and this can be a limiting factor in finding all matching features across different pages. Feature recognition could be improved by employing other computational methods, such as image recognition or machine learning, but this is out of the scope of this work.

Usage History Updates

For each user, page visit history is recorded, and usage-based metrics are updated each time new pages are visited. For each page, a unique entry in the history is maintained, and includes information related to time spent on the page (duration), frequency of visits, recency, and order of visits. When an existing page in history is re-visited, the frequency (f_{page}) is incremented, and the recency of that page is reset such that it is the "most recent" page (i.e. $r_{page} = 0$). For every other page in the history, the recency (r_{page}) is incremented, thus decreasing the recency score (S_r).

Familiar Design Computation

All pages in the history are considered to compute the familiar layout design. For web browsing specifically, different categories of websites have distinctly different features. To avoid mismatch in domain, website categorisation can be performed as an intermediate step, and only pages within the same category as the currently-visited page are considered. For example, if the user intends to visit a shopping website, only pages from the shopping domain are considered while computing the base layout design.

Principles I and II consider a single page as the 'familiar design'. Familiarity scores are calculated accordingly, and the highest-scoring page is selected. Matching features are extracted from the familiar design, and their xy positions are used for the base layout design. Principles III and 4 compute familiar designs based on all pages in the user history. For principle III, probability distribution maps are created for features detected on all visited pages. Using these, the most likely position for each feature is estimated, resulting in a base design where all feature are located appropriately. For principle IV, activation points for each feature are computed. The point with the highest activation, for a feature, is predicted to be the most familiar position, and hence used for the base design.

Target Page Restructuring

The base design that is computed is used to restructure the newly-visited page. Features on the target page are matched to those on the base design, and repositioned accordingly. In [21], retargeting occasionally resulted in truncated or cropped content due to size mismatch. In our approach, while repositioning, dimensions of the original elements are maintained so as to avoid truncation of contents. This can however result in some overlapping or occlusion of elements.

Overlap Resolution

Overlaps are resolved by setting up a series of overlap-redressal rules. Examples of rules used in Familiariser are as follows:

(a) Left-alignment or top-alignment of two (or more) elements should not be violated. If this is violated, a penalty is applied.

(b) Movement of any element from its preferred (horizontal) location entails a penalty.

(c) The vertical or horizontal sequence of any pair of elements should be honoured.

(d) The canvas width should not be changed. The height may be changed if needed.

The rules are implemented using any standard integer-linear programming solver, resulting in a valid layout without overlaps.

Enabling Familiarisation

During the initial stages of browsing, familiarisation is *disabled*. As users visit different pages, they gradually learn visual layouts of these, thus increasing chances of recall during future visits. Once a user is fluent with a small set of pages, familiarisation is *enabled*, and newly visited pages are be adapted to computed familiar designs. However, we need to determine the ideal moment to enable familiarisation, to make it effective. There is a trade-off between familiarising too early and too late. If enabled too early, users may not have learnt visited pages sufficiently, thus future pages would not benefit from adaptation. On the other hand, if familiarisation is delayed, then users might have already been exposed to a large number of diverse designs, thus making recall harder. For the purpose of our study, we determined (by trial-and-error) that enabling familiarisation after 25 page visits was favourable, given that the number of unique pages was less than or equal to 5. For future systems, familiarity scores can be used to empirically determine when to enable familiarisation, or this could be customisable per user.

With automatic (always-on) familiarisation, pages are automatically adapted once enabled. Familiariser also supports manual (on-demand) familiarisation, where users can request for a familiarised version of page. Here, the original page is displayed by default, and a familiarised version is rendered only when requested. This is similar to how web services such as translation and reader-friendly modes enable users to request for adapted versions of a page. Manual familiarisation obviates the need for a determining a fixed point at which adaptations are enabled.

ARCHITECTURE AND IMPLEMENTATION

Familiariser is implemented in Swift, on MacOS 10.13, as a standalone browser-styled application. In the back-end, the application contains a history model for each user, containing

a dictionary of visited websites. Along with the website URL, and parsed contents, each website object also contains relevant usage information required for modelling familiarity (as described in section 'Modelling Familiarity'). Every time a new page is visited, custom JavaScript code parses the page source, and converts the page into a flattened JSON file, containing absolute (xy) positions of elements, as they appear on the browser, and other element properties, such as tags and CSS styles, required to recreate the page and to infer features. The JSON file is parsed, to construct a webpage object. The cognitive learning model (described in principle IV) is implemented using Common Lisp. Each time familiarisation is required, a CSV file containing relevant history information, including access timestamp, duration, and linked JSON file name, is generated. The CSV file and required JSON files for all webpages are passed as input to the learning model. As output, the model generates a CSV file containing activation points for all detected features, along with confidence values, and returns this to Familiariser. Familiariser uses this to select the desirable positions for features, to generate the familiar design. Overlaps in layout elements are resolved by passing a file containing element positions to a Java application. The application applies overlap-redressal rules to each element. We use an IBM CPLEX linear programming solver to optimally resolve any detected overlaps. The solver returns a corresponding file with resultant element positions. This is used by Familiariser to generate the final valid layout, which is then displayed to the user.

EVALUATION

The goal of our evaluation is to verify the concept of familiarisation, and the presented system. We seek to answer the question: *Does familiarisation improve performance in web browsing for end-users?* To this end, we conducted a comparative user study, and report on quantitative results. We compared original (unmodified) designs against familiarised designs in a study where users were asked to point-and-click on different features on the displayed page. We used the visual statistical learning principle (principle III) for familiarisation as these selection tasks were quite brief in duration. As dependent variables, we analysed visual search time, approximated by pointing time, and number of eye-gaze fixations per target feature. Comparing these two cases enables us to evaluate the potential effects of familiarisation during real-world usage. For the test case, we chose the domain of shopping websites, a frequently used category of webpages that are also plenty in number. These websites typically contain similar features yet vary vastly in their layouts and presentation of these features. This makes them a good test candidate for Familiariser, and provides a realistic use scenario for the study.

Study Tasks

For the study, participants were given the task of selecting (clicking on) a feature element on the displayed webpage, in a browser-based application. Webpages were selected randomly, from a dataset of 30 shopping sites. The target feature was also selected randomly from the page, and included commonly occurring elements, such as logos, navigation menus, search boxes, among others. Participants were requested to perform tasks quickly yet accurately as possible, and avoid unnecessary pauses. As tasks were performed, the software logged mouse movements, click events, and eye gaze information, including eye position (averaged) and fixations.

Apparatus

A 13" MacBook Pro with Retina Display (2560-by-1600 pixels), running MacOS 10.13, was used for the study. The browser-based Familiariser software was displayed as a full-screen application. We selected 30 shopping websites to create the dataset, excluding commonly-used shopping pages to avoid bias. Webpages used in the dataset were rendered offline to avoid delays. Tasks were completed using the in-built trackpad. All cursor motions and events were logged by the study software. For eye-tracking, an EyeTribe Tracker[1] was used, along with a custom Python program to log gaze positions and fixations.

Participants

We recruited 16 participants, aged 21 to 36 (mean 29), with no visual impairments. All participants reported frequent web usage (daily), and also reported exposure to shopping websites. Participants completed tasks with both original (unmodified) and test (familiarised) pages, in a within-subject study design.

Method

During the experiment, participants were exposed to a set of different websites. Before a website was displayed, they were presented with the task of selecting (clicking on) a particular feature element (e.g. logo, main menu, search box) on the page. The target feature element was selected at random from the page to be displayed. The timed trial started once the participant confirmed they were ready, by clicking on a confirmation button. This button was consistently placed at the centre of the screen, ensuring a constant starting point for the cursor during all trials.

The experiment was divided into two phases: *Learning Phase* and *Test Phase*.

1. Learning Phase

For each participant, the system selected 5 webpages, at random, from the dataset of 30 pages, for the learning phase. The learning dataset thus varied for each participant. During each trial, a page from this subset was selected at random, and the participant was asked to point and click at a randomly chosen feature on the page. A total of 25 trials were performed during this phase, and this was used to construct the 'usage history' for the participant.

2. Test Phase

Once the participant had experienced this subset of pages in the training phase, the experiment moved on to the test phase. The 5 pages used in the learning phase were stored in the user history, and excluded from the test dataset. Similar to the learning phase, participants again browsed to randomly selected pages, and were asked to select a target feature on the page. During the test trials, either the original (unmodified) page or a familiarised version was presented to the user, chosen

[1] www.theeyetribe.com

	Visual Search Time	Fixation Count
Original	2.8 seconds	3.4
Familiarised	2.5 seconds	2.6

Table 1. Summary of results for average visual search time and fixation count per target feature.

at random. Participants performed a total of 100 timed trials during the test phase.

The average duration of the study was approximately 30 minutes for each participant.

Results

We created a linear mixed model with search time as dependent variable, page type (Original vs. Familiarised) as fixed independent variable, and participant id and page URL as random variables. Grand mean search times were Original = 2.8 seconds and Familiarised = 2.5 seconds. The difference was statistically significant, $t(663) = 5.3, p < .001$, with model $F(1,663) = 28.5, p < .001$. In standardised units, the benefit of familiarised layout was $\beta = 0.35$.

We compared similarly the number of fixations per target. Grand mean fixation counts were Original = 3.4 and Familiarised = 2.6. The difference was statistically significant, $t(596) = 4.7, p < .001$, with model $F(1,596) = 22.3, p < .001$. In standardised units, the benefit of familiarised layout was $\beta = 0.35$.

The above results indicate that familiarisation improved user performance by reducing both visual search time and number of gaze fixations. Our study evaluates familiarisation using the visual statistical learning principle. A comparison of all four principles requires a more extensive study, and is left as subject of future work.

DISCUSSION

Our work on familiarisation of visual designs situates itself in the field of automatically generated user interfaces that adapt to individual users. It deals with concepts of recall and visual learning to make interfaces appear closer to each user's expectations. We explore four principles of familiarity, inspired by the human visual system, and grounded in literature. Familiariser implements these in an end-user system that captures users' history, and restructures newly visited pages based on automatically generated familiar layout designs. Results from the empirical study provide evidence for our approach. Familiarisation significantly improves visual search time by over 10%, and reduces the number of fixations by over 23%, while searching for features on a given design.

The cost of adaptation is often a concern for adaptive user interfaces. Our work circumvents this as familiarisation restructures only new and unfamiliar designs, instead of continuously adapting or modifying frequently-used layouts. One could criticise our approach for compromising brand identity, or undermining the designer, by modifying designs. However, we argue that usability of an interface supersedes these aspects for the end-user. Additionally, Familiariser addresses this by allowing users to optionally view either original or familiarised versions of designs. Commercial browsers have also used such techniques to improve usability of webpages. For instance, 'reader-friendly mode' on browsers allows users to switch between the original page and a version enhanced for reading.

There are certain limitations for applying familiarisation universally, as it requires (1) logging of a user's history of seen layouts, (2) detailed information and representation of layouts, (3) just-in-time computations of familiar design, and (4) instantiation of page restructuring in runtime, prior to rendering it on a display. By exploring a range of principles for familiarisation, we provide alternatives that enable systems to circumvent some of these limitations. The basic frequency-based approach (principle I) is straightforward to implement, and requires minimal user history information. Serial-position curve (II) requires some additional usage information, and requires calculations of various scores to select a page as the familiar design. The feature-based principles (III and IV) require detailed information about the interface layout, and are computationally more expensive, but offer more accurate representations for a user.

While this paper, and our implementation, focuses on repositioning of features, future familiarisation efforts can apply our work to address other interface aspects, such as colours, fonts, and other visuo-perceptual properties. Such properties are often position-agnostic, and thus for these page-wise familiarisation (I or II) is preferred. Apart from addressing additional interface properties, usability and experience with user interfaces can be further improved. Future systems can explore a dual-optimisation strategy, where the familiarity model can be combined with other predictive models of human perception. Additionally, more interfaces can be covered by applying the concept of familiarity to a variety of mediums (digital, physical, hybrid). For instance, it can be possible to familiarise physical interfaces, adapting them to resemble previously encountered digital or physical interfaces. Finally, by providing a formal model of familiarity, we can implicitly gain an understanding of "unfamiliarity". This opens up possibilities of other applications such as 'unfamiliarisation' or diversification of interfaces. This could encourage alternative design goals such as exploration, or be used to draw users' attention to certain elements.

More information and material related to this paper can be found at http://www.kashyaptodi.com/familiarisation.

ACKNOWLEDGEMENTS

The project has partially received funding from the Academy of Finland project COMPUTED and the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement 637991).

Among others, we thank Niraj Damaya for code related to overlap resolution, Markku Laine for his useful comments, and study participants for their time and involvement.

REFERENCES

1. John R Anderson, D Bothell, C Lebiere, and M Matessa. 1998. An integrated theory of list memory. *Journal of Memory And Language* 38, 4 (1998), 341–380. DOI: http://dx.doi.org/10.1006/jmla.1997.2553
2. John R Anderson, Jon M Fincham, and Scott Douglass. 1999. Practice and retention: A unifying analysis. *Journal of Experimental Psychology-Learning Memory and Cognition* 25, 5 (1999), 1120–1136.
3. Yigal Arens, Lawrence Miller, Stuart C. Shapiro, and Norman K. Sondheimer. 1988. Automatic Construction of User-interface Displays. In *Proceedings of the Seventh AAAI National Conference on Artificial Intelligence (AAAI'88)*. AAAI Press, 808–813. http://dl.acm.org/citation.cfm?id=2887965.2888108
4. Gilles Bailly, Antti Oulasvirta, Timo Kötzing, and Sabrina Hoppe. 2013. MenuOptimizer: Interactive Optimization of Menu Systems. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology (UIST '13)*. ACM, New York, NY, USA, 331–342. DOI: http://dx.doi.org/10.1145/2501988.2502024
5. C. M. Beshers and S. Feiner. 1989. Scope: Automated Generation of Graphical Interfaces. In *Proceedings of the 2Nd Annual ACM SIGGRAPH Symposium on User Interface Software and Technology (UIST '89)*. ACM, New York, NY, USA, 76–85. DOI: http://dx.doi.org/10.1145/73660.73670
6. François Bodart, Anne-Marie Hennebert, Jean-Marie Leheureux, and Jean Vanderdonckt. 1994. Towards a Dynamic Strategy for Computer-aided Visual Placement. In *Proceedings of the Workshop on Advanced Visual Interfaces (AVI '94)*. ACM, New York, NY, USA, 78–87. DOI:http://dx.doi.org/10.1145/192309.192328
7. Zoya Bylinskii, Nam Wook Kim, Peter O'Donovan, Sami Alsheikh, Spandan Madan, Hanspeter Pfister, Fredo Durand, Bryan Russell, and Aaron Hertzmann. 2017. Learning Visual Importance for Graphic Designs and Data Visualizations. *arXiv preprint arXiv:1708.02660* (2017).
8. Marvin M Chun and Yuhong Jiang. 1998. Contextual cueing: Implicit learning and memory of visual context guides spatial attention. *Cognitive psychology* 36, 1 (1998), 28–71.
9. Morgan Dixon and James Fogarty. 2010. Prefab: Implementing Advanced Behaviors Using Pixel-based Reverse Engineering of Interface Structure. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '10)*. ACM, New York, NY, USA, 1525–1534. DOI: http://dx.doi.org/10.1145/1753326.1753554
10. Peter A Frensch. 1994. Composition during serial learning: A serial position effect. *JOURNAL OF EXPERIMENTAL PSYCHOLOGY LEARNING MEMORY AND COGNITION* 20 (1994), 423–423.
11. Krzysztof Gajos and Daniel S. Weld. 2004. SUPPLE: Automatically Generating User Interfaces. In *Proceedings of the 9th International Conference on Intelligent User Interfaces (IUI '04)*. ACM, New York, NY, USA, 93–100. DOI:http://dx.doi.org/10.1145/964442.964461
12. Krzysztof Z. Gajos, Mary Czerwinski, Desney S. Tan, and Daniel S. Weld. 2006a. Exploring the Design Space for Adaptive Graphical User Interfaces *(AVI '06)*. ACM, New York, NY, USA, 201–208. DOI: http://dx.doi.org/10.1145/1133265.1133306
13. Krzysztof Z. Gajos, Jing Jing Long, and Daniel S. Weld. 2006b. Automatically Generating Custom User Interfaces for Users with Physical Disabilities. In *Proceedings of the 8th International ACM SIGACCESS Conference on Computers and Accessibility (Assets '06)*. ACM, New York, NY, USA, 243–244. DOI: http://dx.doi.org/10.1145/1168987.1169036
14. Krzysztof Z. Gajos, Jacob O. Wobbrock, and Daniel S. Weld. 2007. Automatically Generating User Interfaces Adapted to Users' Motor and Vision Capabilities. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology (UIST '07)*. ACM, New York, NY, USA, 231–240. DOI: http://dx.doi.org/10.1145/1294211.1294253
15. Saul Greenberg and Ian H. Witten. 1985. Adaptive personalized interfaces–A question of viability. *Behaviour & Information Technology* 4, 1 (1985), 31–45. DOI:http://dx.doi.org/10.1080/01449298508901785
16. Richard NA Henson. 1996. Unchained memory: Error patterns rule out chaining models of immediate serial recall. *The Quarterly Journal of Experimental Psychology: Section A* 49, 1 (1996), 80–115.
17. Christian Janssen, Anette Weisbecker, and Jürgen Ziegler. 1993. Generating User Interfaces from Data Models and Dialogue Net Specifications. In *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems (CHI '93)*. ACM, New York, NY, USA, 418–423. DOI: http://dx.doi.org/10.1145/169059.169335
18. Jussi P. P. Jokinen, Sayan Sarcar, Antti Oulasvirta, Chaklam Silpasuwanchai, Zhenxin Wang, and Xiangshi Ren. 2017. Modelling Learning of New Keyboard Layouts. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 4203–4215. DOI: http://dx.doi.org/10.1145/3025453.3025580
19. Andruid Kerne, William A. Hamilton, and Zachary O. Toups. 2012. Culturally Based Design: Embodying Trans-surface Interaction in Rummy. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work (CSCW '12)*. ACM, New York, NY, USA, 509–518. DOI: http://dx.doi.org/10.1145/2145204.2145284

20. Ranjitha Kumar, Arvind Satyanarayan, Cesar Torres, Maxine Lim, Salman Ahmad, Scott R. Klemmer, and Jerry O. Talton. 2013. Webzeitgeist: Design Mining the Web. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 3083–3092. DOI: http://dx.doi.org/10.1145/2470654.2466420

21. Ranjitha Kumar, Jerry O. Talton, Salman Ahmad, and Scott R. Klemmer. 2011. Bricolage: Example-based Retargeting for Web Design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. ACM, New York, NY, USA, 2197–2206. DOI: http://dx.doi.org/10.1145/1978942.1979262

22. Talia Lavie and Joachim Meyer. 2010. Benefits and Costs of Adaptive User Interfaces. *Int. J. Hum.-Comput. Stud.* 68, 8 (Aug. 2010), 508–524. DOI: http://dx.doi.org/10.1016/j.ijhcs.2010.01.004

23. Jeffrey Nichols and Tessa Lau. 2008. Mobilization by Demonstration: Using Traces to Re-author Existing Web Sites. In *Proceedings of the 13th International Conference on Intelligent User Interfaces (IUI '08)*. ACM, New York, NY, USA, 149–158. DOI: http://dx.doi.org/10.1145/1378773.1378793

24. Jeffrey Nichols, Brad A. Myers, Michael Higgins, Joseph Hughes, Thomas K. Harris, Roni Rosenfeld, and Mathilde Pignol. 2002. Generating Remote Control Interfaces for Complex Appliances. In *Proceedings of the 15th Annual ACM Symposium on User Interface Software and Technology (UIST '02)*. ACM, New York, NY, USA, 161–170. DOI:http://dx.doi.org/10.1145/571985.572008

25. Jeffrey Nichols, Brad A. Myers, and Brandon Rothrock. 2006. UNIFORM: Automatically Generating Consistent Remote Control User Interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '06)*. ACM, New York, NY, USA, 611–620. DOI:http://dx.doi.org/10.1145/1124772.1124865

26. Jakob Nielsen. 1993. *Usability Engineering*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

27. Peter ODonovan, Aseem Agarwala, and Aaron Hertzmann. 2014. Learning Layouts for Single-PageGraphic Designs. *IEEE Transactions on Visualization and Computer Graphics* 20, 8 (Aug. 2014), 1200–1213. DOI: http://dx.doi.org/10.1109/TVCG.2014.48

28. Peter O'Donovan, Aseem Agarwala, and Aaron Hertzmann. 2015. DesignScape: Design with Interactive Layout Suggestions. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. ACM, New York, NY, USA, 1221–1224. DOI: http://dx.doi.org/10.1145/2702123.2702149

29. D. R. Olsen, Jr. 1989. A Programming Language Basis for User Interface. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '89)*. ACM, New York, NY, USA, 171–176. DOI: http://dx.doi.org/10.1145/67449.67485

30. Angel R. Puerta, Henrik Eriksson, John H. Gennari, and Mark A. Musen. 1994. Model-based Automated Generation of User Interfaces. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (Vol. 1) (AAAI '94)*. American Association for Artificial Intelligence, Menlo Park, CA, USA, 471–477. http://dl.acm.org/citation.cfm?id=199288.184583

31. John Rheinfrank and Shelley Evenson. 1996. Design languages. In *Bringing design to software*. ACM, 63–85.

32. Dario D Salvucci. 2001. An Integrated Model of Eye Movements and Visual Encoding. *Cogn. Syst. Res.* 1, 4 (Feb. 2001), 201–220. DOI: http://dx.doi.org/10.1016/S1389-0417(00)00015-2

33. Sayan Sarcar, Jussi Jokinen, Antti Oulasvirta, Xiangshi Ren, Chaklam Silpasuwanchai, and Zhenxin Wang. 2018. Ability-Based Optimization: Designing Smartphone Text Entry Interface for Older Adults. *IEEE Pervasive Computing* (2018).

34. A. Sears. 1993. Layout appropriateness: a metric for evaluating user interface widget layout. *IEEE Transactions on Software Engineering* 19, 7 (Jul 1993), 707–719. DOI:http://dx.doi.org/10.1109/32.238571

35. Andrew Sears, Julie A Jacko, Josey Chu, and Francisco Moro. 2001. The role of visual search in the design of effective soft keyboards. *Behaviour & Information Technology* 20, 3 (2001), 159–166.

36. Kashyap Todi, Daryl Weir, and Antti Oulasvirta. 2016. Sketchplore: Sketch and Explore with a Layout Optimiser. In *Proceedings of the 2016 ACM Conference on Designing Interactive Systems (DIS '16)*. ACM, New York, NY, USA, 543–555. DOI: http://dx.doi.org/10.1145/2901790.2901817

37. Martijn Van Welie and Gerrit C Van der Veer. 2003. Pattern languages in interaction design: Structure and organization. In *Proceedings of interact*, Vol. 3. 1–5.

38. Jacob O. Wobbrock, Shaun K. Kane, Krzysztof Z. Gajos, Susumu Harada, and Jon Froehlich. 2011. Ability-Based Design: Concept, Principles and Examples. *ACM Trans. Access. Comput.* 3, 3, Article 9 (April 2011), 27 pages. DOI:http://dx.doi.org/10.1145/1952383.1952384

39. Tom Yeh, Tsung-Hsiang Chang, and Robert C. Miller. 2009. Sikuli: Using GUI Screenshots for Search and Automation. In *Proceedings of the 22nd Annual ACM Symposium on User Interface Software and Technology (UIST '09)*. ACM, New York, NY, USA, 183–192. DOI: http://dx.doi.org/10.1145/1622176.1622213