Technical Report: TR_CS-07-267

Training Towards Mastery

Brian Krisler
Brandeis University
Volen Center for Complex Systems
Waltham, MA 02454-9110
bkrisler@cs.brandeis.edu

Richard Alterman
Brandeis University
Volen Center for Complex Systems
Waltham, MA 02454-9110
alterman@cs.brandeis.edu

ABSTRACT
Few users ever attain mastery with an application. Mastery, the state of knowing how to work efficiently with an application requires an understanding of the underlying conceptual model of the system. The Active User Paradox is one of the main inhibitors of mastery. In this study, we present HotKeyCoach, a training method designed to insert learning events into the flow of the activity that provides contextually relevant training and helps the user circumvent the active user paradox in the pursuit of application mastery.

Keywords
Active User Paradox, Activity Theory, Skill Acquisition, Mastery, Training, Design

INTRODUCTION

A well-designed user interface should accommodate both novice and expert users and provide a means for a novice to transition towards expertise [1].

At the novice level, the interface should encourage exploration and promote the discovery of the functionality required to perform their goal attaining tasks, within the context of their activity. Once the user establishes a comfort level with the interface, she should continue to learn, transitioning beyond the visual elements of the interface, to develop the deep structural knowledge characteristic of expertise [2] that will enable her to work more efficiently with the interface.

Most user interfaces contain the scaffolding required to achieve expertise. For example, menu items display the keyboard shortcut operator next to the function name, thereby eliminating the need to locate documentation for expediting frequent operations. Previous studies [3, 4] have shown that even expert users continue to place a heavy reliance on the visual display elements of the interface for successful recall of common operations. The reliance upon the display as the basis of application action retards the emergence of better thought out and more effective plan-based use of the application [5].

One inhibitor to attaining mastery is the *Paradox of the Active User* [6]. Once a user acquires a basic understanding of the operations required to perform a task, she continues to repeat the successful operations even when she is aware that more efficient execution methods probably exist [7]. As she gains experience with the application, becoming more comfortable with its functionality she spends less time exploring the interface to discover new functions and features. Her focus switches away from learning the interface towards performing her domain-associated tasks. Because she spends little or no time learning the advanced features of the interface, she must rely upon her existing functional knowledge for completing her tasks. A user plateaus in her knowledge of and efficiency with the application and its interface.

One of the objectives of our research is the development of a method for overcoming the active user paradox. Our goal is to develop a method that scaffolds users toward mastery without disrupting their everyday computer-mediated tasks. We have developed an experimental training tool, HotKeyCoach (HKC) that provides a user with the means to continue her progression towards expertise and eventually mastery of the application.

Existing training methods are frequently ineffective. User manuals fall victim to the paradox of the active user [8]. Their consultation produces a disruption to the central activity. They require the user to stop work in order to locate both the manual and the desired instructional information. Another issue is the lack of context. Designers of user manuals cannot expect to understand the multitudes of usage patterns for an application. For example a scientist performing data analysis would use Microsoft Excel differently than an accountant would for tracking spending. For training to succeed it must take into consideration the tasks of the users. The training must extend beyond the simple tasks like underlining text, towards the analysis of data by a scientist or the summary worksheet of the accountant.

In our research, we are designing a training method to help the user circumvent the active user paradox. Our model is designed to introduce relevant training as learning events within the context of the activity. Results from our study confirm the effectiveness of this method towards training.

THEORITICAL FOUNDATION

Interface Staging

Graphical user interfaces (GUIs) support three stages of application proficiency [9]. When a novice encounters an application for the first time, she searches the menus within

the interface to discover the correct item for executing the desired operation. Menu searching provides visual feedback to the user helping to reinforce the learning experience. As the user becomes more familiar with the application and the operations required for performing tasks, she transitions towards the usage of toolbars and context menus, reducing the operational distance between the desired task and its execution. These intermediate methods usually require fewer interface interactions from the user, often resulting in just a single mouse click for successful execution. Finally, as the user approaches expertise, she becomes even more comfortable interacting with the interface, requiring even less feedback. The use of keyboard shortcuts and macros helps to improve efficiency even further by reducing the time required to move the hand between the mouse and the keyboard [8]. The end goal is an optimal user experience where desired functionality is executed with little or no distraction from the interface.

However, in most cases, the transition tends to breakdown before the user reaches true efficiency, with most experienced users rarely employing keyboard shortcuts and macros [9].

Novice vs. Expert Users

As an individual transitions from novice to expert, the users knowledge about the tool transitions from a set of abstract principles towards a series of concrete usage patterns [10]. For example, a novice user of an application interface sees the interface as a set of context-free operators that she can understand without reference to a specific situation [11]. Many interface actions require exploration and repeated use in order to fully grasp their utility.

Learning how to use *Paste Special* in Microsoft Word is an obvious example of how operators require exploration and continued use to be fully understood. But even more fundamental operators require exploration and practice to gain mastery. After the user starts creating documents and applying the functionality of the interface operations, such as *Format, Font*, she starts to develop a meaning for the operation within the context of her goals. She starts to understand that each character within the document contains attributes, with each attribute effecting the visual representation of the character. She discovers, with practice, that the selection mechanisms of her document editing application provide her with the means to define the range of text to format. The process of applying operators to actual tasks and observing their effect on the problem is where expertise develops.

Once expertise is obtained, the individual no longer requires an analytical principle to connect the current situation to a relevant action. An expert can execute the desired operation without the assistance of outside mediation. When the advanced user of Microsoft Word edits her document, she can select a text region and execute *command-d*, via the keyboard to display the font properties for the selected text, without moving the action into working memory. She no longer requires the visual cue of the *Format* menu item to assist with her desired operation. She possesses the conceptual knowledge to successfully modify the font attributes without mediation.

The ability to execute operations subconsciously demonstrates a deep structural knowledge [2] of the operation. When a novice performs an operation, she relies on external cues from the interface to guide her. It is not until the interface information becomes internalized, that working memory is available to accommodate domain specific tasks. Once information becomes internalized, the user has the knowledge to construct plans, making it possible to optimize performance [12]. However, for many interactive tasks, behavior is driven by the surface features of the display [13], with most users relying on the feedback provided from the interface which prevents her from internalizing the operations. In other words, using keyboard shortcuts frees up resources to do the kind of planning that leads to better thought out and more effective application use.

Skill Acquisition to Attain Mastery

Anderson [14] describes two stages of skill acquisition, the declarative stage and the procedural stage. During the declarative stage, the user has enough information to generate the desired behavior of a skill in some form. As the user practices this skill, through the process of *compilation*, the knowledge transforms into a procedural form where it can be directly applied without outside mediation. Once compiled, the skill continues to get fine-tuned resulting in a gradual speed-up during the execution of the skill.

When it comes to using an interface, the compiling of frequent operations results in a speed up of user work. As the user becomes more efficient and knowledgeable about the articulation work it takes to use the interface, she frees up more time to perform her actual work. As a result, the application becomes transparent and unobtrusive [18]. This transparency leads to a more optimal user experience, where the user stays in the flow of the activity [19], and concentrates for extended periods of time without interruption and demonstrates expertise with the interface.

However, users frequently form procedural knowledge at the menu or toolbar stage of the activity, never reaching the keyboard shortcut stage. Models on adaptive choice [15, 16] are based on the principle that if a procedure was successful in the past, it will most likely be successful again in the future. Unfortunately, the adapted choice principle is not always the most efficient for the given task. Part of this lack of efficiency stems from an insufficient knowledge about the underlying functionality of the system [17].

By not developing an understanding of the interface actions, the user never develops a mastery of the application. Because of adaptive choice they might compile into a procedural form the wrong set of interface actions.

Mastery means you need to continue to learn despite the problem of adaptive choice and the tendency to compile into procedural form too soon.

Mastery, a state of "knowing" how to productively work through the technology, is a problem of skill acquisition. To learn to operate at a more effective level of an activity many users need help advancing beyond the exclusive mouse usage typical of the earlier stages of interface utilization. To achieve mastery requires an understanding of the conceptual model that underlies an application, i.e. learning the operations and methods that best apply to different kinds of situations. As the user becomes more efficient and knowledgeable about the articulation work it takes to use the application, she can remain within the flow of the activity for longer periods of interrupted time. By developing a deep conceptual model of the system, the amount of work required to accomplish goals diminishes, resulting in a more efficient utilization of the system.

Active User Paradox

As technology advances, application interfaces become more complex, incorporating many new features that require an increased amount of understanding for successful utilization. Training manuals are a standard method for learning how to use new technology; however, learners at every level tend to avoid reading [6], with most users preferring instead to dive straight into a system [20] and discovering the intricacies of the interface as they attempt to complete their work. As a result of this as-you-go learning method, most users typically adapt their familiarity with similar systems and functionality and improvise when required. For example, when an architect uses a new computer-assisted drawing (CAD) application she applies her existing knowledge to the new application, preferring to discover the details of the new interface while she works. This tendency to make do with existing knowledge is referred to as the *assimilation bias* [6]: the individual employs already known operations even when she knows or suspects more efficient methods probably exist [7].

Complicating the transition towards expertise even further is the *production bias*. The production bias concerns the tension between short and long term competency and productivity [6]. When a user requires a tool for completing a task, she must balance the time required to learn the functionality of the tool with the application of the tool to the task at hand. This balancing act often results in the suspension of learning new interface functionality to improve overall proficiency. When the architect employs her new CAD tool, she prefers to avoid the tutorials and example exercises because of the overhead they entail and the conflicting time pressure to finish the tasks associated with her job.

These mutually reinforcing conflicts are referred to as the *Paradox of the Active User*. Because of the assimilation bias, users will make do with what they already know how to do. Because of the production bias, users tend to focus on the their current task, not the application. Users who frequently use an application would benefit from learning more efficient methods for task completion. But because of the production and assimilation biases the users who would benefit most from learning more about the application are also the most likely to not take the time to do the necessary learning.

The use of an application's menus versus the time it takes to learn the comparable keyboard shortcut is an illustrative example of the active user paradox. Once a user becomes comfortable with using an application's interface via menus and toolbars they will continue to leverage these operators for completing their tasks (assimilation bias). Knowing that a potentially more efficient method exists for invoking the operation, such as a keyboard shortcut exists, but not taking the time to refer to the application's manual to learn the more efficient methods is an example of the production paradox.

Our focus is on the effect of the *active user paradox* as an inhibitor to skill acquisition, as the user approaches interface mastery. When an individual turns to an application as a tool for completing a task, she must balance the time required to learn the details of the tool's functionality and interface with the task of completing her work. As a result, when the user encounters a new task based activity, she tends to rely on already known operations to complete the activity. This reliance results in the suspension of new learning, preventing the user from acquiring new, more efficient methods for satisfying the activity. In order to achieve immediate throughput, the user sacrifices long-term knowledge and efficiency in order to satisfy the short-term goal of activity completion.

TRAINING METHOD

The tool we developed, HotKeyCoach (HKC), supports the user's progression towards mastery during the course of her normal work activities. To attain this goal, the HKC model turns the user's interface actions into *learning events*: the user briefly switches from her task to reasoning about the domain [21]. At the intermediate level of skill acquisition, an individual is still required, on occasion, to take the time to make new discoveries and obtain knowledge relevant to their activity. All changes in strategy occur during learning events [22].

The HKC is built upon the following principles:

- Users have many learning opportunities for attaining mastery of the technology that mediates her normal work.
- Learning events can either introduce new material and/or provide the learner with an ability to practice previously introduced material.
- Learning events should minimize task interruption.
- The user should have complete control of the learning process.

Minimal Task Interruption

To make training during an activity's execution successful, preservation of the activity's flow is imperative. The introduction of a learning event should occur as a minimal distraction to the user. Microsoft's paperclip is a training method that results in an inappropriate distraction to the user. When the paperclip appears, a new activity is created where the user's attention is directed towards the paperclip and away from the user's work. After a few untimely interruptions, many users quickly abandon this training method [23].

Bødker, [24], defines two types of interruption that direct the user's attention away from her central activity: *breakdowns* and *focus-shifts*. A breakdown occurs when something unexpected happens resulting in the creation of a new activity. The new activity interrupts the work of the user, becoming the focus of attention. Another type of interruption is a focus-shift. A focus-shift is not as severe as a breakdown and usually occurs when a subconscious operation becomes a conscious activity, such as the articulation of a point, method or procedure. The key difference between a breakdown and a focus-shift is how it affects the flow of the central activity. A breakdown results in a severe disruption of the activity flow, while a focus-shift has a minimal effect on the flow of the activity. Not knowing how to pin an inserted object to a position in a Microsoft Word document results in a breakdown. Using a menu to select the "replace" operator in Word is a focus-shift.

To preserve the flow of activity, the user must be able to concentrate on her activity with minimal interruption [19]. The HKC model of learning introduces learning opportunities during focus-shifts. Upon the introduction of a learning event, the user has the option to ignore the introduced event and continue with her main activity, or, she can selectively learn an operation that is relevant to what she is doing at that very moment during the focus-shift. This method preserves the flow of the activity.

For example, during a document editing session, when a user attempts to change the indentation for the current paragraph, she must perform multiple steps. To satisfy this goal, she uses the mouse to select the menu operation for formatting a paragraph; selecting the menu item causes a focus-shift. The focus-shift creates an opportunity for the introduction of a learning event where she can learn the keyboard shortcut for the most recently executed formatting activity, which could lead to a reduction of work in the future.

Numerous learning events of this sort are potentially available. If every time the user selects an operation from a menu, she had the option to learn an alternate and more efficient method for achieving the same goal, over time she would gain some mastery at using the application. Leveraging these learning opportunities has multiple benefits. The more the user performs an operation, the more opportunity she has to acquire mastery of the operation. And, her learning is highly contextualized: learning is directly associated with the current activity.

The problem is that if the user must access the help system to discover the appropriate keyboard shortcut the learning event is causing a breakdown: the process of locating the proper keyboard shortcut to execute the desired operation results in an extended deviation from the main task of finishing the document. Even if she had an easily accessible cheat-sheet of keyboard shortcuts, the likelihood that the short-term cost of learning as-you-go will outweigh the long-term benefit (production bias) is low. Costly learning events of this sort introduce a breakdown in her main activity and a slowdown of her work, resulting in a reduction of productivity that is likely to be too costly to maintain for any extended period of time.

A better design for a user to learn "as-she-goes" is for a learning event to occur as a focus-shift. The HKC model of learning introduces learning events as focus-shifts rather than breakdowns. At the very moment she selects the menu item, a semi-transparent "coach" window appears informing her that she can perform the same operation using the keystrokes *option-command-L* (see Figure 1). She also has the ability to ignore *coaching*, giving her complete control over the training process.

Complete Interface Control

One of the eight golden rules of interface design [8] is that an interface must *support internal locus of control*. The user should have complete control over the interface at all times. There should be no surprising actions resulting from unexpected interface behavior. The HKC model does not usurp the user as the locus of control.

Once the user is presented with the coach tip, she has a few choices. She can either practice the keyboard shortcut at that very moment or ignore the coaching tip and continue with her work. If she is interested in learning the keyboard operation, she is given the option to execute the displayed keyboard sequence. When the user chooses to practice the keyboard sequence, the HKC software will validate the sequence and dismiss the tip if the executed keystrokes matched the proper sequence for the operation. If the user mistyped the sequence, a notification indicating the sequence was invalid is displayed. Practice leads to speedup [15] in the acquisition of the skill.

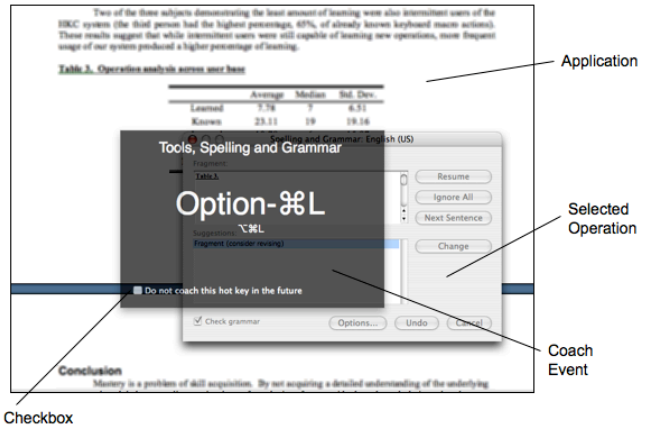

Figure 1. Example of an introduced coach event, demonstrating an instance where the subject was coached on the keyboard shortcut for executing spell checking on the document. The checkbox provides the ability not display the window for this action in the future.

However, if the user decides to ignore the tip for this particular instance, she can either click anywhere on the screen or tap the return key to dismiss the event. She also has the option of ignoring the tip for all future occurrences by selecting a checkbox in the coach window.

By providing these controls to the user, she has control over what she is taught. Having the capability to turn off coaching means the user can selectively decide on what she wants to learn.

IMPLEMENTATION

To collect evidence that our model promotes learning beyond the visual elements of the interface, we developed a native OS X 10.4+ application to implement our methods.

The application we developed, HotKeyCoach (HKC), runs as a background service and is designed to provide *coaching* to any application installed on the user's system.

In order to achieve universal coaching, we developed our application using the accessibility API [25] built into the OS X operating system. By leveraging the features of this API, we gained the capability to easily monitor all application interactions performed by the user from the operating system level. Each accessibility object also contains useful information about the most recently executed element, such as the application name, the selected operation and the keyboard shortcut if one exists. Having access to this information expanded the range of applications we were able to provide coaching for out-of-the-box because it eliminated the need to create action/shortcut mappings for all interface applications.

Leveraging the accessibility API provides an almost universal, system wide application of our training method. It places no extra requirements upon the training system for new applications. However, the accessibility API does have its limitations in that not all interface widgets return useful objects containing pertinent information. For example, the palettes within Microsoft Office provide quick access to functionality but from an accessibility point of view, they do not provide any useful information pertaining to the corresponding keyboard shortcut. Even worse, in some cases they do not even contain an easily obtainable unique identifier that would allow the training application to refer to a lookup map. We are currently implementing a solution that will help us identify all interface elements.

Data Collection

To analyze the effectiveness of our methods, HKC is designed to transfer data to a central data harvester. Once harvested, the data is stored in an SQL database for data analysis.

A major requirement of our design is the protection of user privacy. We assign each user a randomly generated identification number the first time the application runs. Each subject is known only as a random number for the duration of the study.

For each interface interaction, we collect a few key items for analysis:

- Timestamp of the interaction
- Method of interaction (mouse or keyboard)
- Name of the interaction (*Copy, Paste,* etc.*)*
- Name of the application (*Word, Safari,* etc.)
- If the interaction is ignored by the user

Information Presentation

To present a learning event to the user, HKC displays the information relevant to the learning event within a semi-transparent window (see Figure 1), also referred to as a heads-up-display (HUD). The transparent nature of the display enables the user to become aware of the learning event without fully disengaging their thought from the central activity [26].

Experimental Design

To test our methods, our software application, HotKeyCoach (HKC), was made publicly available to the Brandeis University population. Nine users volunteered, installing and using our software for an average of 52 days. The subject base consisted of both students and faculty. During the study, we collected data for all interface actions while HKC was active, providing a detailed view into the activities of each user.

Overall, we observed a total of 38,000 interface actions across 85 unique applications; an interface action is the execution within an application of functionality by either the keyboard or the mouse.

To analyze our data, we grouped all interactions on a per day basis and then quartered the data, with each quarter containing roughly the same number of days. For example,

	A	B	C	D	E	F	G	H
		Applications			Actions			Potential
Subject	Usage	Unique	Ignored	Unique	Ignored	Known	Learned	Learning
5	91%	6	0	19	0	3	2	16
2	81.3%	24	1	105	5	19	21	86
1	76.9%	17	0	83	2	41	3	42
4	76.4%	29	0	110	0	48	9	62
8	66.7%	9	2	133	4	7	13	126
7	52.8%	19	0	127	9	34	11	93
3	43.1%	36	0	124	7	46	7	78
6	30.8%	27	0	200	1	8	3	192
9	9.1%	7	0	15	0	2	1	13
Average	58.68%	19.33	0.4	101.78	4.2	23.11	7.78	78.67
Median	66.7%	19	0	110	2	19	7	78
Std Dev	26.9%	10.6	0.7	57.6	3.3	19.16	6.51	56.13

Table 1. Subject Analysis. Details subject knowledge prior to and at the end of our study. Column A displays the usage level of our software, B, C is the number applications used during the study and the number ignored in HKC. D-G display the total actions performed, total ignored in HKC, percent keyboard shortcut known prior to study and the percent learned. Column H is the potential number of actions the subject can learn.

if a subject used HKC for 40 days, then each quarter would consist of 10 days worth of data.

- Novice (greater than 90% mouse usage)
- Intermediate (10 - 90% mouse usage)
- Expert (less than 10% mouse usage)

From this classification, we determined the subject base consisted of two expert and seven intermediate subjects; one intermediate subject was borderline novice entering the study with 89.5% mouse usage.

Adoption of Our Method

One of our major design goals was to promote learning through minimal distractions to the subject's workflow. If a subject finds our method interferes with her work, she has three possible courses of action:

- Ignore coaching for a single operation
- Ignore coaching for an entire application
- Or disable the software entirely

We discovered that ignoring complete applications (Table 1, column C) was rare and that subjects chose to ignore single operations (Table 1, column E), but not very often, leading us to conclude that if the subjects were in fact using HKC, our method was not interfering with their everyday work tasks.

To determine that the subjects actually used our training method, we calculated the percentage of time our software was used as a ratio of the total install time to actual days of usage. Using a scale from 0 to 100%, where 100% indicates daily usage, and 0% signifies no usage, the results were divided into three levels:

- Greater than 90% (Almost daily usage)
- 50-90% (Regular usage)
- Less than 50% (Intermittent usage)

We determined that 67% of our subjects were regular users (See Table 1, column A) of HKC over the course of the analysis period. Given the anonymous nature of our subjects, we were unable to query the three intermittent users about their experiences with HKC, however analysis of their usage patterns revealed that two of the three intermittent users turned off HKC for a longer than normal period (10 and 22 days) in the middle of the experiment but otherwise demonstrated regular use patterns.

The subject population for our study was self-selected; nobody was under any obligation to use HKC and they were free to disable it as they saw fit. By adopting our method, our subjects presented evidence that our method helped them to learn despite the production bias. HKC did not interfere with the subject's throughput.

Circumvention of the Assimilation Bias

Having determined that our subjects found our method useful, we next analyzed the data to determine if the subjects actually learned. Overcoming the assimilation bias requires the adoption of more efficient methods for performing common tasks. To determine if our subjects acquired more efficient operators for existing knowledge, we identified interface actions that transformed from mouse usage to keyboard usage over the course of the study.

To determine if learning occurred, we first has to identify areas of potential learning, by determining which actions the subjects already knew prior to participation in the study. An action is classified as known if the subject never used the mouse for executing the action over the duration of the study.

Table 1 details the learning potential for each subject. Column B displays the number of unique applications the subject used during the study. Column D indicates the number of unique actions executed per subject and column F shows the number of actions known upon entry to the study. To establish the potential learning for each subject, we subtracted the known actions from the unique actions. For example, subject 3 executed 124 unique actions, where an action is the selection of an interface element, within 36 different applications during the course of the study. The subject demonstrated existing keyboard proficiency for 46 of the unique actions, never once using the mouse for their execution. Based on this analysis, we can determine that subject 3 has the potential to learn a new, more efficient method for performing 78 interface actions.

For this study, we used a liberal definition of potential learning: any observed interface action was classified as a potential learning opportunity. However, this is often not the case, since not every interface action contains an associated keyboard shortcut. Subject 1, for example, indicates a learning potential of 42 interface actions, however in reality, the true learning potential could be much lower.

We classified an interface action as learned if by the fourth quarter of the study, a subject used the keyboard shortcut for a interface action at more than 50% of the time. Figure 2 charts the overall progression of learning across the subject base for each quarter, displaying how over time, most subjects transitioned to the keyboard for their common interface actions. Of the three subjects in the figure that exhibited increased mouse reliance in the final quarter, two of them were also two of the intermittent subjects (subjects 3 and 6), suggesting that they did not have enough opportunity to practice the skills for successful acquisition.

Overall, we determined that the subjects acquired an average of 7.78 new keyboard shortcuts for interface actions during the study (See Table 1, column G); one user acquired 21 new operations. Across all subjects we identified 25 actions at an intermediate stage of learning; the users had some success at using the keyboard shortcut, but their use of the shortcut had not reached the 50% threshold.

Three subjects in the study demonstrated below average learning (subjects 1, 3 and 6). Of those subjects two were intermittent users of the HKC system (subjects 3 and 6), suggesting that while learning is possible at an intermittent level, a more frequent utilization of our system produced more learning. The third below average learner, subject 1, was unique in that he/she had the highest percentage, 65.1%, of already known keyboard shortcuts.

The results indicate that while frequent usage of our application produced better results, even intermittent usage led to the acquisition of new, more efficient means for performing previously known operations. Our subjects demonstrated their ability to continue learning new interface actions despite the assimilation bias, leading them towards mastery.

Subject	Application	1st Half	2nd Half	Improved
1	Finder	3%	0.3%	90%
1	TextEdit	3%	0.9%	70%
2	Finder	29%	9%	69%
2	Safari	47%	11%	77%
3	Finder	8%	4%	50%
4	BBEdit	4%	0.1%	98%
4	Safari	4%	3%	25%
5	Safari	52%	30%	42%
7	Word	34%	22%	35%
8	Mail	46%	34%	26%

Table 2. Sampling of learning within an application. Each row contains an instance of learning for a single subject. The columns indicate the application, the percent of mouse usage for the first and second half of data, and finally the percent improve.

Learning Within an Application

Discovering that the subject acquired an individual keyboard shortcut is a sign of improved efficiency. However to "free up" the sort of mental resources that would enable a user to develop a plan for a more thought out and effective user experience requires the acquisition of multiple keyboard shortcuts.

We analyzed the data for instances of learning within an entire application by aggregating our data per application

for each subject, looking for a decrease in mouse executed actions between the first and last half of observed data. We then filtered the data to instances that consisted of at least 50 actions per half, to ensure that learning occurred. Table 2 displays a sampling of data where subjects demonstrated a shift from the mouse to the keyboard within a single application.

The results in Table 2 highlight the effectiveness of the HKC training method across our subject base.

For subject 4, we collected a total of 7150 actions for the application BBEdit. During the first half of the study, he/she relied on the mouse for only 4% of the 397 performed interface actions. During the second half of the study, she/he increased the usage of BBEdit to 6753 performed actions and only relied on the mouse only 0.1% of the time, showing a 98% improvement in keyboard proficiency over the course of our study.

In another instance, subject 1 relied on the mouse for only 3% of his/her interactions with *Finder* during the first half, demonstrating strong keyboard proficiency. Using HKC allowed this subject to decrease the reliance even more, to just 0.3% mouse utilization in the second half of the study, an improvement of 90% over the course of the study.

Both of these examples demonstrate learning at the expert skill level. In Table 2, there are five instances where the subjects entered the study using the keyboard for 90% of their interface work, yet they still improved their keyboard shortcut proficiency by at least 25% over the course of the study.

The results in Table 2**Error! Reference source not found.** also indicate that intermediate application users also improved their overall keyboard proficiency. Subject 5 entered the study using the mouse 52% of the time for all interactions. By the end of the study, she/he had decreased their reliance on the mouse to 30%, an improvement of 42%. Subject 8 also demonstrated a 26% improved learning for the application Mail during the study, having started with a 46% reliance upon the mouse.

Learning Beyond Operations

One of the authors of this paper is a long-term user of HKC. During preparation for lectures he often copies images from a PDF document using the PDF viewer *Preview* into a Microsoft *PowerPoint* presentation, using the screen capture utility, *Grab*.

Prior to using HKC, he would ask one of the undergraduate assistants in the department office to scan figures out of the printed version of the PDF so he could then paste the figures into his class presentations. The same result can also be achieved without scanning, but it requires a substantial amount of interface work. Using *Grab*, he captures the desired figure, places it on the clipboard, switches to *PowerPoint*, creates a new slide, pasting in the captured image. He navigates between *Grab, Preview*, and *PowerPoint* using the *command-tab* keyboard shortcut, executes his cut and paste operations, closes windows. Performing all operations without moving his hands from the keyboard. When he forgets the keyboard shortcut for *Grab* that allows him to select a subsection of a window—as opposed to grabbing the whole window or all the windows for an application he can rely on HKC to remind him.

The internalization of a skill is not just an elimination of the external components; it is a redistribution of the internal and external components within a process [27], changing the way the individual performs and activity.

Once he begins to learn to use some keyboard shortcuts, and keeps his hands at home on the keyboard, a new process for transferring figures between documents becomes available, one that is more efficient than the scanning-based method. As a result of acquiring keyboard shortcuts, a new more efficient method evolves. A task constructed on the fly that once required multiple transitions between the keyboard and the mouse, or offline work by an office assistant, is now rapidly done with the keyboard.

CONCLUSION

During the course of computer-mediated work, novice and expert users differ in their expectations of and interactions with an applications interface. Novice users expect an easy to learn and easy to use interface that allows them to explore the interface without serious repercussions. They navigate an interface visually and rely on the design to direct them towards the appropriate methods for achieving their goals. Expert users however, expect efficient interfaces that reduce the work to use the application, reducing the interference it presents so that the translation of thought to action is minimized. Because of their deep structural knowledge of the application, expert users are able to quickly form goals and construct action sequences for achieving those goals [1] and thereby exhibit minimal reliance upon the visual elements of the interface.

Most users rarely attain expertise with an application. As users transition beyond the novice level, they still place a heavy emphasis on the visual characteristics of the user interface, preventing them from leveraging the functionality designed to improve their efficiency as they progress from intermediate to expert users. This failure to develop the required structural knowledge of the interface is the result of the active user paradox.

In our research, we have developed the HKC tool. It helps the user move beyond the visual characteristics they initially rely on and continue learning new interface methods and operations. The HKC tool allows the user to maintain complete control over the training process.

There are many opportunities within a workflow for training. The HKC tool converts these opportunities into learning events that are relevant to the user and require

focus-shifts rather than interruptions for learning. By only coaching where it is relevant and keeping the amount of new information presented to the user to a minimum the user is not continuously removed from her central work task. It is for these reasons that the HKC tool can circumvent the active user paradox.

The study presented in this paper provides evidence of the effectiveness for the HKC learning model as a training method that enables users to circumvent the active user paradox. Sixty seven percent of the subjects who participated in the study were regular users of HKC. The subjects demonstrated an average increase of 58% in keyboard shortcut knowledge over the duration of the study. All subjects displayed instances of learning for both single interface actions and for complete applications. Overall, the subjects acquired an average of 7.78 new keyboard shortcuts for interface actions, with one subject demonstrating the acquisition of 21 new keyboard shortcuts. Having entered the study knowing only 19 shortcuts, this was an improvement of 110%. The results indicate that while frequent usage of HKC produced better results, even intermittent usage led to the acquisition of new, more efficient means for performing previously known operations. The level of the subject upon entry to the study also had no effect on overall learning, even subjects with expert-level keyboard proficiency continued to increase their knowledge of keyboard shortcuts with the help of the HKC application.

FUTURE RESEARCH

Most applications do not assign keyboard shortcuts for every operation. It is not uncommon for a user to have an interface action that they do frequently but has no associated keyboard shortcut. Most applications and operating systems provide mechanisms that allow users to create custom shortcuts for various functions, however this feature is rarely used. This kind of *user adapted interaction* makes for much more efficient usage of the application. A natural extension of the HKC tool is to collect data on the user's menu actions that have no corresponding keyboard shortcut. If the user does the action often enough HKC will initiate adding a keyboard shortcut, which it can subsequently train the user to perform.

Once an individual acquires the keyboard shortcut for an operation, will they remember it after a period of infrequent usage? If they have forgotten it, what will it take re-learn the shortcut? We have observed instances in our data that users typically only need a single reminder to refresh their memory after a period of non-usage. On other occasions, when a user has not used an application for an extended period of time, the user forgets many of keyboard shortcuts that she knows how to use. In situations like these, a heads-up display that shows to the user the keyboard shortcuts for her most frequent actions for that application may prove to be useful. We will investigate how to introduce this kind of information given the HKC model of training.

There are other areas of training that we plan to explore leveraging the HKC model of learning. In each of these cases, the HKC tool can introduce learning events into the flow of the user's activity as focus shifts rather than interruptions, encouraging the user to learn. In each of these cases, the HKC tool maintains the user as the locus of control.

REFERENCES

[1] Wu, J. *Accomodating both Experts and Novices in One Interface. Universal Usability Guide. Department of Computer Science, University of Maryland*. City, 2000.

[2] Chi, M. T. H., Feltovich, P. J. and Glaser, R. Categorization and Representation of Physics Problems by Experts and Novices1981).

[3] Mayes, J. T., Draper, S. W., McGregor, A. M. and Oatley, K. Information flow in a user interface: the effect of experience and context on the recall of MacWrite screens. *Proceedings of the Fourth Conference of the British Computer Society on People and computers IV table of contents*1988), 275-289.

[4] Payne, S. J. Display-based action at the user interface. *International Journal of Man-Machine Studies*, 35, 3 1991), 275-289.

[5] O'Hara, K. P. and Payne, S. J. Planning and the user interface: The effects of lockout time and error recovery cost. *International Journal of Human-Computers Studies*, 50, 1 1999), 41-59.

[6] Carroll, J. M. and Rosson, M. B. 5 Paradox of the Active User. *Interfacing Thought: Cognitive Aspects of Human-Computer Interaction*1987), 31.

[7] Bhavnani, S. K., Reif, F. and John, B. E. Beyond command knowledge: identifying and teaching strategic knowledge for using complex computer applications. *Proceedings of the SIGCHI conference on Human factors in computing systems*2001), 229-236.

[8] Shneiderman, B. *Designing the user interface: strategies for effective human-computer interaction*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1992.

[9] Lane, D. M., Napier, H. A., Peres, S. C. and Sándor, 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*, 16, 2 2005), 133-144.

[10] Benner, P. From Novice to Expert. *The American Journal of Nursing*, 82, 3 1982), 402-407.

[11] Dreyfus, S. E. and Dreyfus, H. L. A Five-Stage Model of the Mental Activities Involved in Directed Skill Acquisition1980).

[12] Nimwegen, C., Oostendorp, H., Schijf, H. J. M. and Burgos, D. The Paradox of the Assisted User: Guidance Leads to more Shallow Behavior2005).

[13] Larkin, J. H. Display-Based Problem Solving. *Complex Information Processing: The Impact of Herbert A. Simon*1989).

[14] Anderson, J. R. *Acquisition of Cognitive Skill*. Dept. of Psychology, Carnegie-Mellon University, 1981.
[15] Anderson, J. R. *Rules of the Mind*. Lawrence Erlbaum Associates, 1993.
[16] Payne, J. W., Bettman, J. R. and Johnson, E. J. *The Adaptive Decision Maker*. Cambridge University Press, 1993.
[17] Reimann, P. and Neubert, C. The role of self-explanation in learning to use a spreadsheet through examples. *Journal of Computer Assisted Learning*, 16, 4 2000), 316-325.
[18] Norman, D. A. *The invisible computer*. MIT Press Cambridge, MA, USA, 1998.
[19] Bederson, B. B. *Interfaces for staying in the flow*. ACM Press, 2004.
[20] Carroll, J. M. The Nurnberg funnel: designing minimalist instruction for practical computer skill1990).
[21] VanLehn, K. Cognitive skill acquisition. *Annu Rev Psychol*, 471996), 513-539.
[22] VanLehn, K. Rule Acquisition Events in the Discovery of Problem-Solving Strategies. *Cognitive Science*, 15, 1 1991), 1-47.
[23] Shroyer, R. Actual Readers Versus Implied Readers: Role Conflicts in Office 97. *Technical Communication*, 47, 2 2000), 238-240.
[24] Bødker, S. Applying activity theory to video analysis: how to make sense of video data in human-computer interaction. *Context and consciousness: activity theory and human-computer interaction table of contents*1995), 147-174.
[25] Apple Computer Inc. *Introduction to Accessibility Overview*. City, 2007.
[26] Harrison, B. L., Ishii, H., Vicente, K. J. and Buxton, W. A. S. Transparent layered user interfaces: an evaluation of a display design to enhance focused and divided attention. *Proceedings of the SIGCHI conference on Human factors in computing systems*1995), 317-324.
[27] Kaptelinin, V. and Nardi, B. A. Acting with technology: activity theory and interaction design2006).