Anchored Customization: Anchoring Settings to the Application Interface to Afford Customization

Antoine Ponsard
University of British Columbia
aponsard@cs.ubc.ca

Joanna McGrenere
University of British Columbia
joanna@cs.ubc.ca

ABSTRACT

The settings panel is the standard customization mechanism used in software applications today, yet it has undergone minimal design improvement since its introduction in the 1980s. Entirely disconnected from the application UI, these panels require users to rely on often-cryptic text labels to identify the settings they want to change. We propose the Anchored Customization approach, which anchors settings to *conceptually related* elements of the application UI. Our Customization Layer prototype instantiates this approach: users can see which UI elements are customizable, and access their associated settings. We designed three variants of Customization Layer based on multi-layered interfaces, and implemented these variants on top of a popular web application for task management, Wunderlist. Two experiments (Mechanical Turk and face-to-face) with a total of 60 participants showed that the two minimalist variants were 35% faster than Wunderlist's settings panel. Our approach provides significant benefits for users while requiring little extra work from designers and developers of applications.

Author Keywords
customization; settings; contextual interaction.

ACM Classification Keywords
H.5.2 User Interfaces: Graphical user interfaces (GUI)

INTRODUCTION

The HCI community has identified the potential benefits of supporting users to adapt their software to their own tasks and preferences [26]. Yet many users do not customize, or only do so rarely. In today's software, the de facto standard customization mechanism is the settings panel, also known as the options menu or preferences dialog. These panels have significant usability limitations in that the settings they offer are entirely disconnected from the application UI. There are no visual affordances in the UI that can help answer the question: "Is it possible to customize X?" Users must open the settings panel and then rely on text labels to identify the settings they want to change, with little to no other contextual information. Thus, the classic vocabulary problem applies: the language used by designers to describe an aspect of the interface is often different from the language of users [17]. This creates a deep gulf of execution between the user's intentions, formulated in the context of the application UI , and the actions offered in the settings panel. The fact that customization occurs fairly infrequently over long periods of time [25] only exacerbates the problem: users are likely to forget where things are in the settings panel between customization episodes.

CHI'16, May 07 - 12, 2016, San Jose, CA, USA
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-3362-7/16/05$15.00
DOI: http://dx.doi.org/10.1145/2858036.2858129

Despite obvious usability problems, settings panels have undergone only minor improvements since they were first introduced along with the graphical user interface in the 1980s [22]. The interaction paradigm fundamentally hasn't changed: the user can browse tabs of settings, tick checkboxes, and choose values from drop-downs. These panels essentially represent a *developer's* point of view of the application's customization opportunities: they are simple graphical representations of traditional UNIX config files.

We propose Anchored Customization, a novel approach to customization designed from the *user's* point of view: *anchoring* settings to visual elements of the UI that are *conceptually related* to these settings. This reduces the gulf of execution, and leverages users' existing knowledge of the application UI. While others have proposed the idea of moving the access point of a customization closer to the feature that it affects [35], we take the approach further by allowing users to access all conceptually related functions from an anchor point. For instance, a notification icon in the UI can be used as the anchor to change not only *how* popup notifications (emanating from the icon) are displayed, but also the *frequency* of various notifications (e.g., email), and other related settings. Anchored Customization leverages *mental associations* that users can form intuitively between visual elements of the interface and changes they want to make to their software.

Once settings are anchored to UI elements, there are different possible ways of visualizing and accessing them. We designed, implemented, and evaluated Customization Layer, as a concrete instantiation of the Anchored Customization approach. Customization opportunities are available in a meta-layer on top of the interface. Users can see all UI anchors at once in the customization layer, and can access the settings associated with each, by clicking on each anchor. This type of layered design has been successfully used in other contexts, such as online help [7]. Although only a small subset of settings is initially available from any given anchor, the user can expand a subset with ease to see the full default settings panel.

Our approach is pragmatic: it recognizes that settings panels are the standard in the software industry. Developers rely on this simple mechanism to provide end-user customization, since these panels are (presumably) an inexpensive part of the UI to develop. And users are accustomed to them. Introducing a new customization mechanism therefore requires careful consideration of the cost/benefit tradeoffs at play. We show that anchoring settings to the UI through a customization layer has the potential to provide significant benefits for users, while requiring only minimal extra work from the designers/developers. Our goal is to propose a well-considered interface improvement that can be adopted widely.

A customization mechanism must be implemented to customize a specific app. Although our design approach is applicable to any software with a GUI that is customizable via a settings panel, we had to choose a particular application domain for a first evaluation. Since task management tools are widely used, and significant individual differences have been observed in the way people manage their tasks [20], we considered that Personal Tasks Management (PTM) would be an appropriate first application domain for our research.

Our work contributes the following: (1) The concept of Anchored Customization: to anchor customization opportunities to conceptually related elements in the application UI. (2) A systematic review of PTM apps to evaluate a priori the feasibility and potential usefulness of adopting this concept. (3) A realization of this concept in a customization mechanism, the Customization Layer, with three variants that explore different trade-offs between multi-layered interfaces. (4) An implementation architecture of these three variants for web applications: our prototype augments a real-world PTM app, Wunderlist [19], but can be adapted to other modern web apps with minimal additional work for developers and designers. (5) Two experiments (Mechanical Turk and face-to-face) with 60 participants showing performance improvements of the Customization Layer over the regular settings panel provided by Wunderlist, as well as providing a preliminary understanding of how users apprehend Anchored Customization.

RELATED WORK

What exactly constitutes customization is not well-established in the HCI community, despite there being a rich literature in this space. For example, end-user customization has been defined as "the mechanisms by which users may specify individual preferences, and preserve their preferred patterns of use, without writing code" [25], but the requirement to be code-free is not universally adopted. Regardless, previous research has studied the benefits of letting users adapt software to their own preferences and tasks [25, 26, 28]. The challenges of designing and building *adaptable interfaces* has produced a rich and varied literature. Multi-layered interfaces [30, 28] can speed up access to the most often-used features by grouping them into a dedicated layer. Sophisticated techniques have been proposed to let users modify GUIs at run time [11, 10], even without access to the source code [33, 9, 29]. The end-user development paradigm goes even further, by empowering users to build their own applications [24, 15]. Yet, powerful adaptable approaches generally require users to spend significant time and effort personalizing their interfaces, often by writing scripts or editing code—which limits the potential audience to power users. As a result, these approaches have not yet been widely adopted in industry.

Adaptive approaches attempt to address that issue by automatically generating user models to predict potentially useful customizations [14]. However, giving control of the user interface to an automated system has significant drawbacks, as it tends to make the UI spatially unstable and unpredictable [12]. *Mixed-initiative approaches* can alleviate those problems by suggesting possible customizations to the user, who remains in control of their interface [8, 6]. Ultimately, adaptive and mixed-initiative approaches put the onus on developers to build and integrate complex user models into the interface, which is difficult to do well [21].

Our work focuses on leveraging the customization opportunities already available in existing software settings panels, with the goal of minimizing the extra costs of customization for both users and developers. The potential of "representing tailoring functions in the overall interface" was identified 15 years ago [23], but has received little attention since then. An early work proposed the concept of Direct Activation [35] which states that "the access point of the tailoring function should be designed related to the one of the tailorable function". In other words, the settings that affect a given feature should be accessible from (or around) the access point of the feature they affect. We broaden this approach by anchoring any setting to any conceptually related elements of the UI —not only elements objectively affected by a setting. For instance, the setting for changing the keyboard shortcut used to "open a new tab" could be anchored to the button that opens a new tab. This setting does not tailor the "open a new tab" function itself, but conceptually it makes sense to associate a keyboard shortcut and a button that call the same function.

The benefits of preserving *interface context* have been exploited in other areas of HCI. Perhaps the most common example is the context menu, which reduces the time needed to select options related to the current location of the pointer. Contextual help is another successful example: the application interface can be augmented with links to a wiki [32], user-generated Q&As [7], or information on the current state [18]. Yet customization presents unique challenges that distinguishes it from contextual help. While help queries tend to naturally correspond to elements of the interface that users are trying to learn, the correspondence between settings and interface elements may be less clear, as settings can affect software in arbitrary ways. Thus, it was not obvious at the outset of our research that such direct correspondences could be found from settings to UI elements. The literature offers promising examples of representing *meta-information* about an application as an overlay on top of the app interface: recent changes [2, 4], computational wear [27], predicted use [16], even low-level usability problems [34]. This motivated our visual design approach, which represents customization opportunities as a meta-layer on top of the interface.

SYSTEMATIC ASSESSMENT OF FEASIBILITY

The motivation of Anchored Customization is to address the shortcomings of settings panels, which appear to be the stan-

Application	Platform	# settings	directly	indirectly	not
Astrid	Android	41	25	6	10
Evernote	Desktop	31	11	12	8
Gmail	Web	29	16	7	6
RTM	Android	26	10	11	5
Toodledo	Web	53	37	7	9
Toodledo	Android	43	26	16	1
Wunderlist	Web	41	12	28	1
Word	Desktop	132	90	24	18
total		396	227	111	58
normalized %		100%	52%	32%	16%

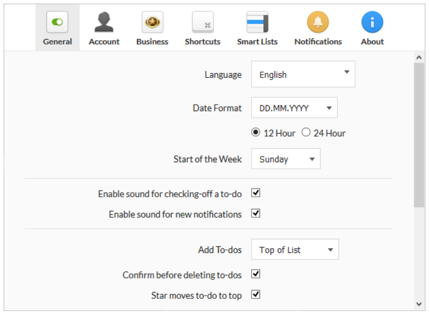
Table 1. Summary of a systematic review of 8 PTM apps to determine the feasibility of Anchored Customization. For each app, the settings provided in the settings panel were classified as directly, indirectly, or not anchorable.

dard in the software industry. To verify this assumption, we performed a systematic review of a set of PTM apps, investigating which customization mechanisms they provide. We then conducted a second review to evaluate the feasibility of the Anchored Customization approach.

Existing customization mechanisms. We chose a particular yet fairly generic application domain: Personal Task Management. We selected 12 of the most popular PTM apps[1], for a total of 20 unique apps (counting separately both desktop and Android versions). For each application we recorded which customization mechanisms were offered, how many user actions were needed to access them, and the type and number of settings each mechanism could change. Our review showed that an overwhelming majority of settings were accessed via settings panels. In fact, only 1 of the 20 applications didn't offer a settings panel at review time—but it now does. The *structure* of these panels was remarkably similar: tabs for desktops apps, multi-level menus on Android. In both cases, additional visual delimiters such as lines or boxes were sometimes used inside each tab or level, to create subsections of related settings. The same widgets were used to change settings: checkboxes or toggles for binary settings, drop-downs or radio buttons for multiple choices, and in some cases sliders for numbers. We found that settings panels also contain a variety of non-settings items, such as an "About" page.

Feasibility of Anchored Customization. The widespread use of settings panels highlights the potential *usefulness* of Anchored Customization. We next evaluated the *feasibility* of this approach on a subset of apps from our first review. We selected apps with more than 15 settings, and included Microsoft Word and Gmail which, while being general-purpose software, can both also be used for task management. As shown in Table 1, according to our estimation, 52% of settings could be anchored easily in the interface. A large number of these correspond to settings that directly show or hide interface elements, or change their position or visual appearance. Another 32% of settings could be *indirectly* anchored based on an intuitive mental association. For instance, the frequency of automatic syncing could be anchored to the button that performs a manual synchronization. This button could also be used as an anchor for setting the keyboard shortcut

[1]Evernote, OneNote, Any.do, Cal, Wunderlist, Todoist, Remember The Milk, Toodledoo, Astrid, Clear, GTasks, and Tasks.



Language: English
Date Format: DD.MM.YYYY
[✓ 12 Hour] 24 Hour
Start of the Week: Sunday

Enable sound for checking-off a to-do ✓
Enable sound for new notifications ✓

Add To-dos: Top of List
Confirm before deleting to-dos ✓
Star moves to-do to top ✓

Figure 1. The settings panel offered in Wunderlist, which was used as the Control condition in Experiments 1 and 2.

that triggers the same function (manual synchronization). Finally, we estimated that only 16% of settings could not be anchored anywhere in a meaningful way. These settings correspond either to advanced options (e.g., number of weeks after which completed tasks are archived), or global options (e.g., display language) which are not related to any particular UI location.

Overall, the results of our second review provide solid evidence for the viability of our Anchored Customization approach, at least in the PTM domain: approximately 84% of settings can be anchored. Yet a significant number of settings cannot be anchored (16%), which must be taken into account when designing anchored customization mechanisms.

DESIGN

In most applications, settings are a list of parameters that can take some predefined values. They are generally given a human readable label, and sometimes a short description. For example, a "confirm before deleting" setting can be either true or false, and could have the description "show ok/cancel popup when clicking on the delete button". UNIX-type config files are the most barebone representation of settings, and the closest to the programming domain. Users can manually change settings by editing the config file in a text editor.

Settings panels offer two key improvements over config files. First, they prevent errors by restricting the values that a given setting can take. For instance, checkboxes only toggle booleans, and drop-downs offer a limited set of options to choose from. Second, settings panels often group related settings together into tabs or other forms of subsections, with the aim to reduce the time needed to access a particular setting. In this way, settings panels are a *static abstract partition* of settings: each setting appears in only one section; sections are based on abstract categories, such as "shortcuts" or "display;" and these categories are set by designers at the outset. An example is shown in Figure 1.

Anchored Customization

The Anchored Customization approach, by contrast, promotes a *contextual many-to-many mapping* of settings to UI elements. Any UI element can be used as an *anchor*—for instance, icons, buttons, menus, even an empty area. The goal of the mapping is to provide context for each setting. One setting can be mapped to multiple anchors, and multiple settings can be mapped to the same anchor. The visibility of anchors and thus their associated settings changes dynamically, depending on the current state of the interface. The Anchored Customization approach leverages users' pre-existing knowledge of the application UI, instead of requiring them to learn the abstract structure of a settings panel. Hence, the key idea of Anchored Customization is that the application interface itself is used to organize and navigate the settings space.

There are two main dimensions in the design space for mechanisms that instantiate Anchored Customization: the display of the settings and the display of the anchors.

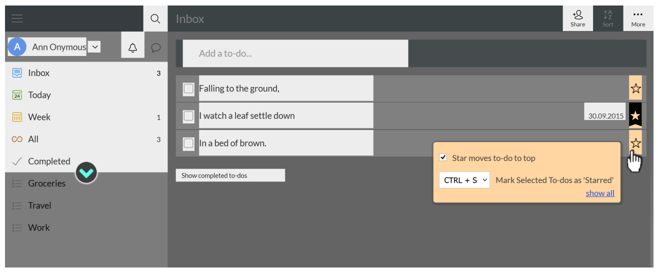
Display of settings. It is not desirable to permanently show the settings associated with the various anchors. Most applications offer many settings, which would clutter the interface if they were always visible. Further, customization is typically very much a secondary infrequent activity. Thus, in general, the settings associated with an anchor should be displayed *on demand*, when the user expresses interest in an anchor—by clicking or hovering on it, for instance. There are many ways to display settings once demanded. We explored three possibilities, inspired by multi-layered interfaces (described below). But the design space is much larger. Some interesting dimensions include: *Which* settings should be displayed when clicking on an anchor. (Only the settings mapped to this anchor, or a local neighborhood?) How to *represent* each setting. (Now that settings are placed in context, could the text labels be shortened or replaced by an icon?)

Display of anchors. There are a number of different possibilities when considering how the anchors can be displayed. One familiar approach is through context menus: anchors are not explicitly marked visually, but can be used by the user to demand a corresponding setting. For instance, in Microsoft Word, right-clicking the ribbon shows a contextual menu with an option to "Customize the ribbon". A tooltip that appears when hovering over an anchor for a short duration could be used in a similar way. This local, targeted approach answers the question: Is *this* customizable? The downside of this approach, however, is that there is no way for the user to get a holistic overview of all the settings available within the UI. Serially invoking the context menu from many different anchors is too tedious, and the user must resort to bringing up the settings panel.

An alternative approach is to provide a *global overview* of all the customization opportunities available, by making all the anchors visible at once, likely through a mode. This approach helps answer the question: *What* is customizable? The downside of the approach is that entering a distinct mode could interrupt the user's workflow, which is why modes should be used sparingly in interaction design. The contextual menu approach and the global approaches are not mutually exclusive; providing both would allow users to choose the one most appropriate for their current need.

Customization Layer

We designed and implemented Customization Layer which instantiates the anchored customization approach. In the design dimensions described above, this mechanism uses a *global overview* approach to show all anchors in an extra



☑ Star moves to-do to top

CTRL + S Mark Selected To-dos as 'Starred'

show all

Figure 2. Customization Layer displayed on top of Wunderlist. The user is currently hovering over the bottom "star" button on the right-hand side of the screen; as a result, all star buttons are highlighted in orange because they share the same settings. Clicking on an anchor displays the settings associated with it—here, shown in the Minimal panel.

layer on top of the interface. We explore three ways to display settings on demand, inspired by multi-layered interfaces.

Anchors visible in a layer

To avoid clutter, we did not add graphical elements to visually mark anchors (contrary to the question marks displayed in [7] to represent help queries). Instead, anchors are visually highlighted when users activate a *layer* on top of the regular application. In our implementation, the app interface becomes darker and less saturated, but is still clearly visible through the Customization Layer. Anchors are shown with a white background and dark gray text, to optimize legibility and ensure visibility above the dimmed application interface (Figure 2).

When the user hovers over an anchor A, that anchor and all other anchors mapped to the same settings as A are highlighted in orange (Figure 2). This linked highlighting helps users create a mental model of the mapping of settings to UI elements. For instance, in a PTM app, hovering over a button highlights all the buttons that share the same setting (e.g., a keyboard shortcut for marking a "todo" as important).

There is one important special case that required attention when anchoring settings to UI elements: what if a setting governs the visibility of *the UI anchor itself?* Consider for instance settings that show or hide buttons in a toolbar. Of course it makes sense to anchor such a setting to the button it affects; but after hiding the button, how can users access its associated setting to make the button visible again? To avoid this problem, we introduce the notion of *ghost anchors:* anchors that correspond to hidden elements in the application UI. These anchors offer the same functionality as regular anchors, but are displayed in a darker gray to indicate their transient status.

Representing all ghost anchors visually is not necessarily desirable: if an application has many optional, hideable components, showing all of them as ghost anchors could break the application layout, clutter the customization layer, and overwhelm the user. Furthermore, the customization layer should look as similar as possible to the current interface of the application, to help users alternate seamlessly between the two. For these reasons, we implemented a collapsing mechanism: ghost anchors are by default represented by a simple "chevron" icon in the customization layer (Figure 3). To further reduce visual clutter, ghost anchors that are close to each

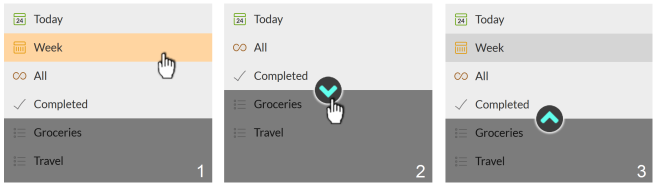


Figure 3. (1) The user selects the "Week" filter, then disables it (not shown). (2) The corresponding anchor is collapsed under a chevron icon. (3) Clicking on the chevron reveals the "Week" filter as a ghost anchor.

other are collapsed under the same chevron icon, based on a simple proximity clustering algorithm. Clicking on a chevron icon reveals the ghosts that it was previously hiding.

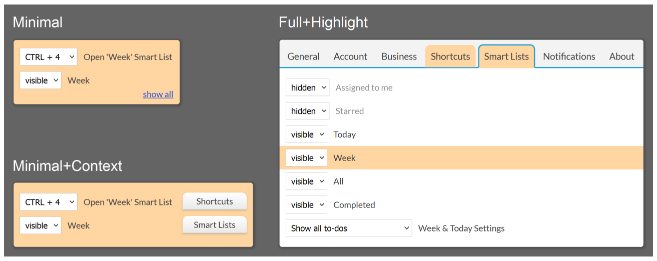
Three variants for displaying settings

When the Customization Layer is shown, clicking on an anchor displays the subset of settings anchored to that UI element. We explored three visual representations of this subset (see Figure 4). The Minimal panel (M) only shows the settings in the subset, with an orange background to indicate that they are related to the anchor that was clicked. Minimal+Context (M+C) is very similar, but a tab name next to each setting indicates which tabbed pane in the full settings panel it comes from. In contrast, Full+Highlight (F+H) keeps the structure of the complete settings panel, and the settings from the anchor's subset are highlighted in orange, as are the tabs in which they appear. Some tabs contain more settings than can be shown in the height of the panel, so we automatically scroll down to bring into view the first highlighted setting, if any.

Because some settings might not be anchorable to the interface (cf. Feasibility section), all designs must provide a fallback access to the complete set of settings. In F+H, users can browse all the tabs freely, as they would in a regular settings panel. In M+C, the tab names next to each setting are actual buttons; clicking on them opens the full panel at the corresponding tab. In Minimal, a "show all" hyperlink is provided at the bottom of the mini panel. By default, it opens the full panel at the tab that contains the most settings from the anchor's subset. Users can return to the minimal panel by clicking a backward arrow at the top left of the full panel.

These three variants reflect different points in the multi-layered interface design space [30]. The two minimal variants only show the anchor's subset of the settings initially, while F+H uses visual highlighting to distinguish the subset from all other settings. All three designs are intended (to varying degrees) to reduce complexity, speed up access to the most relevant settings, and help the user become aware of the full set of settings. Hence, they reflect different trade-offs between a minimal subset of settings and the full settings panel.

As with any multi-layered interface, transitioning from a lower layer to an upper layer can be challenging for users [30]. In our implementation, we provide animated transitions between the minimal variants and the full panel to help users understand the relationship between the two. The minimal panels expand smoothly to the full size, while the highlighted settings glide into their new position. The remaining (non-highlighted) settings fade in afterwards. Informal pilots found these relatively simple animations to be helpful for convey-



CTRL + 4 Open 'Week' Smart List
visible Week

CTRL + 4 Open 'Week' Smart List
visible Week

hidden Assigned to me
hidden Starred
visible Today
visible Week
visible All
visible Completed
Show all to-dos Week & Today Settings

Figure 4. Minimal, Minimal+Context, and Full+Highlight variants showing the settings anchored to the "Week" Smart List. Wunderlist offers several such Smart Lists, which display a list of all todos filtered by a particular criteria—here, all the todos due this week. Two settings are mapped to this Smart List: the first changes the shortcut for opening this Smart List; the second determines whether this Smart List is visible or hidden in the main interface.

ing the intended mental model. We are not aware of another multi-layer interface that uses moderately complex animated transitions like ours, instead of simply fading-in the new elements

Search. Text search is another effective approach for accessing settings. It essentially provides a way to "jump" to a particular setting without having to locate an anchor or browse the settings space. Search is afforded by traditional config files, but rarely offered in settings panels—except in complex software such as Eclipse. The downside with search is that the vocabulary problem [17] applies, as users can only guess search terms [3]. Furthermore, relying on search may hinder users' ability to learn other settings. Although not yet implemented, Customization Layer is compatible with text search: UI anchors could be filtered out if their associated settings don't match the query, or visually highlighted if they do.

Software architecture. Our software architecture was designed to be app-independent and extensible to other web apps. The Evaluation section explains why we chose web apps, and Wunderlist in particular, as a concrete starting point for design and evaluation. Adding our Customization Layer to a web app requires: (1) an API to read and write the settings, and (2) a mapping between settings and visual elements of the interface, provided by the designers of the app. Since designers are already familiar with the settings of their app, creating the mapping should take at most a few hours, except for very large applications. In the mapping, UI elements are represented by CSS selectors, which allows among other things selecting many similar anchors via a single CSS class.

Implementation. Since Wunderlist doesn't provide an API to its settings, we reverse-engineered its front-end to access the underlying settings directly. We then manually mapped these settings to appropriate elements of the interface, which took two hours. From this mapping, our code automatically generates anchors by creating copies of the DOM elements matched by the CSS selectors provided. Anchors are then positioned precisely on top of the original element, creating the illusion that the elements themselves are highlighted. We were able to access Wunderlist's settings directly on the web client, effectively bypassing the settings panel to customize the app in real time in our experiments.

EVALUATION

We focus our evaluation on assessing the usability of our Customization Layer. Given the novelty of this new mechanism, our two experiments were intended to be exploratory and provide an idea of the impact of some of its design elements. We certainly hoped that Customization Layer would help participants find settings faster than browsing a traditional settings panel; but which of the three variants would be the fastest or most preferred was unknown. Further, we anticipated that the three variants would impact users' awareness of the full set of settings differently: the more contextual information a design provides, the greater the user's awareness should be [13].

To maximize the ecological validity of our evaluation, we decided to evaluate Customization Layer within an existing application with its actual settings. We chose Wunderlist [19], one of the most popular PTM apps today (over 10 million downloads on Android and iPhone) which offers a well-designed web interface. Of the 20 apps from our systematic review, Wunderlist had a particularly clean settings panel, with numerous and varied settings. It therefore appeared to be a good baseline for a fair comparison.

Experiment 1: Remote Mechanical Turk

The primary goal of this experiment was to evaluate the performance of our Customization Layer for changing settings, compared to a traditional settings panel. We used a between-subject design to avoid negative carry-over effects, because switching back-and-forth between two very different mental models could have been confusing for participants. We deployed this experiment on Amazon Mechanical Turk.

Participants. All 48 participants (aged 19-60, median 28.5, 16 females) were regular computer users, and none had tried Wunderlist before. We had replaced 3 participants, who were either 2.5 Inter-Quartile Range slower than others in their condition, or did 2 IQR more errors than everyone else.

Task. The experiment consisted of a sequence of settings changes. At the beginning of each trial, a popup instructed participants to change one setting to a given value. Pressing a "Go!" button would close the popup and start the timer. The instructions were written in layman's terms, and did not necessarily use the same words as the label of the target setting. For example, the instruction "Change shortcut for checking off todos" applied to the setting "Mark selected do-dos as completed" in the Shortcuts tab. To help ensure that participants read the instructions before starting the trial, the "Go!" button was kept inactive for the amount of time required to read the instructions at an average reading speed. During that time, the settings panel (in Control) or the UI anchors (in Customization Layer) were hidden, to prevent participants from planning their actions before the timer was started. During the trial itself the instructions were visible at the top of the window, in case participants had forgotten them.

Participants had to change at least one setting before the "Next" button became available in the bar at the top of the screen. Clicking this button would stop the timer, indicate whether the task had been successfully completed, and move on to the next trial. We enforced a 2-minute time limit for each trial, after which the trial was marked as a timeout and participants were taken to the next trial. If participants made mistakes, the settings that had been changed incorrectly were reset at the end of the trial, and the target setting changed to the correct value.

Measures. The duration of a trial was measured between the time when participants pressed the "Go!" button and when they changed a setting. At the end of the experiment, participants completed two recognition questionnaires in order to evaluate awareness of the full set of settings: one questionnaire on tab names, one on individual settings. In both cases, half of the answers were made up by the authors, but plausible. Finally, participants were asked to rate their satisfaction on a 7-point Likert scale: the extent to which they liked or disliked the customization mechanism they were using, and how easy it was to find the settings they were looking for.

Conditions. We compared four customization mechanisms: the three variants of Customization Layer (M, M+C, F+H) and the actual settings panel offered by Wunderlist (Control), shown in Figure 1. While we accurately reproduced the structure of Wunderlist's panel in our F+H prototype, the visual style was slightly different: Wunderlist generally looks more polished, with custom drop-downs and an icon on each tab.

Wunderlist currently offers 41 settings in four tabs[2]. Out of these settings, we discarded the application language one, as changing it would make the interface indecipherable for our participants. The experiment tasks were not equally difficult: some settings were indeed harder to find than others, especially if their text label was unclear. To reduce the variability between subjects, we partitioned the settings into two groups of 20, carefully chosen to balance the type and location of the settings they offered. Each participant was assigned one set. To assess any early learning, participants were asked to change the same group of 20 settings twice, but in a different randomized order for each of the blocks of 20.

Design. A 2-factor mixed design was used: 4 customization mechanisms (M, M+C, F+H, Control; between-subjects) x 2 blocks (within-subject). The settings group control variable (group 1, group 2) was fully counterbalanced between participants. Each participant completed 2 blocks of 20 trials, for a total of 1920 trials across all 48 participants.

Procedure. After accepting the HIT (work assignment in MTurk), participants were asked to create a temporary account on Wunderlist, using a randomly-generated email address. Then participants had to drag a custom bookmarklet (a "bookmark applet") to their bookmarks bar, and to click on it to load the experiment code in the Wunderlist webapp. A series of popups walked participants through a quick tutorial, demonstrating the most useful features of Wunderlist. Participants were then given one practice trial, before starting any trial. In the 3 Customization Layer conditions, the practice trial contained a few additional instructions on how to use this new customization mechanism. Finally, participants were asked to complete the two recognition tests, to rate the

[2]Three other tabs show non-settings information, such as "upgrade your account" and "about this product." We did include these tabs in our prototype.

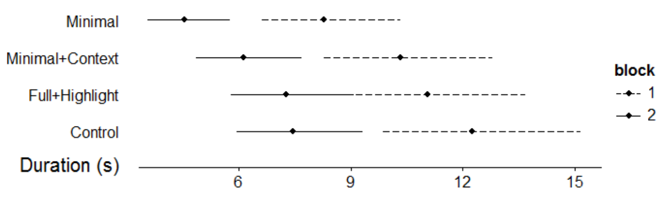


Figure 5. Experiment 1: 95% confidence intervals of the median duration per Customization Mechanism and Block. M is significantly faster than F+H and Control, but not significantly different from M+C. (*N*=48)

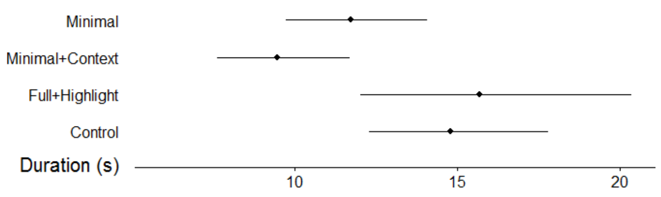


Figure 6. Experiment 2: 95% confidence intervals of the median duration per Customization Mechanism. M+C is significantly faster than F+H and Control, but not significantly different from M. (*N*=12)

customization mechanism they were using, and to complete a simple demographics questionnaire. The whole procedure lasted 32 minutes on average, and participants received an average compensation of \$4.34, depending on their performance in the trials and recognition questionnaires. (Incentivizing participants with a variable bonus reward is a standard practice for MTurk studies.)

MTurk Results

We ran a mixed-design ANOVA on the duration of trials. The data was log-transformed to satisfy the assumption of normality, and we used medians to reduce the influence of outliers. The effect sizes reported are generalized eta-square (η^2_G), interpreted as follows: .02, .13, .26 for a small, medium, and large effect size respectively [1]. As shown on Figure 5, there was a main effect of customization mechanism ($F_{3,44} = 3.58$, $p < .05$, $\eta^2_G = 0.17$), as well as block ($F_{1,44} = 124.63$, $p < .001$, $\eta^2_G = 0.32$), but no interaction between the two. M was significantly faster than Control ($p < .05$, 35% faster) and F+H ($p < .05$, 30% faster). No other pair was significantly different. All pairwise comparisons use a Bonferroni correction unless otherwise mentioned. As anticipated, there was no effect of the group of settings used. There were few timeouts and errors across all conditions (20 and 54, respectively, for 1920 trials), and they had no significant effect on our results.

A single-factor ANOVA on the tabs recognition scores found a significant effect of customization mechanism ($F_{3,44} = 5.82$, $p < .01$, $\eta^2_G = 0.28$). M scored lower than both F+H ($p = .05$) and Control ($p < .05$), by 2 out of 10 points lower on average. No other pair was significantly different. There was no difference either in the recognition scores for settings, nor in the subjective ratings provided at the end of the experiment. The median rating for ease of use and satisfaction was the same for each of the four customization mechanisms: 2 on a scale ranging from -3 to +3.

Experiment 2: Face-to-Face in the Lab

Although MTurk provided quick access to a diverse set of participants at a moderate cost, conducting remote experiments has limitations in terms of the insights and qualitative feedback that can be gathered. Thus, we ran a second experiment in a face-to-face lab setting. This time the customization mechanism was treated as a within-subject factor, to allow participants to make informed comparative judgments.

Method differences from Study 1. We recruited 12 participants (age 21-42, median 24.5, 5 females), all regular computer users. Two had tried Wunderlist before, but were not using it regularly. The experiment task was identical, except that participants were encouraged to talk aloud.

A single-factor within-subject design was used, with the same four customization mechanisms. Participants had to change 10 different settings with each mechanism, using all the 40 settings available in Wunderlist, for a total of 12 x 40 = 480 trials. The three Customization Layer variants were blocked together, to limit the number of times participants had to switch between the two mental models. Half of the participants began with the Customization Layer block, the other half with the Control condition. Within the Customization Layer block, the presentation order of the three variants was fully counterbalanced. The settings were randomly ordered across all blocks.

The experiment procedure was similar to the MTurk study, except for a few changes reflecting the within-subject design. After each mechanism condition, participants were asked to *rate* on a 7-point Likert scale the mechanism on three metrics: ease of use, perceived speed, and satisfaction. At the end of the experiment, participants were asked to *rank* the four mechanisms on the same metrics. No recognition questionnaires were administered, since it would have been difficult to tease apart the learning that happened in each condition. The experiment was concluded by a brief semi-structured interview. Participants were asked about their perception of the four different customization mechanisms, and we gathered insights on how they developed a mental model of Anchored Customization.

Face-to-Face Lab Results

We present the quantitative and qualitative results.

Quantitative results

We ran a repeated measures ANOVA on the duration of trials. As in Experiment 1, the data was log-transformed and we used medians. As shown on Figure 6, there was a main effect of customization mechanism ($F_{3,33} = 5.61$, $p < .01$, $\eta^2_G = 0.16$). M+C was faster than Control ($p < .05$, 36% faster) and F+H ($p = .06$, 40% faster). No other pair was significantly different. The order in which participants saw the conditions (Control first, or Customization Layer first) had no significant effect either.

The subjective ratings collected after each block were analyzed with a Friedman test. No differences were found for satisfaction and ease of use. Perceived speed was borderline significant ($\chi^2_3 = 7.1$, $p = .07$), with the majority of the difference coming from M+C reported to be faster than F+H ($p = .1$). In terms of the ranking data, participants were almost evenly divided between Customization Layer (6 top ranks) and Control (7) —one participant ranked two mechanisms as best. There was also no clear consensus among which of the three CL variants was best: with 1, 3, and 2 votes going to M, M+C, and F+H respectively.

Qualitative results

The comments made by participants during the experiment and their answers during the interview were recorded. We report the findings of a thematic analysis [5].

Anchored Customization. Nine out of 12 participants acquired the intended mental model of Anchored Customization: 6 acquired it immediately with the first variant of Customization Layer they encountered, the other 3 with the second variant. Once participants understood the concept of Anchored Customization, they were able to use it successfully: *"I'm used to all the settings hidden away in a menu, but I think this (Minimal) makes a lot of sense"* (P5), *"I think the point of this (F+H) is that I don't need to think under which category each setting is"* (P3). In some cases, participants even imagined appropriate anchors that were not actually present in Wunderlist, which shows a clear grasp of the Anchor Customization Concept. For instance, P1 said: *"I was looking for a printing icon"*, while searching for the setting "don't print completed todos when printing a list".

Three out of 12 participants did not seem to have acquired the intended mental model: in more than 50% of trials, they clicked on any anchor, from which they immediately opened the full panel and browsed the tabs to find the desired setting. These 3 participants were among the 6 who started the experiment in the Control condition, which could have negatively affected their behavior in the subsequent Customization Layer conditions. The only time these participants searched for an appropriate anchor (rather than just any anchor) was when the target setting was related to one of the Smart Lists, in the left sidebar (see Figure 2). These 3 participants may have reasoned by analogy with their one practice trial, which instructed them to change a (different) Smart Lists setting.

Customization Layer variants. Interestingly, at least 3 participants did not notice any difference between M and M+C during the trials, and only realized that they were different at the end when they were asked to rank the four customization mechanisms. Some participants liked the extra structure provided by the tab names in M+C: *"In Minimal I don't know where I am in the greater structure. [M+C] is more organized" (P1)*; *"The links in [M+C] are clearer, you know where you're going"* (P8).

In F+H, at least 4 participants did not see the orange highlighting, although it was clearly visible (Figure 4). This could be a case of inattentional blindness [31]: participants who hadn't yet acquired the intended mental model may have discarded the highlighting because it had no meaning to them. Other participants thought of the highlighting as a soft suggestion from the system: *"At one point I assumed the highlighting was like a hint"*(P8), *"[the system] is saying: I think you are looking for this, but I'm not sure"* (P12).

Wunderlist. Although it was not an explicit goal, this experiment revealed some usability problems with Wunderlist's settings panel, which was used as the Control condition. The most problematic aspect is that 10 settings are hidden under

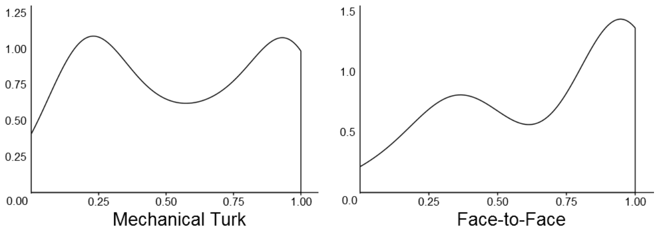


Figure 7. Density plots of the distribution of the percentage of trials in which participants selected a "correct" anchor—one associated with the target setting. Participants in the right mode followed an "anchor search" strategy, while participants in the left mode resorted to a "full panel search".

a "show more" button in the Shortcuts tab. This design decision was probably intended to avoid showing all 20 shortcuts settings at once, by hiding the ones less likely to be changed. However, this "show more" button was not salient enough, and 10/12 participants missed it the first time they looked for one of the settings it was hiding. Interestingly, this problem was naturally alleviated in all Customization Layer variants: in both M and M+C, only the relevant settings are shown when clicking on an anchor, so there is no need for a hiding mechanism. F+H automatically unhides all of the 10 more advanced shortcuts if one of them is highlighted, and scrolls down to the first highlighted setting.

Visual design. Our primary concern while designing the Customization Layer was interaction design, not aesthetics. However, as is often the case in HCI experiments, some participants tended to focus on the visual appearance of the different interfaces more than their behavior: 4/12 participants indicated that they liked the look of the Wunderlist settings panel better. In particular, the presence of an icon on each tab was appreciated (P9). This might explain why 7 participants ranked Control as well or above Customization Layer: as P10 puts it, *"for customers pretty is more important"*.

Secondary Exploratory Analyses

The fact that we obtained different performance results for M and M+C in our two experiments was surprising and prompted us to probe the data further. M was found to be faster than F+H and Control in the first study, whereas it was M+C that was faster than F+H and Control in the second study. Yet in both cases, the p-value and effect sizes are similar. A closer look shows that these results are not contradictory: M and M+C were never found to be significantly different from each other, nor significantly slower than F+H or Control.

In order to tease this apart, we looked at the logged data with a focus to understand participants' approach to finding the target setting. Of particular interest is whether participants found a correct anchor to click on to access the target setting (what we call "anchor search"), or if they defaulted to clicking on any anchor from which they then browsed the full settings panel ("full panel search"). For each participant, we computed the percentage of trials in which they found a correct anchor, out of the total number of trials in the experiment. The distribution of this metric across all participants is clearly bimodal, both on MTurk and in the lab (Figure 7). The posi-

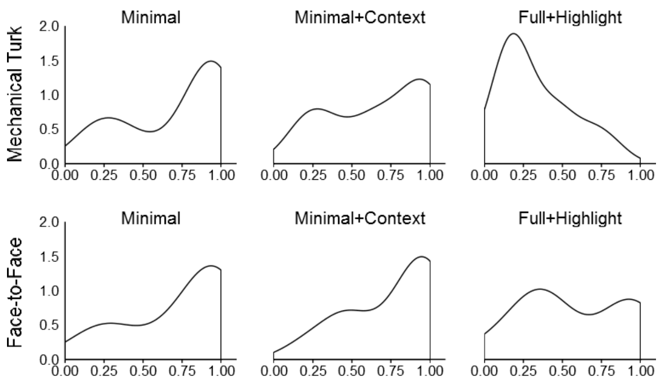


Figure 8. Density plots of the percentage of trials in which participants selected a "correct" anchor, for each Customization Layer variant, on MTurk (top) and in the lab (bottom). The minimal variants all have a higher right mode ("anchor search"), while F+H has higher left mode ("full panel search").

tion of the "valley" (local minimum) of the two distributions is also similar: 61% on MTurk, 63% in the lab.

This analysis points to a likely hidden variable in our data: the search strategy used by participants. Indeed the three Customization Layer conditions vary along this extra dimension, as shown in Figure 8. While these variations might explain the difference between the two minimal variants and F+H, the distributions for M and M+C are too similar to explain the difference in results observed between the two experiments.

For the next step in our exploratory analysis, we looked only at the data from participants who adopted an "anchor search" strategy to see if we could see any differences between M and M+C. More specifically, we re-ran the ANOVA on trial duration only on the data that fell above the middle point. For the MTurk experiment, M and M+C were both faster than Control ($p < .001$ each), and not significantly different from each other. For the lab experiment, M and M+C were also both faster than both F+H and Control (all $p < .05$). This secondary analysis suggests that the results of MTurk and the face-to-face experiments may be consistent, as long as you take search strategy into account: M and M+C are *both* faster than Control, and not significantly different from each other. As with any non-planned analysis, these results need to be interpreted with caution.

DISCUSSION

The results of our experiments are promising: whether on MTurk or in the lab, one of the minimal variants was significantly faster than Control, with a medium effect size, and no variant was slower. These performance improvements were obtained even though participants were exposed to Anchored Customization for the first time, and received little information to help them build an appropriate mental model. We now discuss the insights gathered on the three Customization Layer variants, and reflect on the possibilities offered by this new customization mechanism.

Reconciling the performance results. M and M+C performed differently in the two experiments. Our secondary analyses revealed a likely hidden variable, namely search strategy. For participants using the anchor search strategy, the results of the two experiments are consistent: M and M+C are *both* faster than Control, and not significantly different from each other. Hence there must be a significant difference in how the "full panel search" participants performed in the two experiments which translated into different relative performances for M and M+C on MTurk and in the lab. It might just be a statistical fluke. For instance, looking at the data shows that on MTurk all 5 of of the M+C participants that did a "full panel search" were disproportionately slower than the "full panel search" participants in *all* other conditions. This inconsistency suggests that the randomization of participant assignment to conditions may not have equalized individual differences. More research will be needed to further assess this issue.

Few participants in the F+H condition used the "anchor search" strategy: 2/12 on MTurk, 5/12 in the lab. It could be that highlighting settings in a panel is not a strong enough cue to help users acquire the Anchored Customization mental model. Since the structure of the settings panels is retained in F+H, there may be a strong transfer effect that encourages users to default to the "full panel search", instead of exploring anchors. Since few participants used F+H the way it was intended, we cannot conclude with certainty on its potential performance: is F+H necessarily slower than M and M+C, or can it be as fast when properly used? In any case, F+H is not slower than Control, so it would not be detrimental. Since some participants perceived the highlighting as a hint, F+H could be used as a "softer" version of Customization Layer for users who might not be comfortable with the degree of minimalism of the two minimal variants.

Performance/Awareness tradeoff. On MTurk, M was significantly faster than F+H and Control, but it also scored significantly lower on recognition of tab names. This could be interpreted as a performance/awareness tradeoff found in other multi-layered interfaces [13]. Yet, users in M did just as well as others in terms of recognizing the settings themselves. Thus, the awareness tradeoff here seems to affect only awareness of the *structure* of the upper layer (the full settings panel), not its content, as there were no differences in recognition scores for the settings themselves. This is not entirely surprising: in Customization Layer, all the settings are accessible via the minimal panel, albeit from different anchors. By contrast, the personalized interfaces studied by Findlater *et al.* [13] only display a static subset of features in the first layer, while the others were only visible in a different layer.

Applicability to other software. We focused our work on task management applications, and the question remains whether Anchored Customization could provide similar benefits for other types of software. At one extreme is text editors, which are often highly customizable. These editors typically have very few always-visible UI elements, and rely mostly on menus and keyboard shortcuts. Thus, they are not well-suited for Anchored Customization. Furthermore, text editor users are generally comfortable customizing their software by editing the config files directly. The opposite extreme is complex software applications designed for non-technical users, such as Adobe Creative Suite. These applications have many widgets that provide access to lots of features. Anchoring settings to these various UI elements is possible, but the resulting Customization Layer may be overwhelming. The main interface of these applications can itself be overwhelming, however, especially for new users. Anchored Customization would simply reflect the complexity of the underlying software.

Applicability to handheld devices. Our systematic review showed that mobile apps organize their settings in very similar ways to desktop applications. Mobile apps generally rely on icons and buttons for user input, since keyboard shortcuts and extensive menus are not available. As such, they are well suited for Anchored Customization. The limited screen real estate would warrant using a minimal variant. The anchored customization mechanism could be activated via a standard application menu, but touchscreens offer other possibilities: for instance, users could long-press or multi-tap an anchor to see its associated settings, or use a special standardized gesture to activate the Customization Layer. We observed that mobile applications generally offer fewer settings than desktop apps. The introduction of a well-designed customization mechanism could lead to more customizable mobile apps.

The developers' point of view. Beyond its benefits to users, our Anchored Customization approach may also change the way designers and developers think about customization. Although they are not *required* to change any setting to adopt Anchored Customization, the process of mapping settings to UI elements could have a positive effect on the settings offered. For instance, designers might realize that some parts of the interface have no setting anchored to them, which would highlight a potential opportunity for providing settings that cater to this area. Creating and labeling settings may also become faster, since the problem of finding appropriate words to refer to interface elements is mitigated by the context provided by the anchors. Finding categories to organize settings into tabs might also become superfluous.

Limitations

While our results are promising, our experiments had limitations, which come mostly from the challenges of evaluating customization in an artificial setting. We point out three. (1) To maximize ecological validity, we used Wunderlist's actual settings panel as a baseline for the Control condition. But some participants focused their feedback on its high quality visual design relative to our prototyped Customization Layer variants. In retrospect, we should have recreated this panel in the same visual style as our prototypes which might have increased the internal validity of our experiments. (2) The practice trial could have been more effective at conveying the intended model. Participants performed only one trial, thus only had to click on one anchor to complete it. However, to really understand the concept of Anchored Customization, one must click on at least two anchors to see that the settings offered are different. Some participants took time during the practice trial to explore the Customization Layer on their own, clicking on multiple anchors to see the outcome. This free exploration seemed more effective than our practice trial, and would also be more similar to real-world conditions. (3) Our experiments only included one application with a medium number of settings. It is possible that the number of settings offered by an app affects the relative performance of Customization Layer and settings panels.

We compared customization mechanisms mainly on how quickly participants could find a designated setting. However, in a real world situation, the *awareness* (or lack thereof) of which settings are available likely plays an important role. In traditional settings panels, awareness is gained by serendipitous discovery (also referred to as "incidental learning" [13]) : users happen to notice a setting of interest while searching the panel for another one. Serendipity can also happen in Anchored Customization, but the notion of proximity is relative to the anchors, instead of the settings panel's structure. Because our experiment task was time-constrained, these different forms of serendipity were not well captured.

CONCLUSIONS AND FUTURE WORK

Anchored Customization is an approach that places settings *in context* within the application interface, so that users are not required to learn the abstract structure of a settings panel. A systematic review of a set of Personal Task Management apps found that approximately 84% of the settings could be anchored in the UI. Our Customization Layer prototype reveals all the anchors as affordances for customization. We designed three variants of Customization Layer based on multi-layered interfaces, and implemented these variants on top of a popular web application for task management, Wunderlist. Two experiments (Mechanical Turk and face-to-face) showed that the two minimalist variants were 35-36% faster than Wunderlist's settings panel.

Evaluating the long term impact of this customization approach remains future work. A longitudinal field study would determine if a more usable customization mechanism does actually increases users' likelihood to customize. It would also help to verify our assumption that Anchored Customization requires one-time learning: once this approach is understood in the context of one particular app, the mental model should be transferable to other apps. Our prototype could be distributed to real users of Wunderlist as a browser extension, or adapted to other applications to compare the effect of different types of settings and different application domains.

Currently, app designers need to provide the mapping between settings and UI elements. With the growing popularity of advanced front-end frameworks in web development, code analysis techniques could possibly generate part of the mapping automatically, by determining which UI elements and listener functions are affected by a setting. Another possibility would be to involve users in the mapping process. Contrary to crowdsourced contextual help [7], users cannot be expected to *generate* the entire mapping themselves, but they could *tweak* a designer's mapping to better match their expectations. The idea of refining the mapping by aggregating data from individual users could be expanded to other customization opportunities as well. For instance, the most popular extensions and plugins could be anchored to the UI.

ACKNOWLEDGEMENT

This work was was funded by the Graphics Animation and New Media (GRAND) NCE and an NSERC Discovery grant.

REFERENCES

1. Roger Bakeman. 2005. Recommended effect size statistics for repeated measures designs. *Behavior Research Methods* 37, 3 (2005), 379–384. DOI: http://dx.doi.org/10.3758/BF03192707

2. Patrick Baudisch, Desney Tan, Maxime Collomb, Dan Robbins, 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. DOI: http://dx.doi.org/10.1145/1166253.1166280

3. Ofer Bergman, Ruth Beyth-Marom, Rafi Nachmias, Noa Gradovitch, and Steve Whittaker. 2008. Improved Search Engines and Navigation Preference in Personal Information Management. *ACM Transactions on Information Systems* 26, 4, Article 20 (Oct. 2008), 24 pages. DOI: http://dx.doi.org/10.1145/1402256.1402259

4. Anastasia Bezerianos, Pierre Dragicevic, and Ravin Balakrishnan. 2006. Mnemonic Rendering: An Image-based Approach for Exposing Hidden Changes in Dynamic Displays. In *Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology (UIST '06)*. ACM, New York, NY, USA, 159–168. DOI: http://dx.doi.org/10.1145/1166253.1166279

5. Virginia Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. *Qualitative Research in Psychology* 3, 2 (2006), 77–101. DOI: http://dx.doi.org/10.1191/1478088706qp063oa

6. Andrea Bunt, Cristina Conati, and Joanna McGrenere. 2007. Supporting Interface Customization Using a Mixed-initiative Approach. In *Proceedings of the 12th International Conference on Intelligent User Interfaces (IUI '07)*. ACM, New York, NY, USA, 92–101. DOI: http://dx.doi.org/10.1145/1216295.1216317

7. Parmit K. Chilana, Andrew J. Ko, and Jacob O. Wobbrock. 2012. LemonAid: Selection-based Crowdsourced Contextual Help for Web Applications. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, New York, NY, USA, 1549–1558. DOI: http://dx.doi.org/10.1145/2207676.2208620

8. Matjaz Debevc, Beth Meyer, Dali Donlagic, and Rajko Svecko. 1996. Design and evaluation of an adaptive icon toolbar. *User Modeling and User-Adapted Interaction* 6, 1 (1996), 1–21. DOI: http://dx.doi.org/10.1007/BF00126652

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. James R. Eagan, Michel Beaudouin-Lafon, and Wendy E. Mackay. 2011. Cracking the Cocoa Nut: User Interface Programming at Runtime. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology (UIST '11)*. ACM, New York, NY, USA, 225–234. DOI: http://dx.doi.org/10.1145/2047196.2047226

11. W. Keith Edwards, Scott E. Hudson, Joshua Marinacci, Roy Rodenstein, Thomas Rodriguez, and Ian Smith. 1997. Systematic Output Modification in a 2D User Interface Toolkit. In *Proceedings of the 10th Annual ACM Symposium on User Interface Software and Technology (UIST '97)*. ACM, New York, NY, USA, 151–158. DOI: http://dx.doi.org/10.1145/263407.263537

12. Leah Findlater and Joanna McGrenere. 2008. Impact of Screen Size on Performance, Awareness, and User Satisfaction with Adaptive Graphical User Interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08)*. ACM, New York, NY, USA, 1247–1256. DOI: http://dx.doi.org/10.1145/1357054.1357249

13. Leah Findlater and Joanna McGrenere. 2010. Beyond performance: Feature awareness in personalized interfaces. *International Journal of Human-Computer Studies* 68, 3 (2010), 121 – 137. DOI: http://dx.doi.org/10.1016/j.ijhcs.2009.10.002

14. Gerhard Fischer. 1993. *Adaptive User Interfaces*. Elsevier Science Publishers B.V., Chapter Shared Knowledge in Cooperative Problem-Solving Systems – Integrating Adaptive and Adaptable Components, 49–68. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.310.8392

15. Gerhard Fischer and Elisa Giaccardi. 2006. Meta-design: A Framework for the Future of End-User Development. In *End User Development*, Henry Lieberman, Fabio Paternò, and Volker Wulf (Eds.). Human-Computer Interaction Series, Vol. 9. Springer Netherlands, 427–457. DOI: http://dx.doi.org/10.1007/1-4020-5386-X_19

16. Stephen Fitchett, Andy Cockburn, and Carl Gutwin. 2013. Improving Navigation-based File Retrieval. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 2329–2338. DOI: http://dx.doi.org/10.1145/2470654.2481323

17. G. W. Furnas, T. K. Landauer, L. M. Gomez, and S. T. Dumais. 1987. The Vocabulary Problem in Human-system Communication. *Commun. ACM* 30, 11 (Nov. 1987), 964–971. DOI: http://dx.doi.org/10.1145/32206.32212

18. Alfonso García Frey, Gaëlle Calvary, and Sophie Dupuy-Chessa. 2010. Xplain: An Editor for Building Self-explanatory User Interfaces by Model-driven Engineering. In *Proceedings of the 2nd ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS '10)*. ACM, New York, NY, USA, 41–46. DOI: http://dx.doi.org/10.1145/1822018.1822026

19. 6 Wunderkinder GmbH. 2015. Wunderlist. Version 3.12.3. (2015). http://www.wunderlist.com (Retrieved on 2015-07-01).

20. Mona Haraty, Diane Tam, Shathel Haddad, Joanna McGrenere, and Charlotte Tang. 2012. Individual Differences in Personal Task Management: A Field Study in an Academic Setting. In *Proceedings of Graphics Interface 2012 (GI '12)*. Canadian Information Processing Society, Toronto, Ont., Canada, Canada, 35–44. http://dl.acm.org/citation.cfm?id=2305276.2305284

21. Eric Horvitz, Jack Breese, David Heckerman, David Hovel, and Koos Rommelse. 1998. The LumièRe Project: Bayesian User Modeling for Inferring the Goals and Needs of Software Users. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI'98)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 256–265. http://dl.acm.org/citation.cfm?id=2074094.2074124

22. J. Johnson, T.L. Roberts, W. Verplank, D.C. Smith, C.H. Irby, M. Beard, and K. Mackey. 1989. The Xerox Star: a retrospective. *Computer* 22, 9 (Sept 1989), 11–26. DOI: http://dx.doi.org/10.1109/2.35211

23. Helge Kahler, Anders Mørch, Oliver Stiemerling, and Volker Wulf. 2000. Tailorable systems and cooperative work. *Computer Supported Cooperative Work (CSCW)* 9, 1 (2000), 1–4. DOI: http://dx.doi.org/10.1023/A:1017243824820

24. Henry Lieberman, Fabio Paternò, Markus Klann, and Volker Wulf. 2006. End-User Development: An Emerging Paradigm. In *End User Development*, Henry Lieberman, Fabio Paternò, and Volker Wulf (Eds.). Human-Computer Interaction Series, Vol. 9. Springer Netherlands, 1–8. DOI: http://dx.doi.org/10.1007/1-4020-5386-X_1

25. Wendy E. Mackay. 1991. Triggers and Barriers to Customizing Software. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '91)*. ACM, New York, NY, USA, 153–160. DOI: http://dx.doi.org/10.1145/108844.108867

26. Thomas W. Malone, Kum-Yew Lai, and Christopher Fry. 1995. Experiments with Oval: A Radically Tailorable Tool for Cooperative Work. *ACM Transactions on Information Systems* 13, 2 (April 1995), 177–205. DOI: http://dx.doi.org/10.1145/201040.201047

27. Justin Matejka, Tovi Grossman, and George Fitzmaurice. 2013. Patina: Dynamic Heatmaps for Visualizing Application Usage. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 3227–3236. DOI: http://dx.doi.org/10.1145/2470654.2466442

28. Joanna McGrenere, Ronald M. Baecker, and Kellogg S. Booth. 2002. An Evaluation of a Multiple Interface Design Solution for Bloated Software. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '02)*. ACM, New York, NY, USA, 164–170. DOI: http://dx.doi.org/10.1145/503376.503406

29. Xiaojun Meng, Shengdong Zhao, Yongfeng Huang, Zhongyuan Zhang, James Eagan, and Ramanathan Subramanian. 2014. WADE: Simplified GUI Add-on Development for Third-party Software. In *Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems (CHI '14)*. ACM, New York, NY, USA, 2221–2230. DOI: http://dx.doi.org/10.1145/2556288.2557349

30. Ben Shneiderman. 2003. Promoting Universal Usability with Multi-layer Interface Design. In *Proceedings of the 2003 Conference on Universal Usability (CUU '03)*. ACM, New York, NY, USA, 1–8. DOI: http://dx.doi.org/10.1145/957205.957206

31. Daniel J Simons and Christopher F Chabris. 1999. Gorillas in our midst: Sustained inattentional blindness for dynamic events. *Perception-London* 28, 9 (1999), 1059–1074.

32. Gunnar Stevens and Torben Wiedenhöfer. 2006. CHIC - a Pluggable Solution for Community Help in Context. In *Proceedings of the 4th Nordic Conference on Human-computer Interaction: Changing Roles (NordiCHI '06)*. ACM, New York, NY, USA, 212–221. DOI: http://dx.doi.org/10.1145/1182475.1182498

33. Wolfgang Stuerzlinger, Olivier Chapuis, Dusty Phillips, and Nicolas Roussel. 2006. User Interface Façades: Towards Fully Adaptable User Interfaces. In *Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology (UIST '06)*. ACM, New York, NY, USA, 309–318. DOI: http://dx.doi.org/10.1145/1166253.1166301

34. Gaurav Paruthi Tao Dong, Mark S. Ackerman, Mark W. Newman. 2013. Social Overlays: Collectively Making Websites More Usable. Lecture Notes in Computer Science, Vol. 8120. Springer Berlin Heidelberg, Berlin, Heidelberg. DOI: http://dx.doi.org/10.1007/978-3-642-40498-6

35. Volker Wulf and Björn Golombek. 2001. Direct activation: A concept to encourage tailoring activities. *Behaviour & Information Technology* 20, 4 (2001), 249–263. DOI: http://dx.doi.org/10.1080/01449290110048016