Research in Learning Technology
Vol. 26, 2018

ORIGINAL RESEARCH ARTICLE

Inline training: a technique for continuous, within-task learning

Brian Krisler[a*] and Richard Alterman[b]

[a]*Raytheon BBN Technologies, Information and Knowledge Technologies, Cambridge, MA, USA;* [b]*Department of Computer Science, Volen National Center for Complex Systems, Brandeis University, Waltham, MA USA*

(Received 31 July 2017; final version received 5 February 2018)

As software continues to grow in power and complexity, frequent on-the-job training is essential to maintain a proficient and productive skill set. However once a base operational skill set is attained, software users rarely continue to become proficient with the tools they use on a daily basis. This lack of proficiency results in the frequent occurrence of workflow interruptions due to the continued locating and re-locating of the operators required to perform both new and routine tasks. Aids such as reference cards and application help systems exist to make the user aware of efficient methods for task completion; however, these resources are seldom used. This study presents a new and efficient approach to help software users continue to learn about the tools they use to complete their work. This new approach to learning, called inline training, leverages common workflow interruptions to facilitate the discovery of new application knowledge. At issue is fitting the amount of work necessary to use the trainer into the already occurring interruption window. By understanding the amount of within-interruption work tolerated by the user, including an inline trainer within the window, promotes a deeper understanding of the application, resulting in a more efficient workflow.

Keywords: training; continuous learning; learning; software training; interruptions

Introduction

In the modern technological landscape, the tools required to perform everyday tasks change at a rapid pace. Therefore, today's workers must adjust to improvements to the applications that support their work-related activities (Boothby, Dufour, and Tang 2010). The drive for increased productivity, mixed with the pace of technology evolution, makes essential the development of learning technologies for users to remain competitive in the modern workplace (Gravill and Compeau 2008).

Although the skills learned in the classroom are an important starting point, in this rapidly changing environment, continuous learning is imperative. The research reported in this article explores *inline training* as a training style that weds the everyday use of a technology with incremental learning and training, which is directly relevant to the activities of the user (Krisler 2014).

This article closely examines how users interact with the tools they use on a daily basis. Initially, they develop a working knowledge of the interface. With this, users can find their way around the interface to complete routine tasks. However, the necessity to get work done prevents the user from progressing beyond a basic understanding of

*Corresponding author. Brian.Krisler@Raytheon.com

the application (Carroll and Rosson 1987). This failure to develop proficiency results from a strong dependence on basic operators. This limited functional knowledge is the cause of workflow interruptions. These slowdowns manifest themselves through searching for known or 'hoped for' operations.

Presented here is an analysis of an experiment designed to leverage the naturally occurring workflow interruptions in normal everyday activities as learning opportunities. This inline trainer enables an everyday user to leverage common workflow interruptions as learning opportunities, that is, when the user interrupts his/her main task to focus on how to use an application. Instead of perform and forget, they learn and perform.

A between-participants experiment was designed where three groups completed multiple task sets over a 2-day period. The participant actions were closely examined from the time the flow of activity was disrupted until the time the primary task was resumed. The data show that the interruptions exist and that the users are willing to leverage an inline during those interruptions.

The experimental analysis will focus on the ability of the participants to learn within two types of interruptions: (1) *Insufficient knowledge* interruptions, where the user disrupts the task to search the menus for a solution and (2) *Re-location* interruptions that occur when the user must re-locate previously found or assumed to exist operators.

For most users, these two types of application-related interruptions are frequent and inescapable, presenting opportunities for introducing continued learning. This opens new opportunities for developing training platforms that would encourage the end-user to explore and adapt the application and learn alternate and better ways to complete common tasks.

Background

Once a user attains an intermediate level of application knowledge, training has stopped and his/her focus is now directed towards productivity. Ideally, however, training would never stop. With each new version, the functional set of a software application increases (Findlater and McGrenere 2004), and to remain effective, the user's knowledge of the tool should also increase. Through continued learning of the ever-increasing operator set of an application, more advanced functionality could be leveraged and more efficient work patterns would emerge.

However, limitations in training prevent all but the most dedicated users from effectively attaining proficiency. Once a sufficient level of working knowledge is established, a training gap occurs, where current methods fail to encourage the intermediate user to advance (Gupta, Bostrom, and Huber 2010). Because of this limitation in training, most users tend to *plateau* in their learning (Gray 2017; Olfman and Mandviwalla 1994). Ideally, training would provide the necessary mechanisms to elevate users above this plateau by allowing them to ration and prioritize their learning through aids, such as the use of a learning cache.

Caching is a mechanism that allows users to store, in an easily accessible location, operations they recently discovered but have not yet mastered. Caching addresses the issue of prioritized learning (Buckler 1996), where only the operators relevant to the current work task are part of the learning agenda. When there is too much content to learn, preferential ordering of the material is essential for successful acquisition.

Multiple techniques have been explored for preferential ordering of the operators displayed in an interface, such as split menus (Sears and Shneiderman 1994), which are aimed at easing relocation through reordering.

The wholesale application of this technique is unfortunately fraught. The development of the location of the operator in the menu occurs through the process of implicit learning (Reber 1989), where over a period of time, the structure of the environment is learned and exploited to guide the search effort. If the menu structure loses its static nature, the ability to fully develop the visual-spatial location of the operator is lost. If the whole menu becomes dynamic, over the long term, the cost of re-finding operators will remain constant. To address the spatial consistency issue, Findlater *et al.* (2009) have developed the concept of ephemeral adaptation; while improving overall usability of the interface, this technique still relies on searching as a strategy for locating functionality.

Reference cards are another form of preferential ordering that provide the ability to quickly recall information. Reference cards (Goodwin 1994; Haramundanis 2014) have become a common approach in helping users quickly find functionality and in assisting the transition from menus to keyboard shortcuts.

These techniques reflect the designers' views on what are basic and/or important operators for the user to know about. However, once the user becomes comfortable with an application, the ability of the designer to effectively predict and stage the next set of useful operators diminishes as the tool use becomes more specialized. Here, the prioritization task should transfer from the designer to the user.

Another approach to continuous training is *Caching*. Caching, a form of user-controlled prioritization, is more effective for advanced users because it provides a personalized learning experience aligned with the needs and requirements of the user (Eagan and Stasko 2008). Although an effective technique for continuous learning, current interface designs do not facilitate caching. Instead, a cache used to reduce the impact of an interruption must be explicitly stored by the user (Minassian, Muller, and Gruen 2004). An alternative is to incorporate a cache as a component of an inline trainer, where it can be efficiently integrated into the users' workflow.

Approach

Interruptions are common (Digmayer and Jakobs 2014). They are so common, in fact, that it has been observed that workers switch tasks due to an interruption about every 12 min (González and Mark 2004). Interruptions vary (McFarlane and Latorella 2002) in size and are often detrimental (Mark, Gonzalez, and Harris 2005), with the worker easily losing their train of thought and over 40% of interrupted tasks (O'Conaill and Frohlich 1995) never getting resumed.

To avoid this issue, research has been conducted on training technologies designed to reduce the duration of workflow interruptions (Matejka, Grossman, and FitzMaurice 2011).

However, interruptions are not always a hindrance (Jin and Dabbish 2009) and can actually aid task completion. Furthermore, by leveraging an already occurring interruption, the overall cost of the interruption is reduced (Iqbal and Bailey 2007). The question addressed within this study is as follows: during frequent and expected interruptions, are users willing to do the additional work necessary to increase their understanding and skill with the applications used on a daily basis?

This study will focus on evaluating a learning technology designed to leverage two types of application-related interruptions as training opportunities:

Insufficient Knowledge: Insufficient application knowledge manifests through the search of a new operator in the interface. Discovering an operator in a menu is a process of model matching (Norman 2013). Menus facilitate matching through structured organization (Hollands and Merikle 1987). However, when an operator is not easily located, an interruption occurs.

Re-location: Once an operator has been discovered, there is no guarantee that it will be easily re-discovered (Kim and Ritter 2013). When the direct path to attaining a goal is not the most efficient, a *knowledge error* occurs (Zapf *et al.* 1992). Not recalling the location of an operator on a menu introduces unnecessary searching.

This study will show that during these interruptions, participants will leverage an inline trainer to improve their workflow by (1) learning the application's underlying conceptual model, (2) discovering new operators, and (3) caching knowledge for future learning.

To evaluate how users interact with their tools, and to assess their overall willingness to leverage interruptions as learning opportunities, interruption-related data were collected of participants completing a series of task sets over a 2-day period using the core OS X applications: Finder, Mail, and Safari.

To effectively analyze the results of the experiment, tools were created that enabled the recording of user actions and generated a chronological event transcript for each participant. By recording data at this level, it was possible to identify when each participant struggled and further determine what they were doing during each observed interruption.

Experimental design

The experiment design consisted of three task sets replicating typical, daily application interactions within the categories: working with files, using a web browser, and interacting with an email client.

For each task set, participants were asked to complete one or more objectives, which, based on the skill level of the participant, could be completed using a variety of approaches. For example, in one task, participants were asked to locate information on the Internet and then email a link to the page. This could be accomplished through simple copy and paste, or through the use of a special *mail link* operator in the web browser. Here, the experiment design tested how participants might avail themselves of a simpler, but lesser known path toward emailing a web link. We established three test conditions: (1) a control group which would be left to its own devices, (2) a reference card group offered a standard, paper-based summary of features, and (3) an inline trainer group given access to the Learn From Friends (LFF) (see section 'The LFF trainer') inline training tool.

All participants completed the first task set without any additional training. This was conducted to establish a baseline comprehension against which we were able to analyze. Prior to the second task set, all participants received a training manual, which summarized the applications' more advanced features. This traditional self-help meant that all participants had straightforward access to the answers for all the experiment's tasks.

The experiment was completed over the course of two consecutive days. The objective here was the testing of retention: could participants remember any of the new features to execute slight variations on the previous day's tasks.

The participants relied on the standard application interfaces to complete the experiment. No modifications were made to Finder, Mail, or Safari. At any point in the experiment, participants were also permitted to use the applications' built-in help systems. They were also free to access information from the Internet, that is, using Google to find out how to solve a specific task. Therefore, in terms of self-based learning, our participants had unfettered access through manuals, online help, or the Internet. In addition to these resources, participants in the reference card group were given a pen and a single page printout of keyboard shortcuts for the three primary applications. This sheet listed the application operators and associated keyboard shortcuts typical of all software reference cards. During the experiment, the participants in this group could mark up, highlight, or annotate the reference card as needed. Participants in the training group were presented with a 10-min screencast tutorial demonstrating the basic features and functionality of the training tool.

Participants

A total of 21 male and female students participated in the experiment. The participants had a mean number of college years attained as 3.22 (SD = 1.22). Prior to the experiment, each participant completed a survey assessing his or her own skill level and daily computer usage. A majority of the participants rated themselves as 'much experienced' (i.e. they classified themselves as having vast computer experience). They all stated they used a computer 'several times a day'.

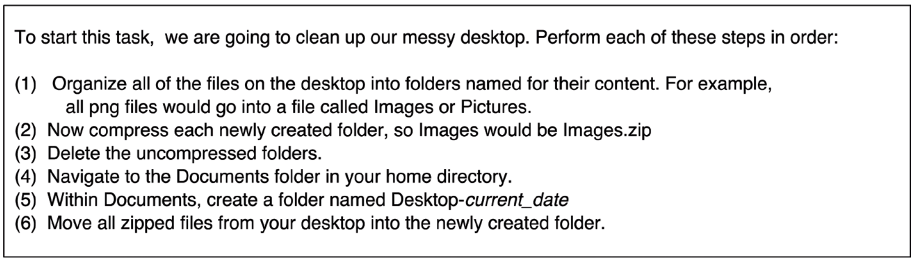
On the first day of the experiment, each participant was briefed on the procedure and randomly assigned to one of the three groups: control, reference card, and LFF training group.

All of the participants were compensated for their participation. On completion of the first day, the participants were given $5, and on the follow-up day, another $10 for a total of $15. Twenty-one participants participated, with each group containing seven participants. Two of the participants in the LFF group did not return for the follow-up day.

Tasks

The experiment consisted of 25 tasks that could be performed through a mixture of standard or lesser-known interface operations. The expectation was that all of the participants would exhibit some inefficient behaviors in how they completed the work.

Each set of tasks was designed to capture how participants worked outside of a typical experiment's boundaries where they were free to complete the tasks as they wish, in the order they preferred. Many of the same operators were frequently required within and across tasks. This re-introduction of operators created a repetitive pattern with the intent of leveraging new knowledge. Participants were also instructed to order the tasks as they saw fit, tackling easier ones first if needed. And participants were told they could abandon a task if it was too hard. The experiment placed minimal constraints on the participants' workflows.


To start this task, we are going to clean up our messy desktop. Perform each of these steps in order:

(1) Organize all of the files on the desktop into folders named for their content. For example, all png files would go into a file called Images or Pictures.
(2) Now compress each newly created folder, so Images would be Images.zip
(3) Delete the uncompressed folders.
(4) Navigate to the Documents folder in your home directory.
(5) Within Documents, create a folder named Desktop-*current_date*
(6) Move all zipped files from your desktop into the newly created folder.

Figure 1. Task number one as presented to each participant in the experiment. In this task, the participant was asked to clean up a messy desktop.

Figure 1 shows an initial task from set one, which established the baseline knowledge of the participants. It was a clear multi-step process for moving and creating new files. This was typical of all tasks throughout the three-step process.

Apparatus

All participants used one of two 15-inch MacBook Pro's running OS X 10.8 Mountain Lion with a high-speed wireless Internet connection. The task bar in the operating system contained only the three critical applications for the experiment: Finder, Safari, and Mail.

Each laptop contained the experiment training software. For the participants not in the LFF group, the visual windows for the trainer were disabled and the trainer ran in the background, collecting data.

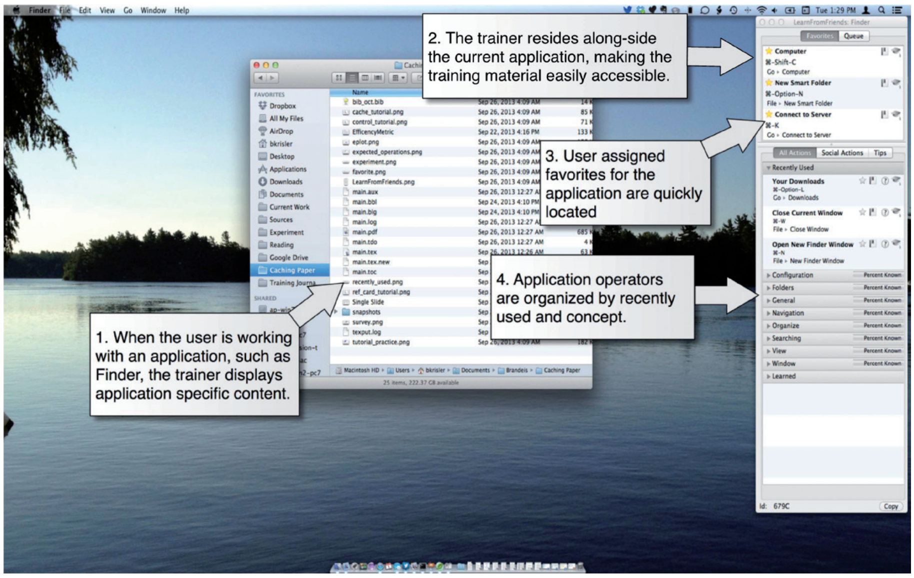
The LFF trainer

The LFF inline trainer is a custom developed OS X application designed to run in parallel to the main application (i.e. Finder or Safari), providing the user with the ability to quickly navigate between task-work and training (see Figure 2). The intent of the trainer is to address the many shortcomings of existing training while leveraging the possible opportunities inherent in workflow interruptions.

Using a split pane interface, the lower pane, consisting of multiple tabs, offered a novel organization of all the application's operators. Here, a concept hierarchy (Figure 3a) was created around application-specific actions like finding, searching, and organizing. The concept hierarchy differs from the existing menu structure, in that the operators are organized by function or concept, unlike the existing menu structures, where a standardized cross-application consistency is maintained. For example, the Find operator, which, in the LFF trainer, is located in the 'Searching' concept, is more typically located in an application's 'Edit' menu.

We grouped each application's operators into buckets for three reasons: (1) it simplified the discovery process for new features, (2) it allowed the finding of similar or complementary features, and (3) re-finding an item via the recently used tab (Figure 3b) offered a supportive path to item reuse.

These tabs became expanded lists of operators. Each included the operators name, the keyboard shortcut, the application menu path, and fast access to the associated help text that did not require the launching of the built-in help system itself.


1. When the user is working with an application, such as Finder, the trainer displays application specific content.

2. The trainer resides along-side the current application, making the training material easily accessible.

3. User assigned favorites for the application are quickly located

4. Application operators are organized by recently used and concept.

Figure 2. When displayed, the trainer can appear alongside the primary application, allowing quick access to the training.

The second tab in the lower pane, Social Actions (Figure 3c), offered another perspective to an application's operators. This pane displayed operators rank based on their popularity among a social group, providing the user insight into how others were finding value in the application.

The icon of a grayed-out star (Figure 3d) was another important design feature. Appearing to the right of the operator name, the star could be clicked, turning it gold, and marking its associated operator as a 'favorite'. This moved the operator to the upper pane (Figure 3e), which allowed the user to customize a list of features he/she was most interested in. This customized pane called Favorites provided the user, via a quick glance, access to the necessary information for locating the operators they are in the process of learning.

As users performed experiment tasks, which required switching between applications, the content of the trainer changed correspondingly to match the current application. By design, the LFF trainer did not provide any information not already accessible through other user interface displays.

Procedure

Each participant was presented the following instructions:

- They should complete the tasks to the best of their abilities.
- They could use any application on the computer for completing the task.
- They did not need to perform the tasks in any specific order.
- If they got stuck, they were free to use any source of help (i.e. Internet or application help) to find a solution.
- If they could not complete a task, they were free to skip it and move on.

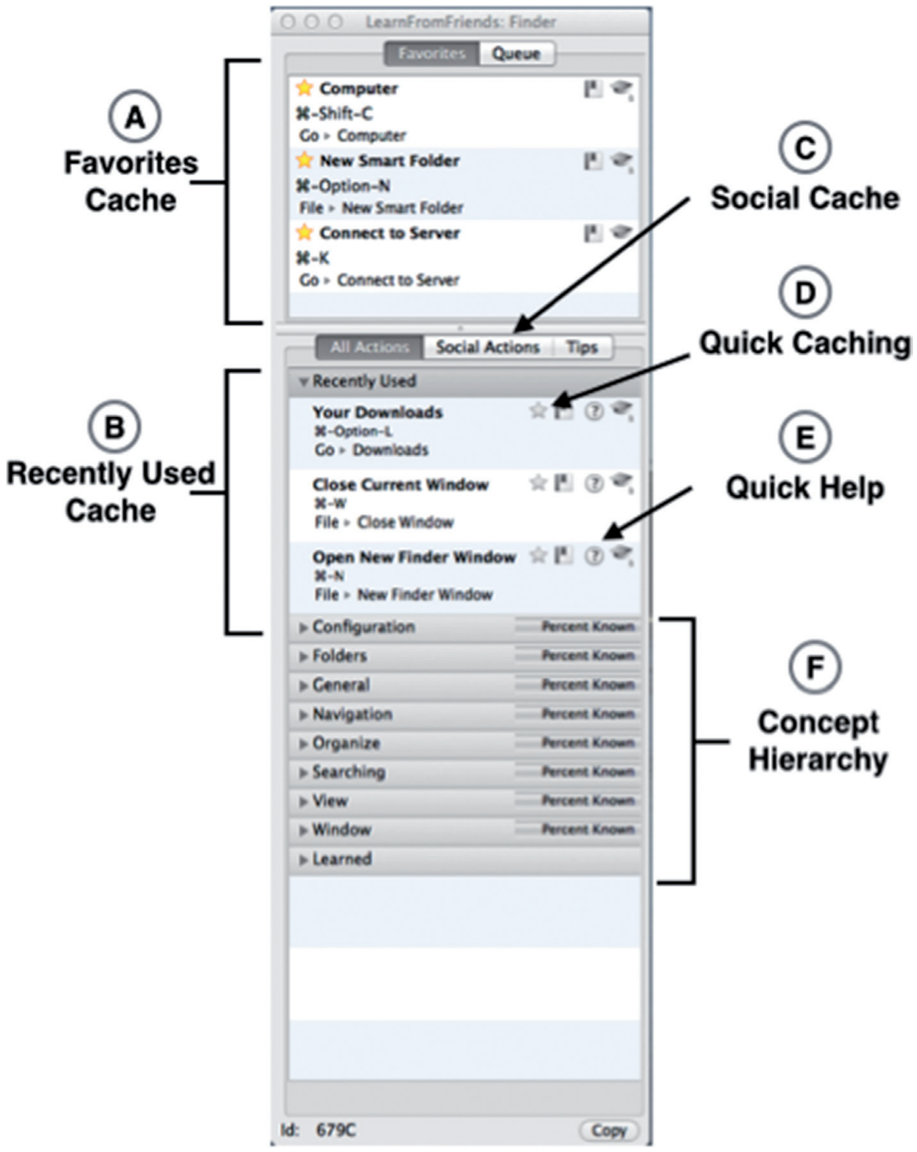
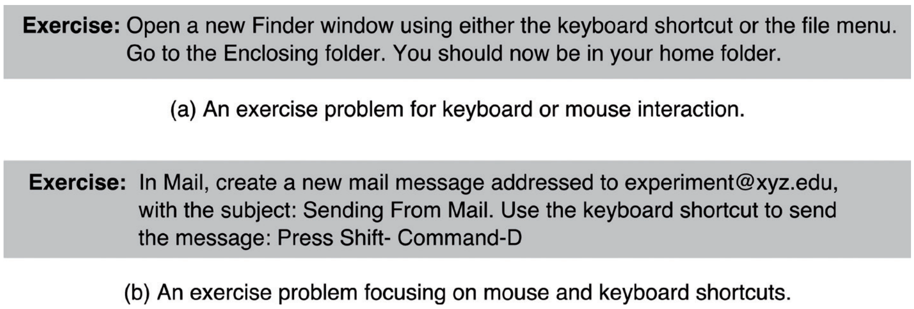


Figure 3. The trainer is composed of multiple mechanisms designed to leverage opportunistic interruption-based training.

After completing the first assessment, the participant was handed a tutorial designed to teach efficient usage of the three core experiment applications. This manual was roughly six pages in length and consisted of both descriptive text and exercise callout boxes (Figure 4) highlighting some useful (and occasionally obscure) operators. The participant was given as much time as required to read the manual and perform the practice exercises. Upon completion of the tutorial, the participant started the second task set by clicking a bookmark in the web browser. The second task set was disseminated as a series of emails. The participant was told to expect nine emails during the second task. At the completion of the second task, the participant was compensated for his/her



Exercise: Open a new Finder window using either the keyboard shortcut or the file menu. Go to the Enclosing folder. You should now be in your home folder.

(a) An exercise problem for keyboard or mouse interaction.

Exercise: In Mail, create a new mail message addressed to experiment@xyz.edu, with the subject: Sending From Mail. Use the keyboard shortcut to send the message: Press Shift- Command-D

(b) An exercise problem focusing on mouse and keyboard shortcuts.

Figure 4. Exercise box examples from the tutorial. Each participant was presented with a tutorial containing practice exercises. The exercises reinforced the operations introduced in the tutorial. Some exercise boxes allowed the participant to choose their preferred interaction method, while others specifically taught the keyboard shortcut for the operation.

Table 1. All recorded events consisted of nine fields to facilitate data analysis.

Label	Description	Example
TID	The task id	1.1
PID	The participant id	708A
DAY	The day of the observed event	1
TIMESTAMP	The timestamp of the event	21113100500
APPLICATION	The active application for the event	Finder
ACTION	The observed action	File/New Finder Window
INTERFACE	The event interface (Keyboard:0, Mouse:1)	1
DETAILS	Auxiliary information about the event	Marked Favorite
GROUP	The experiment group of the participant	Control

participation. On the following day, participants in the training group had the state of the trainer preserved, ensuring that any staging, such as the caching of operators, was available for the final task. For the participants in the reference card group, they were handed back their reference card prior to starting. Like task set two, each task arrived via email. The participant was told to expect nine emails, one per task. At the completion of the final task, the participant was compensated for the day's participation.

Data collection

To evaluate how each participant interacted with the system during the experiment, an event logger was developed that recorded experiment interactions. Each observed event contained nine fields to facilitate offline analysis (see Table 1).

Table 2 is a snippet from a collected transcript. In this block, a sequence of actions performed by the participant 387C during task 1.1 is presented. In *line 1*, the experiment proctor launched the event-recording tool. Five minutes later, the application Finder became the primary application, and the File menu was opened (*line 2*). In the File menu, the New Folder operator was selected with the mouse (*line 4*).

By analyzing the transcripts, we were able to recreate each participant's actions and discover how they searched, found, and executed the required operators for completing each task.

Table 2. A recorded experiment transcript segment showing the opening and closing of a menu followed by a keyboard performed New Folder operation.

S. No.	TID	SUBID	DAY	TS	APP	ACT	INT	DETAIL	GROUP
1	SETUP	387C	1	20121025140538	LFF		2	LAUNCHED	Ctrl
2	1.1	387C	1	20121025141028	Finder	MENU_OPENED	2	File	Ctrl
3	1.1	387C	1	20121025141028	Finder	MENU_CLOSED	2		Ctrl
4	1.1	387C	1	20121025141031	Finder	File/New Folder	1		Ctrl

Overview of the results

Analysis of the results showed that when new application knowledge was required to complete a task, participants tended to spend their time focused on searching and re-searching the application menus. When it came to frequent reuse of an individual operation, participants tended to work with the knowledge they had, instead of trying to operate more efficiently, even when resources where available that could have reduced their workload. However, the addition of an inline trainer demonstrated effectiveness in quickly providing the knowledge necessary to proceed with the task. Participants used the inline trainer to discover and better understand the operators and set up future training by caching operator knowledge that they wanted to learn. These results demonstrate that incorporating learning into the workflow, via a learning platform that is concise and task-targeted, is an effective means to encourage continuous learning.

Menus compensate for a lack of knowledge

The experiment asked participants to perform work, which required modifications to their existing routines. The data showed that when users did not have enough application knowledge to proceed, they fell back to the menus for a solution. This menu dependence resulted in extensive and ineffective searching.

To determine the extent of menu dependency, we analyzed the exact steps performed by each participant during the experiment and created a search cost measure.

Search cost measured the amount of extra work expended by a participant to perform an operator by computing the difference between the observed actions and the expected actions. The example shown in Table 3 compares the efficient menu-based execution of the Clean Up By Kind operator versus an observed interaction, where the user was unsure. The efficient search required three actions compared to the observed with five actions, resulting in a search cost of 2 (5 observed – 3 expected).

Table 3. A search cost was used to measure the amount of extra work performed per action by comparing the optimal approach to the observed approach.

Search	
Optimal	Observed
View	File
Clean Up By	Edit
Kind	View
	Clean Up By
	Kind

A search cost analysis over the entire data set showed that extra work was required to perform each operator (M = 3.68, SD = 4.93). Further analysis revealed this extra work translated into time inefficiencies. The amount of time spent searching the menus compared to the total experiment time revealed that an average of 9.17% of the experiment time was spent searching the menus.

New operator knowledge comes from the menus

When a participant did not have enough knowledge about an application to proceed with a task, she would rely on menus to provide a way forward.

Trying to discover a way to convert all of an email's text to uppercase, the transcript in Table 4 documents the work of one participant relying on the menus as the primary approach to acquiring new knowledge. During the search for the Make Upper Case operator, the participant opens and re-opens the same menu (lines 5 and 8). When a plausible operator is found, by trial-and-error the participant discovers which operator produces the desired results (line 6 and line 12). In this example, the process of locating and performing a single operation required 12 interface events over the course of 33 s.

Once an operator has been found, finding on a second occasion may or may not be easier to do. For example, one participant, while performing a desktop cleanup task, was observed using the New Folder operator three times over the course of a minute and a half. Often, rediscovering the operator the second and third time took more work than it did the first time.

Initially, the operator was found in just 4 s (search cost = 1). By the third time, the search cost increased to 25, with the participant requiring 10 s to find the operator. Difficulties in relocating previously discovered operators occurred frequently in the transcripts.

Existing operator knowledge drives usage

When a new task is encountered, a user will try to apply existing knowledge to achieve their goal, cobbling together basic operations rather than seeking out a better and

Table 4. New knowledge is attained through searching.

	Observed user action		
	Time (sec)	Menu Search	Operator Performed
1.	0	Edit	
2.	5	Attachments	
3.	6	Find	
4.	7	Spelling and Grammar	
5.	8	Substitutions	
6.	12		Text Replacement
7.	17	Edit	
8.	19	Substitutions	
9.	19	Transformations	
10.	20	Speech	
11.	20	Transformations	
12	33		Make Upper Case

This process of knowledge acquisition is both time-consuming and prone to error. In this example, the participant opens multiple menus in the search of the correct operator. During this search, trial-and-error is used to test out potential operators

Table 5. One type of failure observed in the experiment was the failure to locate a more efficient operator to complete a task.

	Observed user action		
	Time (sec)	Menu Search	Operator Performed
1.	0		Copy
Switched to mail application			
2.	2		New Message
3.	5		Paste
4.	16		Send

a) When a procedure for performing a task is successful, it often becomes the default approach.

	Observed user Action		
	(sec)	Menu Search	Operator Performed
1.	0		Mail Link To Page
2.	4		Send

b) Learning a more efficient operator can simplify a procedure

more efficient method (Fu and Gray 2004). The frequent reuse of these kinds of procedures composed of basic operations is the basis of inefficient work habits.

Table 5 presents a transcript of a participant relying on existing knowledge to complete the task of emailing a link to a webpage. Safari has a menu operator for performing this task that was introduced in version 2.0. Prior to this operator, sending a link involved copying and pasting the link from the browser into a new email message. However, by using an operator that simplifies this multi-step process (Table 5), much less user work is required. Had the participant found this operator, the search cost would have been zero instead of six.

The transcripts show, for example, that basic operators like Copy and Paste were used in multiple situations to compensate for the user's lack of knowledge of an existing more powerful operation that could significantly reduce work. An over-reliance on these procedures deterred discovering new and more efficient methods.

To measure the extent of this over-reliance on general procedures, a task operator analysis was performed. Table 6 lists all of the operators whose use was expected during the experiment had each task been completed using the most efficient approach possible. The *expected* column lists the total number of expected occurrences over the course of the experiment, per participant. For example, there were five tasks that, if completed optimally, would have used the Mail Contents of This Page operator. The *observed* column presents the average observed count of each operator per participant during the experiment. The average observed count for Mail Contents of This Page was 4.05, demonstrating that this operator was under-utilized during the experiment. The *difference* column is the expected minus the observed. This calculation highlighted both over- and under-utilization per operator.

The results in Table 6 demonstrate the extent of the over-reliance on general procedures across the participant base. This is apparent with the overuse of the Copy and Paste operators. If each task were performed optimally, these operators would have only been required five and three times, respectively; however, their actual usage exceeded that, with actual usage being 16.38 and 16.21.

Table 6. Experiment operations and the number of times they were expected to be observed compared to the mean observed value across all participants.

Operator	Expected	Observed	Difference
		(mean)	
Google Search	13	1.80	11.2
New Message	5	2.44	2.56
Save As	3	2.00	1.00
Mail Contents of This Page	5	4.05	0.95
Mail Link to This Page	4	1.58	0.94
Compress	6	5.29	0.71
Center	2	2.00	0.00
Insert Bullet List	1	1.00	0.00
Downloads	1	1.00	0.00
Make Upper Case	1	1.00	0.00
Forward as Attachment	1	1.00	0.00
Open Location	3	0.00	0.00
Find	2	0.00	0.00
New Folder with Selection	4	0.00	0.00
Paste as Quotation	2	2.17	-0.17
Reply	9	9.19	-0.19
Forward	1	1.47	-0.47
Documents	1	1.86	-0.86
Get All New Mail	1	2.00	-1.0
New Tab	2	3.10	-1.1
Send	20	21.29	-1.29
Close Tab	1	3.75	-2.75
Create New Folder	1	4.28	-3.28
Copy	5	16.38	-11.38
Paste	3	16.21	-13.21

Other traditional sources of user interface knowledge

During the experiment, other resources for learning about the application were available, but the participants rarely took advantage of them.

All of the participants in the reference card group were given a printed reference card. On this one-page sheet were all of the functionality and associated keyboard shortcuts necessary to complete the experiment. However, the experiment proctor observed that none of the participants in that group leveraged this resource. For the participants in the reference card group, the answers were in plain view, yet no one considered the printed sheet as a viable option for solving the experiment tasks.

Between the first and second task set, all groups were given a paper tutorial that contained descriptions of useful operators that would help them improve their performance during the experiment. Upon being handed the tutorial, the participants were informed that reading the tutorial was not required. The average time spent with the tutorial was 8.5 min (SD = 7.74). Of the 21 participants that participated in the experiment, six ignored the tutorial.

After each section, the tutorial included exercise problems that allowed the participants to practice what they just learned. For the participants that read the tutorial, at least some practice exercises were completed (M = 10.33, SD = 8.11). There were some observations of the newly learned operator being used in the immediately preceding tasks; however, many of the participants reverted back to their previous methods.

Participants were also told that they were free to use any other existing resources, such as the application help or the Internet (e.g., Google Search), to solve a task. The transcripts revealed that no participants ever used the application help when they encountered an unknown task. Some participants did attempt to use a search engine to solve tasks with mixed results.

There were 10 observed instances of participants attempting to use a Google Search to discover how to perform a task. The success rate was low, only 30% – the participants were unable to construct an appropriate search term to describe the problem (Ekstrand *et al*. 2011).

The LFF inline trainer

The LFF inline trainer expanded the opportunities for a user to explore and discover new information about the applications they currently use on a daily basis.

Participants staged learning with caching

The LFF group was given access to a tool, LFF, that allowed them to quickly and easily cache potentially interesting and useful operations for future recall.

Table 7 presents transcript segments of participants caching operations for future recall.

Table 7. Caching operations during the interruption allows the user to stage future learning.

	Time	Actions performed by participant	
	(sec)	LFF Actions	Finder Actions
1.	0	Expand Configuration	
2.	2	Collapse Configuration	
3.	4	Expand Organize	
4.	8		Perform Arrange By Kind
5.	31	Cache Arrange By Kind	
6.	226		Perform Arrange By Kind
7.	227		Perform Arrange By Kind

(a) Discovering and caching Arrange By Kind operator

	Time	Actions performed by participant	
	(sec)	LFF Actions	Mail and Finder Actions
1.	0		Perform Send
2.	3	Expand Recently Used	
3.	9	Cache Send	
4.	12	Cache Get All New Mail	
5.	13	Cache Reply	
6.	19	Cache Paste as Quotation	
7.	25	Cache Forward	
8.	50	Expand Organize	
9.	53		Perform Documents

(b) Caching multiple recently used operators

In these transcript segments, operations are added to the participants' cache for future, quick recall.

The transcript in Table 7 demonstrates a participant caching a recently found operator. In this transcript segment, the participant is browsing the trainer concept hierarchy (lines 1–3), where the operator *Arrange By/Kind* is found. After performing the operation (line 4), the participant caches the operator (line 5) for future usage. A few minutes after the caching, the participant was able to quickly recall and successfully perform the operator again (lines 6 and 7).

Another example of a participant leveraging the interruption to stage future learning is presented in Table 7. In this transcript segment, the participant is seen starting a new task by referencing the recently used operations (line 2). From this list, the participant adds five operations to her favorites' cache (lines 3–7). In this instance, none of the operations cached were immediately required for the current task; however, the participant extended her own interruption to stage these operators for future recall.

Concept hierarchy led to discovery

Most software applications contain functionally related operators, such as the *Forward* and *Backward* operators in a browser, or Select All and Copy, where one operator relies upon the product of the other. Developing an understanding of the functional relationships between operators helps the user establish a more robust conceptual model of the application (van Merriënboer, Kirschner, and Kester 2003). Ideally, the organization of the menu system groups together functionally related operators. But in practice, this does not always occur. Thus, when the user is looking for a new operator, the operator may be located in an 'odd' place.

The inline trainer provides different access to the operator set of the application. Helping the user to better understand the relationships between operations and the conceptual model that underlies the operator set is the goal. Hence, a conceptual hierarchy that better represents the preferred understanding of the application is appropriate. The concept hierarchy took the menu applications and re-structured them into an enhanced representation of the system. So, in one case, using the menu, the user searches to find an operator to invoke, and in the other case, using the inline trainer, the user reasons more closely about the underlying conceptual model, leading to a deeper understanding of the overall system (Gurlitt, Schuster, and Nückles 2012). The issue is not whether one is more efficient than the other. The issue is: during a menu-based interruption in the main task, will users take the time to increase their knowledge of the application beyond just finding the relevant operator.

Leveraging the interruption window to teach the user about these related operators is an opportune time to ensure increased training effectiveness. Because the introduction of the new material occurs so close to the task objective, the participant can develop a link between goal and operator that is necessary in the development of his/her conceptual model.

Table 8 contains some transcript segments highlighting how various participants discovered new and related operators.

The transcript in Table 8 demonstrates a participant discovering, in the LFF trainer, the New Folder with Selection (line 5) operator while performing the task of creating a new folder. In this task, the participants were asked to place all of the PDF files in the Documents folder into a new folder and compress the new folder. To solve the task, the participant navigated to the Documents folder (line 2) and then introduced an interruption in the task to consult the Folders concept (line 3).

Table 8. The interactions in these tables demonstrate two instances of a participant using LFF to discover a new operator.

	Time	Actions Performed by participant	
	(sec)	LFF Actions	Finder Actions
1.	0	Expand Organize	
2.	3		Perform Documents
3.	35	Expand Recently Used	
4.	56	Expand Folder	
5.	60	Cache New Folder with Selection	
6.	63		Open Edit menu
7.	79		Open File menu
8.	93		Perform New Finder Window
9.	94		Perform Close Window
10.	101	Cache New Folder	
11.	105		Perform New Folder

(a) Interweaving training and task work

	Time	Actions Performed by participant	
	(sec)	LFF Actions	Finder Actions
1.	0		
2.	23	Expand Organize	
3.	43	View the Social Cache	
4.	57	Cache Move to Trash	
5.	61		Perform Move to Trash
6.	64		Perform Move to Trash
7.	66		Perform Move to Trash
8.	68		Perform Move to Trash
9.	70		Perform Move to Trash
10.	90	Cache Empty Trash	

(b) Using the social cache to discover operators

In (a), the participant discovers and caches two operators: New Folder with Selection and New Folder. In (b), the participant discovers and caches two operators for working with the trash. In each of these transcripts, the usage of the trainer is interweaved with the actual work task.

While in this interruption, the participant cached two related and potentially useful operators: New Folder with Selection (line 5) and New Folder (line 10). After caching the operators, the participant ended the interruption and used one of the newly cached operators (line 11) to complete the task. In this example, the participant discovered two possible solutions and then proceeded to use one of the newly cached operators with the keyboard shortcut multiple times during the experiment. This demonstrated not only a decrease in interruptions, but also an efficiency improvement.

The transcript in Table 8 also displays a participant discovering related functionality. While looking for an operator to move some files to the trash (line 2), the participant also discovered the Empty Trash operator (line 4).

The concept hierarchy presented the users with a discovery method that allowed them to find similar operators. When used in conjunction with the caching mechanism, the trainer allowed the participants to stage future learning and expand their conceptual model of the application.

Participants acquired a deeper knowledge with Quick Help

During the process of menu searching, research participants encountered unknown functionality. In this case, users had two choices: (1) ignore the operator and continue to search the menus or (2) perform the operation and observe the result. If the outcome was undesirable, the process could be undone. In one transcript, a participant is observed trying to figure out how to alter the text case. First the participant tries the Paste and Match Style operator to observe its effect. Unsatisfied with the outcome, the operation was undone, and the search resumed.

Although the second approach allows for the potential of learning to occur within the interruption, it also has a high error rate that could result in an even lengthier breakdown and task errors. To test the effectiveness of a learning aid designed to teach within the interruption window, a *Quick Help* feature was added to the training tool concept browser. In the training tool, each operator had a clickable icon that would provide quick and easy access to a descriptive help for the operator without further increasing the interruption. It accessed the help file text without launching the help system, a secondary application.

Table 9 presents two transcript segments of participants using the quick help feature to learn more about an operator. In the first transcript (Table 9), the participant is exploring the *Organizing* concept (line 14). During this exploration, the participant discovered a potentially useful operator, Mail Contents of This Page (line 15). Deciding this operator is something, the participant was interested in learning more about; the quick help allowed the participant to quickly overview the functionality without the potential erroneous effects of trial-and-error learning. In this instance, the operator was not an appropriate match for the current goal, which was to save the contents of a website to their desktop.

The second transcript (Table 9) is a snippet from a task where the participants were asked to clean up the files on their desktops. Here the participant was reviewing the *Folders* concept for the Finder application (line 1), looking for a solution to the current goal of creating a new folder. After creating the new folder (line 2), the participant goes back to the LFF training tool to view the quick help for the New Smart Folder operation (line 3). After reading the quick help, the participant switches back to Finder and executes the operation New Smart Folder (line 4). Here, the participant saw an operator New Smart Folder while browsing the concepts and was able to quickly view the descriptive. In this instance, after reading the help, the participant determined that this operator was useful and proceeded to use the operator.

These transcripts suggest that reducing the amount of in-interruption work required to learn more about an operation will lead to discovery and increased operator usage. If the duration of interruption is brief, the users will take the time required to learn about previously unknown operators with the intent of shortening their future task load. In Table 9 instead of using the typical method of drag-and-drop, the participant looked to the New Smart Folder operation to help reduce the work necessary in cleaning up a folder. By learning this new method, the participant expended some extra time to reduce future work and errors.

Discussion and concluding remarks

All users, irrespective of skill level, struggle with the problem of learning to use an application. As they work, they frequently rely on the menu to discover new knowledge

Table 9. The ability to quickly access and read the help for a function was another feature that helped the participants complete the experiment.

	Time	Actions Performed by participant	
	(sec)	LFF Actions	Safari Actions
1.	0		Open File
2.	6		Open Edit
3.	6		Open View
4.	7		Open History
5.	8		Open Bookmarks
6.	9		Open Window
7.	10		Open Help
8.	10		Open Window
9.	10		Open Bookmarks
10.	11		Open History
11.	12		Open View
12.	38	View All Actions	
13.	47	Expand Sharing	
14.	54	Expand Organizing	
15.	60	Display help for Mail Contents of This Page	
16.	64	View Social Cache	
17.	73	View Tips	
18.	83	View All Actions	
19.	92	Expand Accessibility	
20.	94	Expand Configuration	
21.	102	Expand General	

(a) Using LFF help to learn about Mail Contents of This Page

	Time	Actions Performed by participant	
	(sec)	LFF Actions	Finder Actions
1.	0	Expand Folders	
2.	21		Perform New Folder
3.	75	Display help for New Smart Folder	
4.	83		Perform New Smart Folder

(b) Using LFF help to learn about New Smart Folder

In (a), the participant switches from searching the menus to using LFF to find a way to complete the task. In line 15, the help for the Mail Contents of This Page was accessed. From this transcript, it can be seen that it only required 4 s of the participants'' time to open and review the help for this operator. In (b), another example of a participant interweaving LFF work with task work was provided. In this example, the participant reviews the help for the New Smart Folder operator. In this instance, after investing 8 s reviewing the help, the participant decided to use the operator.

or to re-find operators they previously used. The data collected in this study document the regular occurrence of these kinds of menu-related interruptions throughout the completion of routine tasks. Participants without access to the inline trainer could have chosen to use other resources – like an Internet search, the help system, or a reference card – but the data demonstrated that this did not regularly occur.

By incorporating an inline trainer into the workflow, some of the problems that emerge for users to better understand an application were offset, leading to better usage. One issue is whether or not an inline trainer will be used. In this study, evidence was presented demonstrating participants using and re-using an inline trainer to discover and learn new operational knowledge.

Also deterring the acceptance of an inline trainer was the engagement level of the user. Previous studies have demonstrated that engagement has a positive impact on learning (Carini, Kuh, and Klien 2006), and our results produced similar observations. The data confirmed that engagement level among the participants in each group varied and each of the three participant groups had some engaged users. Those users who had access to the trainer and were engaged were willing to do the extra bits of work necessary to improve their knowledge of the applications.

Interruptions are common and expected in the modern workplace (Noe, Clarke, and Klein 2014). However, not all interruptions are created equal, and one type of interruption, incurred from software menu access, creates a short disruption which under the proper circumstances may be sufficient enough to accommodate training. By closely examining the interactions of the user, opportunities for leveraging frequent interruptions emerge, leading to more effective usage of the tool in the long term, with little impact on short-term work. The inline trainer presented here is a learning technology geared toward leveraging menu-produced interruptions to present new application knowledge, allowing the users to expand their conceptual understanding of the tool.

This study has shown that leveraging the small, but frequently occurring interruptions expected in daily software interaction, users could discover new and more efficient methods for solving typical tasks. By providing the user with a means to quickly identify and address the cause of the interruption, a more robust mental model of the software could be developed, allowing the user to focus more on the domain and less on the tool.

References

Boothby, D., Dufour, A. & Tang, J. (2010) 'Technology adoption, training and productivity performance', *Research Policy*, vol. 39, no. 5, pp. 650–661.

Buckler, B. (1996) 'A learning process model to achieve continuous improvement and innovation', *The Learning Organization*, vol. 3, no. 3, pp. 31–39.

Carini, R., Kuh, G. & Klein, S. (2006) 'Student engagement and student learning: testing the linkages', *Research in Higher Education*, vol. 7, pp. 1–32.

Carroll, J. M. & Rosson, M. B. (1987) *Paradox of the Active User*, The MIT Press, Cambridge.

Digmayer, C. & Jakobs, E. M. (2014, October) 'Corporate lifelong learning 2.0: design of knowledge management systems with social media functions as learning tools', Professional Communication Conference (IPCC), 2014 IEEE International, pp. 1–9, IEEE, Pittsburgh.

Eagan, J. R. & Stasko, J. T. (2008, April) 'The buzz: supporting user tailorability in awareness applications', *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 1729–1738, ACM.

Ekstrand, M., et al., (2011, October) 'Searching for software learning resources using application context', *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, pp. 195–204, ACM, Santa Barbara.

Findlater, L. & McGrenere, J. (2004, April) 'A comparison of static, adaptive, and adaptable menus', *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 89–96, ACM, Vennia.

Findlater, L., et al., (2009, April) 'Ephemeral adaptation: The use of gradual onset to improve menu selection performance', *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems,* pp. 1655–1664, ACM, Boston.

Fu, W. T. & Gray, W. D. (2004) 'Resolving the paradox of the active user: stable suboptimal performance in interactive tasks', *Cognitive Science*, vol. 28, no. 6, pp. 901–935.

González, V. M. & Mark, G. (2004, April) 'Constant, constant, multi-tasking craziness: managing multiple working spheres', *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 113–120, ACM, Vennia.

Goodwin, D. (1994) 'Designing a quick reference guide: a teaching case', *Journal of Technical Writing and Communication*, vol. 24, no. 3, pp. 293–308.

Gravill, J. & Compeau, D. (2008) 'Self-regulated learning strategies and software training', *Information & Management*, vol. 45, no. 5, pp. 288–296.

Gray, W. (2017) 'Plateaus and asymptotes: spurious and real limits in human performance', *Current Directions in Psychological Science*, vol. 26, no. 1, pp. 59–67.

Gupta, S., Bostrom, R. P. & Huber, M. (2010) 'End-user training methods: what we know, need to know', *ACM SIGMIS Database: The DATABASE for Advances in Information Systems*, vol. 41, no. 4, pp. 9–39.

Gurlitt, J., Dummel, S. & Nückles, M. (2012) 'Differently structured advance orgainzers lead to different initial schemata and learning outcomes', *Instructional Science*, vol. 40, pp. 351–369.

Haramundanis, K. (2014) *The Art of Technical Documentation*, Digital Press, Charlottesville.

Hollands, J. G. & Merikle, P. M. (1987) 'Menu organization and user expertise in information search tasks', *Human Factors*, vol. 29, no. 5, pp. 577–586.

Iqbal, S. T. & Bailey, B. P. (2007, April) 'Understanding and developing models for detecting and differentiating breakpoints during interactive tasks', *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 697–706, ACM, San Jose.

Jin, J. & Dabbish, L. A. (2009, April) 'Self-interruption on the computer: a typology of discretionary task interleaving', *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 1799–1808, ACM.

Kim, J. W. & Ritter, F. E. (2015) 'Learning, forgetting, and relearning for keystroke-and-mouse-driven tasks: relearning is important', *Human-Computer Interaction*, vol. 30, no. 1, pp. 1–33.

Krisler, B. (2014) *Continuous Software Training with Three Inline Trainers*, Brandeis University, Waltham.

Mark, G., Gonzalez, V. M. & Harris, J. (2005, April) 'No task left behind?: examining the nature of fragmented work', *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 321–330, ACM, Portland.

Matejka, J., Grossman, T. & Fitzmaurice, G. (2011, May) 'Ambient help', *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 2751–2760, ACM, Vancouver.

McFarlane, D. C. & Latorella, K. A. (2002) 'The scope and importance of human interruption in human-computer interaction design', *Human-Computer Interaction*, vol. 17, no. 1, pp. 1–61.

Minassian, S. O., Muller, M. J. & Gruen, D. (2004) *Diverse Strategies for Interruption Management in Complex Office Activities*, IBM Research, Cambridge.

Noe, R. A., Clarke, A. D. M. & Klein, H. J. (2014) 'Learning in the twenty-first-century workplace', *Annual Review of Organizational Psychology and Organizational Behavior*, vol. 1, pp. 245–275.

Norman, D. (2013) *The Design of Everyday Things: Revised and Expanded Edition*, Basic Books (AZ), New York.

O'Conaill, B. & Frohlich, D. (1995, May) 'Timespace in the workplace: dealing with interruptions', *Conference Companion on Human Factors in Computing Systems*, pp. 262–263, ACM, Denver.

Olfman, L. & Mandviwalla, M. (1994) 'Conceptual versus procedural software training for graphical user interfaces: a longitudinal field experiment', *MIS Quarterly*, vol. 18, no. 4, pp. 405–426.

Reber, A. S. (1989) 'Implicit learning and tacit knowledge', *Journal of Experimental Psychology: General*, vol. 118, no. 3, p. 219.

Sears, A. & Shneiderman, B. (1994) 'Split menus: effectively using selection frequency to organize menus', *ACM Transactions on Computer-Human Interaction (TOCHI)*, vol. 1, no. 1, pp. 27–51.

van Merriënboer, J. J. G., Kirschner, P. A. & Kester, L. (2003) Taking the load off a learner's mind: instructional design for complex learning', *Educational Psychologist*, vol. 38, no. 1, pp. 5–13.

Zapf, D., *et al*., (1992) 'Errors in working with office computers: a first validation of a taxonomy for observed errors in a field setting', *International Journal of Human-Computer Interaction*, vol. 4, no. 4, pp. 311–339.