Skillometers: Reflective Widgets that Motivate and Help Users to Improve Performance

Sylvain Malacria[1] **Joey Scarr**[1] **Andy Cockburn**[1] **Carl Gutwin**[2] **Tovi Grossman**[3]
[1]University of Canterbury, Christchurch, New Zealand
[2]Department of Computer Science, University of Saskatchewan, Saskatoon, Saskatchewan, Canada
[3]Autodesk Research, Toronto, Ontario, Canada
sylvain@malacria.fr, {joey, andy}@cosc.canterbury.ac.nz, gutwin@cs.usask.ca, tovi.grossman@autodesk.com

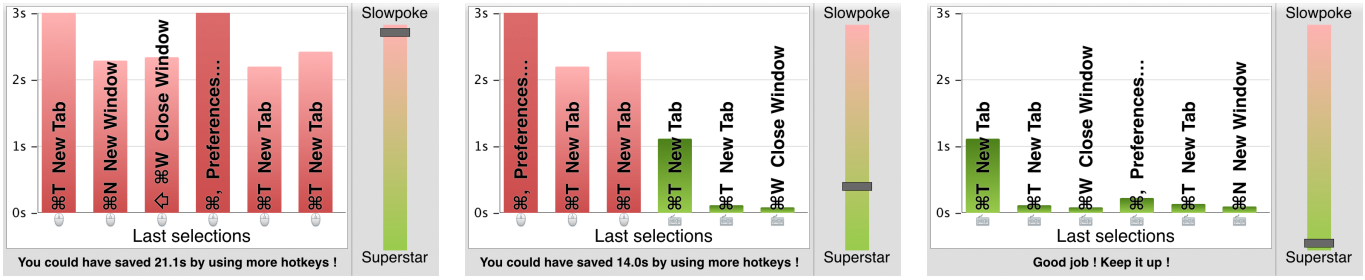

Figure 1. Three snapshots of a skillometer designed to encourage hotkey use. At first (left), the user mostly relies on mouse selections for activating commands. The skillometer indicates that he could save time by using hotkeys, and displays the appropriate hotkey bindings. As he begins to increase his hotkey use (center), the skillometer shows the benefits of the switch, encouraging him to use more hotkeys (right).

ABSTRACT

Applications typically provide ways for expert users to increase their performance, such as keyboard shortcuts or customization, but these facilities are frequently ignored. To help address this problem, we introduce *skillometers* – lightweight displays that visualize the benefits available through practicing, adopting a better technique, or switching to a faster mode of interaction. We present a general framework for skillometer design, then discuss the design and implementation of a real-world skillometer intended to increase hotkey use. A controlled experiment shows that our skillometer successfully encourages earlier and faster learning of hotkeys. Finally, we discuss general lessons for future development and deployment of skillometers.

Author Keywords

Expertise; hotkeys; learning; reflection.

ACM Classification Keywords

H.5.2. Information Interfaces and Presentation: User Interfaces. - Graphical user interfaces.

General Terms

Human Factors; Design; Measurement.

UIST'13, October 6–9, 2013, St. Andrews, United Kingdom.

ACM 978-1-4503-2271-3/13/10...$15.00.
http://dx.doi.org/10.1145/2501988.2501996

INTRODUCTION

In modern applications, users have many opportunities to enhance their productivity. They can alter their environment (e.g., customizing the interface to suit their workflows and bring frequently-used commands close to hand), or they can alter themselves by learning new skills and techniques for the application. These new techniques can dramatically improve performance, but a major problem in many systems is that users ignore these opportunities [23, 27]. For example, many users opt to "satisfice" [38] and stick with the interaction procedures they learnt as novices, even when they are aware of the expert features [12].

This behavior is not unreasonable. Adopting a new mechanism or strategy could require a significant investment of time and effort, possibly outweighing any efficiency gains; if the user has no way of estimating these costs and benefits, then they should not be expected to switch. Critically, it is the user's *perception* of the new mechanism that matters [35], rather than its actual properties – therefore, if we could inform the user about the benefits of adopting a new mechanism or strategy, they might be more inclined to do so.

In this paper, we present and analyse the design potential of using *skillometers* as a solution to these problems. Skillometers are widgets designed to accelerate the development of UI expertise by allowing the user to visualize their past and potential future performance. While the term 'skillometer' has been used in the domain of intelligent tutoring systems to reflect the system's model of the user's understanding (e.g., [4, 5]), we are unaware of their use for interface performance, except for Linton and Schaefer's proposal to use a "Skill-O-Meter" to visualize command vocabulary [26]. Other related systems are reviewed later in the paper. The specific skil-

lometer implemented and evaluated in this paper, shown in Figure 1, displays a visualization of a user's interaction efficiency (in this case, with menus and keyboard shortcuts), and suggests ways in which efficiency could be improved.

This paper makes three specific contributions. First, we present a general skillometer framework which can be used to inform the design of future skillometers. Second, we discuss the design of HotkeySkillometer, a skillometer for encouraging hotkey use, and describe its implementation. Finally, we present the results of a controlled experiment, where participants performed tightly constrained tasks in Apple Keynote. Results demonstrate that the HotkeySkillometer motivates users to increase their use of hotkeys. We conclude by discussing the lessons learned and issues of generalisation.

SKILLOMETER DEFINITION

Reflective interfaces, which inform the users about their own behavior, have been well-studied in the area of tutoring systems (e.g. [5, 24]), which are specifically designed to help the user develop skills and knowledge. In the context of interaction, Linton and Schaefer [26] proposed a reflective visualization called a 'Skill-O-Meter', which displays the user's command vocabulary compared to others in their peer group. However, the effect of this feedback on user behavior was not evaluated. More recently, Bateman et al. [8] developed a 'Search Dashboard' that gave users information about their search habits, including comparison of their behaviors to those of typical and expert users. By displaying information about expert behavior, the Search Dashboard was able to broaden users' knowledge of search engine features.

In this paper, we define a *skillometer* as a reflective widget that motivates and helps the user increase their level of performance with an interface. Importantly, as reflective widgets, skillometers support the user in understanding meta-level aspects of their own interaction. Typically, the meta-level aspect of interaction revealed by a skillometer will concern the user's level of performance, although others may be revealed, such as the user's task strategy or interaction modalities used to select commands (examples follow later in the paper).

Under this definition, a tooltip that displays a hotkey binding would not be considered to be a skillometer because although it assists the user in increasing their level of performance (by helping the user learn the hotkey), it does not facilitate understanding of performance at a higher level. In contrast, the widget shown in Figure 1 is a skillometer because it displays the user's level of performance (a meta-level aspect of interaction) as well as showing hotkey bindings that might be used to complete pointer-based selections (shown in red).

TARGET DOMAINS OF USER PERFORMANCE

Skillometers can be deployed to improve four different target domains of user performance with interfaces: *intramodal*, *intermodal*, *vocabulary extension*, and *strategic*. Each of these domains is described with respect to potential skillometer design and prior work in upcoming subsections. In brief, *intramodal* improvement addresses the rapidity of skill acquisition within one particular interactive method for one particular function (e.g., assisting the user to improve their performance with a novel pointing device); *intermodal* improvement addresses ways to assist users to switch to faster methods for accessing a particular function (e.g., switching from cursor-based pointing to hotkeys); *vocabulary extension* methods focus on broadening the user's knowledge of the range of functions available through the interface; and *strategic* methods address higher level issues of how users combine interface functions to accomplish tasks.

Domain 1: Intramodal Improvement

Human skill acquisition has been shown to reliably follow a 'power law of practice' [31] across a wide range of activities, including interaction with computer systems. The law of practice shows that performance improves quickly at first, gradually leveling off with experience. In many interaction contexts, such as cursor-based selection of menu items, this effect is easily explained: users who are unfamiliar with the interface must initially rely on visual search to identify likely targets, but can gradually use faster methods for interaction based on spatial, muscular and procedural memory. After extensive experience with any particular interaction modality, users will approach a performance ceiling representing the limit of the user's capability for that modality.

While performance improvement within any particular interaction modality will follow a power law of practice, research in the psychomotor literature clearly shows that different forms of practice and training can yield different rates of skill acquisition – see [36] for an extensive review. For example, there are reliable experimental effects for various practice manipulations, including the timing and distribution of practice, the mental effort associated with processing trained material, the specificity of the training material, and various feedback effects.

Accelerating the rate of skill acquisition is one opportunity for skillometers to improve performance. For example, some psychology literature suggests that *knowledge of performance* can improve skill acquisition (although results can vary [36]), and this effect might be employed by a skillometer. A touch-typing skillometer, for instance, might show a real-time line chart of the user's word-per-minute rate (Figure 2a). Additionally, many psychology and sports studies shows that appropriate goal-setting can improve skill acquisition (again, see [36]). This result might be incorporated into the skillometer by continually showing the user's peak typing rate as another line on the chart above their current rate.

Domain 2: Intermodal Improvement

Most interfaces support more than one interaction modality for accessing the same function: for example, the 'Bold' function might be activated by selecting it from a menu, clicking a toolbar button, choosing it in a dialog box, or pressing Ctrl+B. The rate of skill acquisition for each independent modality is the concern of *intramodal* improvement. *Intermodal* improvement, in contrast, concerns the transition from slower modalities to faster ones that ultimately offer a higher performance ceiling.

Many studies have observed that expert interface modalities are often unused or only lightly used. Carroll [12] attributed

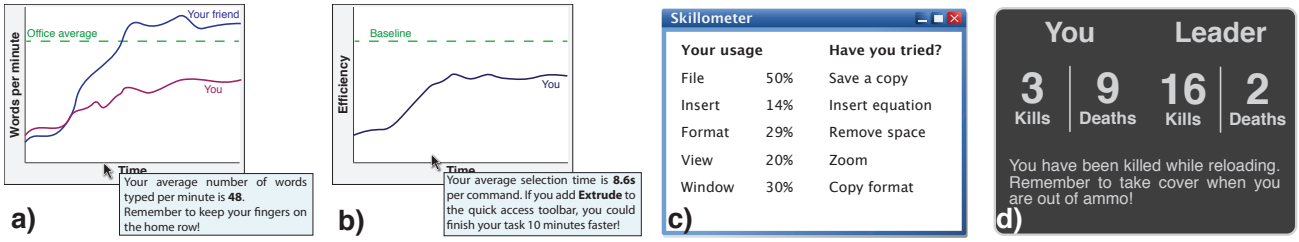

Figure 2. Concept sketches of skillometers for a) typing speed (intramodal), b) interface customization (intermodal), c) command awareness (vocabulary extension) and d) video game training (strategic).

the effect to the *paradox of the active user*, noting that users tend to be too engaged in their primary task to put cognitive effort into learning more efficient ways to accomplish that task. This behavior is an example of a wider human phenomenon called *satisficing* [38], where people make do with sufficient, but suboptimal, strategies due to limited cognitive resources: typical users do not have the necessary information to determine whether learning hotkeys will be a worthwhile investment, nor can they spare the substantial effort required to acquire that information. In related work, Scarr et al. [35] further examined users' failure to make a transition to higher-performance expert modalities, noting that a temporary 'performance dip' while gaining familiarity with a new modality can deter users from switching to a new method.

Several factors act as deterrents or barriers to users switching to modalities that can offer a higher performance ceiling, including the user's lack of awareness of other modalities, their perception of performance with other modalities, and their lack of motivation to use them. Each of these is described below, with particular reference to the opportunities for skillometers.

Awareness of Other Modalities

There are many ways to increase awareness of advanced interaction modalities. For example, keyboard shortcuts are sometimes displayed to the right of menu labels, or in tooltips that appear after a dwell. However, Grossman et al.'s results suggest that even when shortcuts are displayed in dropdown menus, they are commonly ignored by users [20]. This may be attributable to the fact that the shortcut is not displayed until the user has already done most of the work for selecting the command with the mouse, so they have no incentive to learn the hotkey. Malacria et al. [28] and Tak et al. [39] both developed successful systems that provide "feedforward" information about hotkeys when a modifier key (e.g. Ctrl) is held down, displaying hotkey mappings when they are needed rather than after the fact, with both systems increasing hotkey use. Krisler and Alterman showed improved hotkey learning with their HotKeyCoach [22], which displayed a pop-up window that needed to be explicitly dismissed every time the user selected a command with the mouse. However, explicit awareness techniques often come at the cost of undue distraction, and one of the goals of our skillometer is to minimise interference and disruption of normal interaction (explicit dialog boxes clearly violate this goal). Scarr et al.'s "calm notification" [35] technique provides awareness in a less intrusive fashion by using an ambient dialog in the corner of the screen that does not require user response and that automatically fades away if ignored.

The visual presentation of a skillometer widget can incorporate awareness information about appropriate expert modalities. Using a skillometer as the location for this information has two important advantages over other methods: first, it provides a consistent location for feedback on higher-performance methods for completing recent actions; second, the skillometer can show performance over time, allowing users to compare past and potential future performance (e.g., the skillometer in Figure 1 shows information about the last six commands).

Perception of Performance with Other Modalities

In their framework of interface expertise, Scarr et al. [35] note that the user's *perception* of their potential performance in a modality (both over the short and the long term) is a critical factor influencing their decision of whether to switch. However, a user's perception of future performance is often unreliable [13]. In the context of command selection, studies have consistently shown that keyboard shortcuts offer a higher performance ceiling than mouse selection [23, 32], yet Tak et al. found that some participants failed to use known hotkeys because they believed the toolbar buttons to be faster [39].

Appropriate skillometer design may assist with communicating accurate perceptions of the relative potential and actual performance gains of using different modalities. For example, a skillometer might show the actual time taken to select the 'bold' command in a ribbon or menu (calculating time from completing a text selection and selecting the item) and also show the estimated keystroke time for the associated hotkey. Alternatively, time estimates might be aggregated and/or projected to future total time savings, based on the frequency with which particular commands have been used in the past (e.g., the tooltip in Figure 2b). Such information might assist the user in making rational judgements concerning the costs and benefits of changing modality.

Lack of Motivation

The user's motivation for improving performance is another important factor. There are many elements of motivation at play in a user's decision to learn new skills and techniques, and skillometer design can make use of both intrinsic and extrinsic elements. First, intrinsic motivation concerns the user's own interest in improving, and here it is important for the skillometer to clearly visualize the potential for future improvement. However, designers must recognize that

the user's performance goal may depend upon the context – for example, a user browsing the Internet at home may be less concerned with fast command selection than an office worker who carries out the same tasks throughout the day. Interestingly, however, Tak et al. [39] found that increased time pressure did not lead to increased hotkey use, suggesting that workload alone is not sufficient to encourage adoption of more efficient methods (reflecting Carroll's paradox).

Extrinsic motivation can also be used in skillometer design – in particular, there are interesting opportunities for skillometers to use social factors, social learning, and gamification (discussed later) as motivation. For example, recent results from Banovic et al. [7] showed that social influence was important for encouraging tool palette customization. Similarly, Peres et al. [33] found that users were more likely to use keyboard shortcuts if they worked with others who did so, and Bateman et al. [8] found that having knowledge of expert search engine use led people to change their own behaviour.

Domain 3: Vocabulary Extension

Many interfaces allow access to extensive functionality, but users are often unaware of functions that may assist their performance (see [30] for a review of the problem). For example, in a photo editing suite a user may be familiar with a tool for painting filled circles and may use it to remove red-eye from photographs, unaware that a more effective dedicated red-eye removal function is available. Existing interface mechanisms such as 'tip of the day' can assist in extending the user's vocabulary of commands, but the information they provide is typically unconnected with the user's current task, and may therefore be viewed as an unwanted distraction.

The conceptual deterrents and barriers to vocabulary extension are similar to those described for *intermodal improvement* described above. The opportunities for skillometers to assist with overcoming these deterrents and barriers are also similar to those described for intermodal improvement, although there are additional challenges in determining the functional relationships between groups of commands. While it is straightforward for a skillometer to determine and display intermodal relationships (such as the availability of a hotkey when a menu item is selected by pointing), determining functional relationships is likely to benefit from some semantic knowledge of the user's task objective. For example, a skillometer could only represent the efficiency of the 'red-eye' example above if it correctly identified that the user's manipulation of the circle painting tool was directed at that specific task rather than some other activity.

However, skillometers are not restricted to displaying information about task completion times. Rather, they might be used to promote reflection about other meta-level aspects of interaction, such as the range of commands used. In such a deployment, a skillometer might depict the set of commands that the user employs, grounded by a depiction of the set of commands used by other users – Linton's 'Skill-O-Meter' proposal [26] and Bateman's 'Search Dashboard' [8] are examples. Furthermore, the skillometer might incorporate information about the frequency of use of commands among a community of users, thereby alerting a user to unknown commands that others find useful (e.g., Figure 2c). CommunityCommands [30] and Patina [29] were directed at these objectives, with Patina's interface representation using a heatmap stencil overlay on top of the user interface to show the frequency of command use among a community of users.

Domain 4: Strategic Improvement

Strategic improvement involves helping the user to optimize their sequence of actions – i.e., choosing a better *strategy* to complete the task. While vocabulary extension often facilitates strategic improvement (e.g., invoking 'Align Left' is a better strategy than using drag-and-drop to manually left-align items), users may need more guidance than simply providing broader awareness of application functionality. We refer the reader to Bhavnani et al. [10] for a detailed analysis of strategic aspects of computer use, but we use one of their examples to illustrate interaction strategies. Consider a user interacting with a drawing package to create a series of identical arched windows. A novice user might use a 'sequence-by-operation' strategy, first drawing all of the arcs for the window arches (arc operation), then drawing all of the vertical lines for the window sides (line operation), and finally drawing the horizontal lines for the window bases. An experienced user, in contrast, might use a 'detail-aggregate-manipulate' strategy, first drawing the arc and lines for one window (detail), then grouping the lines (aggregate), and finally copying and pasting multiple replicates (manipulate).

In theory, a skillometer targeted at strategic improvement would detect inefficient interaction strategies and suggest better ones. In general, detecting suboptimal user interaction strategies is a difficult problem. However, in some applications (particularly games), there are points where it is easy to tell that the user has performed poorly. For example, a strategic skillometer for a first-person shooter game might analyse players' deaths and give targeted advice such as 'remember to take cover while reloading' or 'keep your movements unpredictable' (Figure 2d).

SKILLOMETER INTERFACE: GOALS AND MECHANISMS

As discussed above, skillometers may be targeted at any of four different domains of user interface performance. However, regardless of the target domain, there are four goals that apply to the user interface of any skillometer, as follows.

Goal 1: Visualize the Possibility for Improvement

A skillometer interface should clearly communicate to the user that their performance can be improved. This can mean different things, depending on the purpose of the skillometer. An *intramodal* skillometer might display a graph of the user's performance compared to a theoretical optimum. For an *intermodal* skillometer, this means making the user aware of different modalities and their benefits – for example, displaying the user's performance compared to their performance if they used hotkeys. Skillometers aimed at *vocabulary extension* and *strategic improvement* might suggest new functionality and interaction strategies to the user, and demonstrate how these might improve efficiency or quality of work.

Goal 2: Provide Specific Details on How to Improve
As well as making the user aware of the possibility of improving their performance, a skillometer should also provide specific guidance to the user about how that improvement could take place. For example, an intermodal skillometer would provide details of how to perform commands with a better modality, such as showing the hotkey bindings for each command performed.

Goal 3: Motivate the User to Improve
Motivation can be provided by a skillometer in several different ways. A skillometer's visualization should stimulate the user's intrinsic motivation: as discussed in Goal 1, it should enable the user to quantify the benefits of improvement. However, there are additional methods available to motivate and persuade users, discussed below.

Social Influence and Competition
As mentioned earlier, social influence can be an important extrinsic motivation for changing user behavior (e.g., [7, 8, 33]). A skillometer could make use of this by comparing the user's performance, modality use, command knowledge, or interaction strategies to others in their peer group. For example, a touch-typing skillometer could display extra lines on the graph representing the typing speeds of co-workers. However, care must be taken, as some users may react negatively to unfavorable comparison with others.

Exaggeration of Benefits
Since the point of a skillometer is to persuade users to change their behavior, one option would be to exaggerate the advantages of doing so. This is a case of *benevolent deception* [3], where an interface lies in order to provide some benefit to the user. As with competition, care needs to be taken with this approach to avoid negative effects – if the deception is too obvious, the user will lose trust in the system and stop attending to the feedback.

Gamification
Gamification is a broad research topic, and we refer readers to [14] and [21] for a review of gamification techniques. In brief, game elements have been demonstrated to enhance the effectiveness of training with software systems (e.g., [15, 25]), and gamification has been deployed to assist learning of commercial software interfaces (e.g., Microsoft's Ribbon Hero [2]). Gamified interactive components could be used as part of a skillometer, such as scoreboards, achievements, and progression-based elements equivalent to game levels.

Goal 4: Minimize Disruption to Normal Interaction
While providing feedback on user performance can be beneficial, it has also been shown to degrade learning if given too frequently [34]. A skillometer should therefore aim to provide information when it is most useful to the user, and minimize disruption of the user's normal workflow. Research by Bødker [11] suggests that there are two types of task interruption: *breakdowns* and *focus-shifts*. Breakdowns result in severe disruption and force the user's attention to a new activity, while focus-shifts cause only a brief attention switch and have a minimal effect. Systems resulting in a breakdown, such as Microsoft's *Clippy*, have been shown to be quickly abandoned after a few untimely interruptions [37].

Ideally, skillometer feedback would be transient, unintrusive, and require no user response, eliciting a focus-shift rather than a breakdown from the user. However, this presents a difficult trade-off: in order for a skillometer to counteract Carroll's *active user paradox*, a certain amount of disruption is necessary to draw attention to the skillometer.

Manipulating the *locus of control* is one way that a skillometer could avoid this tradeoff. Push notifications, such as a window popping up on screen, can be distracting and interrupt the user's workflow. However, if the locus of control rests with the user, they can choose to request the feedback at a time that suits them, and ignore it when they are busy. The risk of this approach is that the user may never request the feedback – so a possible compromise is a small ambient display in the corner of the screen, which constantly displays a performance overview and provides more detailed information and guidance only on demand.

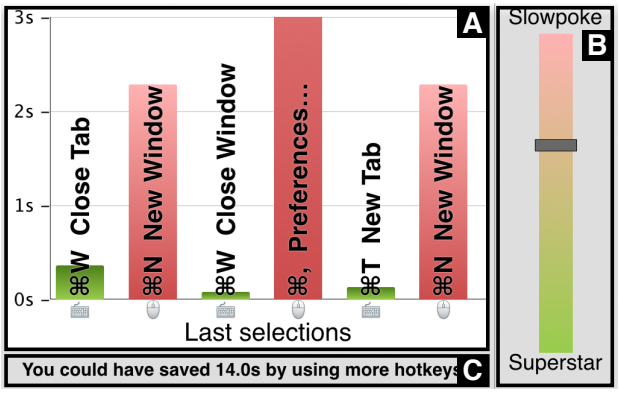

Figure 3. The visual design of HotkeySkillometer.

A SKILLOMETER FOR PROMOTING HOTKEY USE

To gain practical experience with the challenges of designing and implementing skillometers, and to gain initial insights into the validity using skillometers to assist expertise development, we constructed and evaluated a system called HotkeySkillometer. The system, shown in Figure 3, is directed at target domain #2, *intermodal improvement* – it aims to motivate and assist users in transitioning from mouse-driven interaction methods (menu and toolbar selections) to hotkeys [20, 28, 6]. We chose to examine hotkey intermodal improvement because hotkeys are widely available in commercial software, yet they are known to be lightly used [23].

Overview of HotkeySkillometer
HotkeySkillometer monitors the user's interaction with graphical desktop applications, analysing and depicting their level of performance, and displaying alternative and faster hotkey methods for executing commands whenever they are available. Its display consists of three parts, labelled A, B, and C in Figure 3. Part A (Fig. 3, left) uses a bar chart to show the time taken to select each of the six most recently used commands (the Implementation section below details how selection times are calculated to determine each bar's height).

When a new command is selected, an animation slides a new bar item into view, scrolling all other items leftwards. Each bar is coloured dependent on the selection modality (red for mouse, green for hotkey), and an icon below each bar also depicts the modality. Additionally, each bar is overlaid with text showing the name of the command and its hotkey.

Part B (Fig. 3, right) is a meter that grades the user's performance based on the modality used for the last 6 command selections – if all selections were made with hotkeys, the slider will be at the 'Superstar' end; if all used the mouse, it will be at the 'Slowpoke' end. The value of the slider is determined by a weighted interpolation, where more recently selected commands have more weighting – this gives the bar a more noticeable 'swing' toward the positive end each time a hotkey is used, or the negative end with each mouse selection.

Part C (Fig. 3, bottom) shows motivational text advising the user that hotkey selections are faster. It also quantifies, across the past 6 selections, how much time could have been saved if the selections were performed with hotkeys instead (based on a KLM-derived value of 200ms for each hotkey selection).

By default, HotkeySkillometer is a transient window located at the bottom left (or right, based on user preference) of the display. As soon as the user selects a command or presses the Command key, the window fades in and remains until the user stops selecting commands. When the user mouses over the window, it fades out revealing the content beneath it. While the mouse-over fading enables users to interact with underlying content, it also prohibits intentional interaction with components within the skillometer – this design choice is appropriate for HotkeySkillometer, but will be inappropriate for other designs where the user needs to be able to explicitly request information associated with skillometer components (e.g., the roll-over tips shown in Figure 2a).

With respect to the interface goals, HotkeySkillometer supports *Goal 1: Visualize the possibility for improvement* and *Goal 3: Motivate the user to improve* in both the height of mouse-based bar chart items (suggesting poor performance) and through the performance meter, which quickly moves towards the 'Slowpoke' end following mouse selections. It supports *Goal 2: Provide specific details on how to improve* in Part C's text and by the hotkey overlay on the bar chart. *Goal 4: Minimize disruption to normal interaction* is supported by using a transient window that only appears after each command selection and which quickly fades out with inactivity.

IMPLEMENTATION OF HOTKEYSKILLOMETER

The implementation of HotkeySkillometer requires three fundamental building blocks. First, it has to be able to monitor and detect command selections. Second, it has to determine which modality (e.g., mouse or hotkey) was used to select the command. Third, it has to calculate and display the user's performance. Our implementation, HotkeySkillometer, is a Mac OS X application, developed using Objective-C and the Cocoa framework. It uses Apple's Accessibility API [1] to browse and access GUI control elements; all Mac OS X applications implementing this interface can therefore be observed by our skillometer. Details of how HotkeySkillometer achieves each of the three building blocks follow.

Monitoring and Detecting Command Selections

HotkeySkillometer uses two separate components for menus and toolbar selections.

Menu Items

When an application first receives focus, HotkeySkillometer dynamically adds global action listeners to all the menu items located within an application's menu bar and context menus. These action listeners are triggered as soon as the user selects a command (either by clicking on the menu item, or using its associated hotkey), and the listener is provided with all the accessibility attributes of the menu item, which includes the command name and keyboard shortcut.

Toolbar Buttons

Handling toolbar selections is more complicated, since Apple does not provide a global listener for toolbar buttons. Unlike the menu bar, toolbar buttons are often custom widgets that do not implement the Accessibility API, which complicates identifying widgets and accessing any associated hotkey.

To circumvent this limitation, HotkeySkillometer uses a hard-coded description of the mapping between toolbar buttons and their equivalent menu items for each of the client interfaces it monitors. It then uses a combination of a global mouse event listener (that intercepts mouse-up actions) and the Accessibility API to identify the widget located beneath the mouse cursor and thus determine which command was executed. An alternative approach, not used in HotkeySkillometer, would use code injections for extending existing method of the applications at runtime [16]. However, this method would be limited to Cocoa applications and would require a plugin for each monitored application.

Determining the Modality Used

Since toolbar selections use a different detection mechanism to menus, they are easily distinguished. However, for menu items that also possess hotkeys, the same action listener is called for both pointer and hotkey selections. To discriminate between selection modalities, HotkeySkillometer listens to global mouse and keyboard events and checks whether the key corresponding to the keyboard shortcut character was pressed immediately before command selection. If so, the command is recorded as a hotkey selection.

Calculating User Performance

HotkeySkillometer measures pointing time using an Objective-C implementation of Evans and Wobbrock's Input Observer [18], which assumes that a user's current pointing action began with the most recent ballistic movement. The time to select a toolbar item therefore begins at the user's most recent ballistic motion, and ends when the toolbar button is clicked. Menu item selection time is calculated as the sum of the time taken to point to the top-level menu bar item, and the time spent in the menu (and submenus) prior to command selection. For context menus, selection time is simply the time spent in the menu prior to selection.

For keyboard shortcut selections, HotkeySkillometer simply measures the time between the first modifier keypress (or the last command selection in the case of consecutive keyboard shortcut activations) and the command selection time.

Note that both of these measures exclude the time for mental preparation and for precursor movements such as homing the hand to the mouse or key. Consequently, both measures represent the lower-bound of the selection time.

EXPERIMENTAL STUDY

We performed a lab experiment using HotkeySkillometer to gain initial insights into the validity of the skillometer approach to performance improvement. The experimental tasks were conducted using Apple's Keynote software in a between-subjects design, with half of the participants having access to HotkeySkillometer and the other half not.

Participants and Apparatus

24 participants were recruited from a local university (20 male, 4 female). The experiment was performed on an Apple Macbook Pro running Mac OS 10.8. Participants were provided with a standard mouse and QWERTY keyboard, and a 24" screen running at 1920×1080 pixels.

Since Keynote ordinarily lacks keyboard shortcuts for many commands, we used the Mac OS System Preferences to add custom hotkeys for all of the commands used in the experiment. Each shortcut was composed of a single modifier key (Command) followed by one of the 18 alphabetic keys from the left part of the keyboard. No hotkey letter was the same as the first or last letter of the command name.

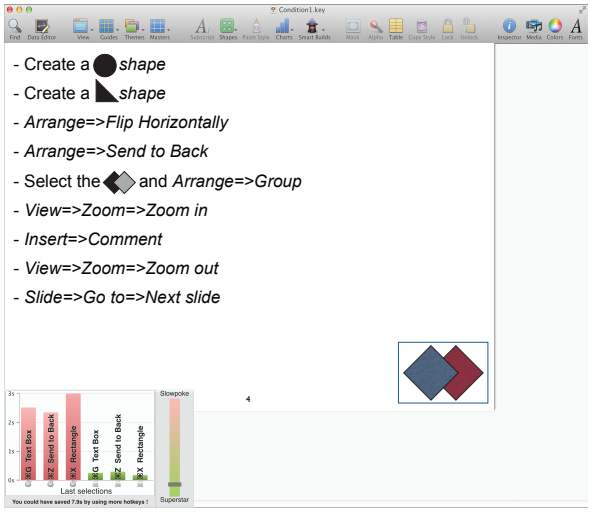

Figure 4. The initial state of a slide used in the experiment. Participants performed tasks on the slide surface, and the instructions were included as a background image. The skillometer appeared at bottom left.

Procedure

Participants were pseudo-randomly assigned to either the *skillometer* or *control* condition (*control* having no access to HotkeySkillometer). Participants were introduced to the Keynote software and to the experimental procedure – they were instructed that they would be completing 'a repetitive series of tasks'. In the *skillometer* condition, participants were additionally given a brief description of the skillometer, explaining the visual presentation of parts A, B, and C shown in Figure 3. To reduce the risk of introducing bias, the skillometer instructions did not mention its overall goal. Participants in both conditions were informed that hotkeys existed for all commands (with labels shown next to menu items). Finally, they were instructed to complete tasks as accurately and quickly as possible.

Each participant performed eight formatting commands on a Keynote slide, then selected a 'next slide' command. They repeated the same eight commands (and next slide) ten times, giving a total of $10 \times 8 = 80$ logged command selections. Instructions for the eight formatting commands were presented in a Keynote slide (Figure 4), and each command involved manipulating components within the slide or view to achieve a particular effect – for example, zooming in or flipping items on the slide. The 'next slide' command initiated the next set of 8 commands on a new slide identical to the preceding one. Timing for each command selection consisted of the elapsed time since the previous correct command selection. All user actions were logged, and the whole experiment took participants 15 minutes on average. To gain insight into the behavior of our participants, we collected post-experiment comments using a short questionnaire. We had considered using a think-aloud protocol to achieve this, but since think-aloud protocols achieve the same goal as skillometers (i.e., promoting reflection during interaction) [17], it would have likely confounded our experimental objectives.

Design and Hypotheses

The experiment was designed as a 2×10 mixed-design ANOVA, with between-subjects factor *Interface* {*skillometer, control*} and within-subjects factor *Repetition* {*1..10*}. *Interface* was a between-subjects factor to minimize unintended learning effects. The two key dependent variables were proportion of hotkey use and task time (the time between command selections). Our primary hypotheses concerning skillometers were as follows:

H1: Proportion of hotkey use will be higher for *skillometer* than for *control*.
H2: Task time will be lower for *skillometer* than for *control*.
H3: There will be an *Interface* × *Repetition* interaction, showing more rapid hotkey adoption with *skillometer*.

Additionally, we wanted to ensure that the skillometer was addressing an appropriate target domain of user performance, leading to **H4:** hotkey selections will be faster than menu and toolbar selections.

Results

Hotkey Use

Figure 5-left shows the proportion of command selections completed using hotkeys for the control (left) and HotkeySkillometer (right) conditions across the ten repetitions. With HotkeySkillometer the mean rate of hotkey use across all repetitions was 50% compared to 28% for the control condition, giving a significant main effect of *Interface* ($F_{1,22} = 4.52$, $p < 0.05$) and supporting **H1**. There was

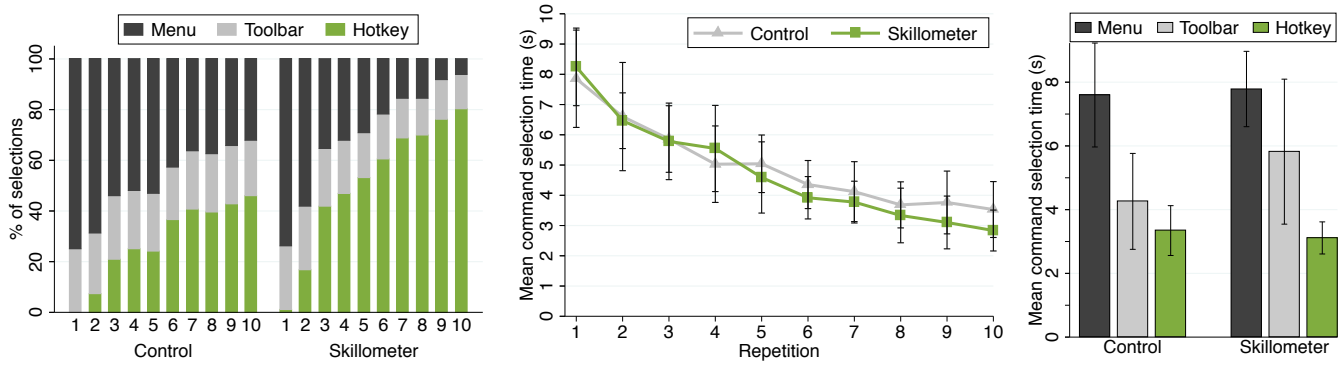

Figure 5. Percentage of selections per modality per repetition in both Control and Skillometer conditions (left). Mean command selection time across all repetitions (center). Mean command selection times across modalities and conditions (right).

also a significant interaction effect of *Interface*×*Repetition* ($F_{9,198} = 3.13$, $p < 0.01$), with participants making an earlier and more substantial switch to hotkeys when using the skillometer, with 80% of commands completed using hotkeys by the 10th repetition, compared to 42% with the control.

Command Selection Time

Figure 5-center summarizes the results for command selection time, showing that selections were initially marginally faster with the control but marginally faster in the skillometer condition across the later repetitions. Given a suggested cross-over effect, it is relatively unsurprising that there is no significant main effect for *Interface* ($F_{1,22} = 0.27$, $p = 0.6$). However, there is no significant *Interface*×*Repetition* interaction ($F_{9,198} = 0.55$, $p = 0.8$). We therefore reject **H2** (with caveats described in the Discussion).

Although there is no significant evidence that the skillometer reduced command selection time, analysis of variance for *Modality* (hotkey, menu, and ribbon) confirms that hotkey selections were much faster than toolbar or menu selections, with mean selection times of 3.3, 5.2 and 7.7 s respectively ($p < 0.001$). We therefore accept **H4**.

Figure 5-right, which shows the mean selection time for each modality in the control and skillometer condition, offers an explanation for the apparent inconsistency between the skillometer resulting in higher use of fast hotkeys, yet *not* resulting in a significant overall performance improvement. Specifically, it shows that toolbar selections were slower with the skillometer present (5.8 s compared to 4.2 s with the control).

One explanation for slow toolbar selections when using the skillometer is that users engaged in additional activities (mental and/or perceptual) in order to memorize and learn the hotkeys associated with commands, and that these activities consumed additional task time. Additionally, Figure 5-center suggests that skillometer users were still improving their performance and Figure 5-left further suggests that they were still increasing in the proportion of hotkeys used. Consequently, we suspect that participants' performance in the skillometer condition would have continued to improve as more hotkeys were learned and routinely executed. However, further empirical work is necessary.

DISCUSSION

To briefly summarise, skillometers are intended to be a widely applicable interface technique that motivates and assists users to increase their level of performance with an interface. They do so by encouraging users to reflect about their level of performance and by assisting them in understanding the performance benefits attainable by altering aspects of their interaction. We described four different domains of performance that skillometers can aim to improve (intramodal performance, intermodal performance, vocabulary extension and user strategies), and we identified four interface goals that any skillometer should seek to achieve (visualize the possibility to improve, provide details on how to improve, motivate the user to improve, and minimize disruption). We then described the design, implementation and evaluation of one specific skillometer addressing intermodal performance improvement – from point and click modality with menus and toolbars to hotkeys. Results of a lab study showed that users quickly learned and used more hotkeys with a commercial software product when presented with an accompanying skillometer.

While the experimental results are encouraging, they are preliminary and cover only a small subset of the potential domains into which skillometers could be deployed. In the following sections we discuss design refinements, explanations for our experimental results, and issues of generalisation.

Understanding and Refining HotkeySkillometer

In designing and implementing HotkeySkillometer, we followed the four interface design goals described earlier in the paper. Although the experimental results showed that the overall design successfully achieved increased hotkey use, it is difficult to know the relative contribution of each of the design goals in that success. This is an important question because the different interface design goals tend to pull the designer in different directions – for example, 'goal 4: minimize disruption' leads to subtle and transient visual effects that may go unnoticed, while goals 1-3 all encourage visual effects that are readily noticeable. Similarly, we are currently unable to determine whether each of goals 1-3 are crucial to the overall success of the system, or whether any one on its own would have had the same effect. For example, it is possible that the display of the hotkey bindings on the bar chart

(motivated by 'goal 2: provide specific details') accounts for all of the increase in hotkey adoption because it made hotkeys more salient in the display.

We attempted to mitigate some of these concerns in our experimental method by displaying only the last six commands within the skillometer (and therefore the previous use of any particular hotkey was not inside the view when next needed). We consider it unlikely that our results are attributable solely to the increased saliency of hotkey bindings, since previous research has failed to find benefits for such techniques [20]. We also attempted to understand the relative importance of different skillometer components by asking for subjective feedback on each, but this feedback yielded few strong insights. In general, participants viewed the history chart and overlaid hotkeys as being most useful, and the performance meter and motivational text least useful, with two stating that they did not look at the motivational text at all. We also noticed some participants tilting their head to read the text, suggesting that a horizontal chart would have been easier to read.

To elicit further feedback on the design we gave a real-world implementation of HotkeySkillometer to eight people to use on their personal computers for a week. While the lack of global toolbar support (see the Implementation section) reduced the usefulness of the tool, we received several comments on the design. One participant said he wanted to "measure improvement or long term performance", suggesting that the performance meter might have a configurable time component. Several participants suggested that the skillometer should not show hotkey selections, because it increases distraction when the user already knows the hotkey (counter to interface goal 4) and because it consumes space in the skillometer that might otherwise be dedicated to unknown hotkeys. However, two of the participants in the lab study would have disagreed with this recommendation, as they stated that hotkey feedback in the skillometer was useful for confirming which command was executed if they suspected an error was made. A related issue was raised regarding the inclusion of repeated command selections in the skillometer – a richer vocabulary of available hotkeys would be displayed if only the most recent repetition was displayed.

A final variation to the design of HotkeySkillometer that we are keen to pursue would blend the *feedback* methods that it currently employs with *feedforward* display techniques. HotkeySkillometer currently 'pushes' data to the user in response to their command selections. However there are rich interaction opportunities in supporting users who wish to explicitly 'pull' assistance prior to completing command execution. For example, ExposeHotkey [28] and OctoPocus [9] demonstrate improved learning of expert interface techniques by providing feedforward assistive information in response to partially complete interactions. A skillometer might use equivalent feedforward methods, such as displaying a list of frequent commands that have not been previously selected by hotkeys in response to the user pressing the command key.

Improving User Performance with Skillometers

An important issue for skillometers in general is whether they do in fact help users to improve overall performance – and in particular, why did this not occur in our initial evaluation? There are two main points for discussion in this issue.

First, the difference in completion time for toolbar selections between the skillometer and control groups may have arisen because groups were attempting to learn both the hotkeys and the toolbar locations at the same time – and in the skillometer condition, participants may have allocated more of their cognitive resources to learning the hotkeys, leading to less effective learning of the toolbar. There are often costs associated with learning new interaction modalities [35] – and it is worth noting that the skillometer group did not suffer any 'performance dip' as they switched to the new mode. More research is needed, however, to determine why the skillometer group continued to use the toolbar (or menu) in some instances – there may have been other types of 'interaction inertia' that future skillometer design could attempt to counteract.

Second, it is clear that hotkeys are substantially faster than the other two interaction modalities, so the observed slowdown in toolbar use for the skillometer group would likely be overcome by the hotkey advantage over a longer term than our 15-minute study (a trend that is also suggested by the divergence of the lines in Figure 5-center).

Challenges in Designing Skillometers for Other Domains

Although we provided design suggestions for each target domain earlier in the paper, it is clear that substantial challenges arise in designing and implementing skillometers for domains other than intermodal improvement, particularly for strategic deployments.

However, even in the complex domain of strategic improvement, there are realistic opportunities for successful deployment. In particular, computing systems are increasingly capable of capturing and usefully processing large pools of meta-information about users' interaction, both at individual and societal levels. These data repositories can be put to use to predict the user's intention and needs, and the predictions facilitate new forms of interaction to assist users in completing their tasks. For example, Fitchett and Cockburn [19] introduced the 'AccessRank' algorithm and demonstrated that it outperformed a variety of previous algorithms for predicting upcoming user actions across a wide range of user tasks. A skillometer could use such predictions to gain an understanding of the user's task, and suggest alternative methods for achieving it when alternatives are known. But again, interesting design questions emerge on whether the skillometer's role extends beyond identifying and suggesting better interactive alternatives (as we have described them), or whether they could additionally provide a shortcut method for achieving the predicted intention (which we see as beyond their scope).

Platform Requirements for Engineering Skillometers

Accessibility APIs remain the best solution for implementing systems like skillometers on a generic level. However, the main goal of these APIs is to be able to provide better access to interfaces for individuals with disabilities. These APIs are therefore usually limited to a shallow description of the UI components, and it is complicated (often impossible) to access deeper information such as the actions associated with

toolbar buttons. Ideally, the next generation of UI toolkits would provide an improved connection between UI components and their semantics.

CONCLUSIONS AND FUTURE WORK

Users have many opportunities to enhance their productivity in modern applications, one of them being to transition to expert modalities such as hotkeys. Yet users tend to stick to their familiar methods and strategies for interaction, rather than trying to improve. As a solution to this problem, this paper introduced the use of *skillometers*, a type of reflective interface that motivates and helps users to increase their level of performance. We provided a framework examining the domains of interaction that can be targeted for improvement, and discussed interface design goals for skillometers. We then described the design and evaluation of HotkeySkillometer, an exemplar skillometer that motivates and helps people to use more hotkeys. A preliminary lab study provided positive results, with experimental participants exposed to the skillometer learning more hotkeys than those in the control condition.

There are two main directions for future work. First, we will refine the design of HotkeySkillometer based on feedback from our study participants, and perform a longer-term, application-specific study with a complex application. Second, we will study skillometers in other settings and for other target domains, such as strategic improvement.

ACKNOWLEDGMENTS

This research was partially funded by Royal Society of New Zealand Marsden Grant 10-UOC-020.

REFERENCES

1. Apple accessibility api. https://developer.apple.com/library/mac/#documentation/Accessibility/Conceptual/AccessibilityMacOSX/OSXAXIntro/OSXAXintro.html.
2. Ribbon hero 2. http://www.ribbonhero.com/.
3. Adar, E., Tan, D., and Teevan, J. Benevolent deception in human computer interaction. In *CHI '13*, ACM (2013).
4. Aleven, V., Koedinger, K., Sinclair, H., and Snyder, J. Combatting shallow learning in a tutor for geometry problem solving. In *Intelligent Tutoring Systems*, Springer (1998).
5. Anderson, J., Corbett, A, Fincham, J., Hoffman, D., and Pelletier, R. General principles for an intelligent tutoring architecture. *Cognitive approaches to automated instruction* (Jan 1992).
6. Bailly, G., Pietrzak, T., Deber, J., and Wigdor, D. J. Métamorphe: augmenting hotkey usage with actuated keys. In *CHI '13*, ACM (2013).
7. Banovic, N., Chevalier, F., Grossman, T., and Fitzmaurice, G. Triggering triggers and burying barriers to customizing software. In *CHI '12*, ACM (2012).
8. Bateman, S., Teevan, J., and White, R. The search dashboard: how reflection and comparison impact search behavior. In *CHI '12*, ACM (2012).
9. Bau, O., and Mackay, W. Octopocus: a dynamic guide for learning gesture-based command sets. In *UIST '08*, ACM (2008).
10. Bhavnani, S., Peck, F., and Reif, F. Strategy-based instruction: Lessons learned in teaching the effective and efficient use of computer applications. *ACM Trans. Comput.-Hum. Interact. 15*, 1 (May 2008).
11. Bødker, S. Applying activity theory to video analysis: how to make sense of video data in human-computer interaction. In *Context and consciousness*, Massachusetts Institute of Technology (1995).
12. Carroll, J., and Rosson, M. Paradox of the active user. *Interfacing thought: Cognitive aspects of human-computer interaction* (1987).
13. Czerwinski, M., Horvitz, E., and Cutrell, E. Subjective duration assessment: An implicit probe for software usability. In *IHM-HCI '01*, vol. 2 (Jan 2001).
14. Deterding, S., Dixon, D., Khaled, R., and Nacke, L. From game design elements to gamefulness: defining "gamification". In *MindTrek '11*, ACM (2011).
15. Dong, T., Dontcheva, M., Joseph, D., Karahalios, K., Newman, M., and Ackerman, M. Discovery-based games for learning software. In *CHI '12*, ACM (2012).
16. Eagan, J., Beaudouin-Lafon, M., and Mackay, W. Cracking the cocoa nut: user interface programming at runtime. In *UIST '11*, ACM (2011).
17. Ericsson, K., and Simon, H. How to study thinking in everyday life: Contrasting think-aloud protocols with descriptions and explanations of thinking. *Mind, Culture, and Activity 5*, 3 (1998).
18. Evans, A., and Wobbrock, J. Taming wild behavior: the input observer for text entry and mouse pointing measures from everyday computer use. In *CHI '12*, ACM (2012).
19. Fitchett, S., and Cockburn, A. Accessrank: predicting what users will do next. In *CHI '12*, ACM (2012).
20. Grossman, T., Dragicevic, P., and Balakrishnan, R. Strategies for accelerating on-line learning of hotkeys. *CHI '07*, ACM (2007).
21. Kapp, K. *The Gamification of Learning and Instruction: Game-based Methods and Strategies for Training and Education*. Pfeiffer, 2012.
22. Krisler, B., and Alterman, R. Training towards mastery: overcoming the active user paradox. *NordiCHI '08*, ACM (2008).
23. Lane, D., Napier, H., Peres, S., and Sandor, A. Hidden costs of graphical user interfaces: Failure to make the transition from menus and icon toolbars to keyboard shortcuts. *International Journal of Human-Computer Interaction 18*, 2 (2005).
24. Lesgold, A., Eggan, G., Katz, S., and Rao, G. Possibilities for assessment using computer-based apprenticeship environments. *Cognitive approaches to automated instruction* (Jan 1992).
25. Li, W., Grossman, T., and Fitzmaurice, G. Gamicad: a gamified tutorial system for first time autocad users. In *UIST '12*, ACM (2012).
26. Linton, F., and Schaefer, H.-P. Recommender systems for learning: building user and expert models through long-term observation of application use. *User Modeling and User-Adapted Interaction 10*, 2.
27. Mackay, W. Triggers and barriers to customizing software. In *CHI '91*, ACM (1991).
28. Malacria, S., Bailly, G., Harrison, J., Cockburn, A., and Gutwin, C. Promoting hotkey use through rehearsal with exposehk. In *CHI '13*, ACM (2013).
29. Matejka, J., Grossman, T., and Fitzmaurice, G. Patina: Dynamic heatmaps for visualizing application usage. In *CHI '13*, ACM (2013).
30. Matejka, J., Li, W., Grossman, T., and Fitzmaurice, G. Communitycommands: command recommendations for software applications. In *UIST '09*, ACM (2009).
31. Newell, A., and Rosenbloom, P. Mechanisms of skill acquisition and the law of practice. *Cognitive skills and their acquisition* (1981).
32. Odell, D., Davis, R., Smith, A., and Wright, P. Toolglasses, marking menus, and hotkeys: a comparison of one and two-handed command selection techniques. In *GI '04*, Canadian Human-Computer Communications Society (2004).
33. Peres, S., Tamborello, F., Fleetwood, M., Chung, P., and Paige-Smith, D. Keyboard shortcut usage: The roles of social factors and computer experience. *the Human Factors and Ergonomics Society Annual Meeting 48*, 5 (2004).
34. Salmoni, A., Schmidt, R., and Walter, C. Knowledge of results and motor learning: A review and critical reappraisal. *Psychological Bulletin 95*, 3.
35. Scarr, J., Cockburn, A., Gutwin, C., and Quinn, P. Dips and ceilings: understanding and supporting transitions to expertise in user interfaces. In *CHI '11*, ACM (2011).
36. Schmidt, R., and Lee, T. *Motor Control and Learning: A Behavioral Emphasis*. Human Kinetics, 2011.
37. Shroyer, R. Actual readers versus implied readers: role conflicts in office 97. *Technical Communication 47*, 2 (2000).
38. Simon, H. Satisficing. *The new Palgrave: a dictionary of economics 4* (1987).
39. Tak, S., Westendorp, P., and van Rooij, I. Satisficing and the use of keyboard shortcuts: Being good enough is enough? *Interacting with Computers* (2013).