Task-Centric Interfaces for Feature-Rich Software

Benjamin Lafreniere[*], Andrea Bunt[†], Michael Terry[*]

[*]University of Waterloo
Waterloo, Ontario, Canada
{*bjlafren*, *mterry*}@cs.uwaterloo.ca

[†]University of Manitoba
Winnipeg, Manitoba, Canada
bunt@cs.umanitoba.ca

ABSTRACT

Feature-rich software can be difficult to learn and use, and current approaches to organizing functionality do little to help users with performing unfamiliar tasks. In this paper, we investigate the potential for alternative *task-centric* interface designs that organize functionality around specific tasks. To understand the potential of this approach, we developed and studied *Workflows,* a prototype task-centric interface design. Our findings suggest that task-centric interfaces scaffold and guide the user's exploration of a subset of application functionality, and thereby help them to avoid common difficulties and inefficiencies caused by self-directed exploration of the full interface. We also found evidence that task-centric interfaces enable a different kind of application learning, in which users associate tasks with relevant keywords as opposed to low-level commands and procedures. This has potential benefits for memorability, because the keywords themselves describe the task, and scalability, because a few keywords can map to an arbitrarily large procedure.

Author Keywords

Task-centric interfaces; feature-rich software; help; tutorials; learning; search-based interaction.

ACM Classification Keywords

H.5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

INTRODUCTION

Feature-rich desktop software relies on the WIMP (Windows Icon Menu Pointer) paradigm, whereby functionality is grouped into a hierarchical, taxonomy-like organization via menus, toolbars, palettes, and tabs. This design has the benefit of efficiently scaling to handle hundreds or thousands of features, but has drawbacks when a user is learning (or re-learning) how to perform a task. In particular, any given task may require functionality located *throughout* the interface, making it difficult for novice users to discover how to complete non-trivial tasks.

In this paper, we investigate an alternative interface design strategy in which a user's current *task* serves as the central organizing principle within the application. In this *task-centric* interface, users communicate their intended goal, and the interface responds by customizing itself for that specific task. We study this approach via *Workflows*, a prototype task-centric interface in which the user types in their goal (e.g., "black and white with color"), and receives step-by-step instructions embedded with the commands and tools necessary for accomplishing that goal (Figure 1).

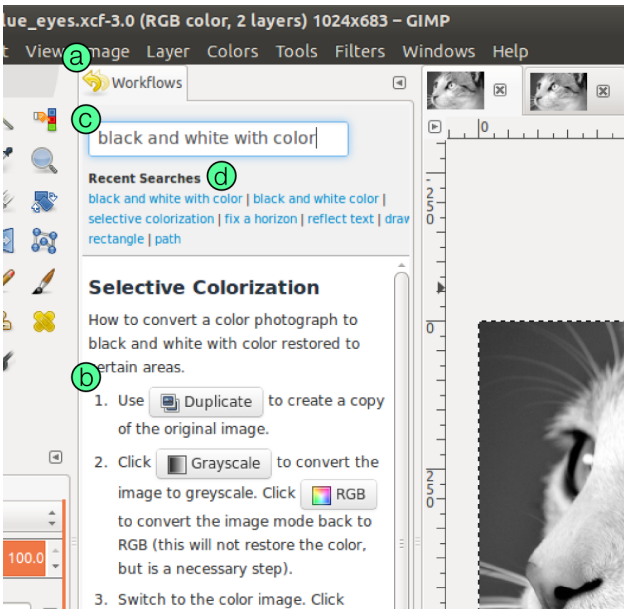

Figure 1. (a) The Workflows interface, (b) Instructions containing actionable commands, (c) Keyword search, (d) Links to return to recent searches.

We validated this design with a study conducted over two sessions. Our findings suggest that task-centric interfaces support a qualitatively different problem solving strategy that results in faster task completion times and reduced cognitive load. These gains appear to come from adjusting how users learn to perform an unfamiliar task. Whereas in current interfaces, users attempt to synthesize a solution from clues found during self-directed exploration of the interface, the task-centric interface guides users to a relevant subset of commands and procedures, which they can then focus on understanding and enacting.

Our findings also suggest that task-centric interfaces encourage a type of application learning in which the user associates tasks with relevant keywords, as opposed to low-level commands and procedures. This *keyword learning* is arguably more natural and economical than what must be learned in today's interfaces. More specifically, the keywords themselves are often the same, or overlap with how the user would naturally describe their goals, enhancing their memorability. Furthermore, since the task-centric interface maps these keywords to the required commands for that task, the keywords can act as a substitute

OzCHI '14, December 02 - 05 2014, Sydney, NSW, Australia

ACM 978-1-4503-0653-9/14/12 $15.00
http://dx.doi.org/10.1145/2686612.2686620

for learning the individual operations and their order. In this way, a short, memorable set of keywords can map to a potentially large and complex set of tools and procedures.

Collectively, these findings suggest specific benefits to mechanisms that re-organize a user interface for achieving particular high-level goals.

In the rest of the paper, we first situate our research within previous work, then present the Workflows prototype that we developed. Next, we describe the study we conducted to validate this design, and our findings. We conclude with a discussion of avenues for future work.

RELATED WORK

This research builds upon prior work in the domains of problem solving with feature-rich software; task-centric help and assistance mechanisms; and personalizable user interfaces.

Problem-Solving Strategies in Feature-Rich Software

Early HCI research showed that users of software are *task-focused*: continuous progress toward goals is paramount, and time spent on other concerns, including learning, is minimized to the greatest extent possible (Carroll and Rosson, 1987; Carroll, 1990; Mackay, 1991).

More recently, work has shown that users of feature-rich software favor trial-and-error strategies in which they explore the interface looking for relevant functionality (Andrade et al., 2009; Novick et al., 2009; Rieman, 1996). Two particular sources of difficulty associated with this approach are *hidden affordances*, where the interface does not sufficiently communicate how to proceed with the task, and *false affordances*, where perceived affordances in the interface mislead the user (Novick et al., 2009).

Our study findings indicate that a trial-and-error approach to learning can be particularly problematic when performing unfamiliar tasks, and show that task-centric interfaces support an effective alternative to this approach.

Task-Centric Help and Documentation

Commercial applications often include in-application help systems, some of which provide keyword search interfaces. However, the focus of these help systems is typically on instructing the user on how to use individual features of the application, rather than walking the user though specific high-level tasks. As a result, use of these systems is typically seen as a deviation from the task at hand, and users are hesitant to use them (Rettig, 1991).

Arguably the most prevalent sources of task-centric help today are web-based tutorials, which exist in abundance for popular feature-rich applications, and act as a kind of community-created help system (Lafreniere et al., 2013). However, the quality of instruction on the web varies widely, and this content is presented external to the application, which our study suggests acts as a disincentive for its use.

Task-centric interfaces can be seen as embracing the approaches of in-application help and use of web-based tutorials, to create an alternative interface for directly performing tasks in an application.

The customizations in our system share some similarities with the HTML tutorials created using Adobe's Tutorial Builder, which included a 'Show Me' button to execute individual tutorial steps in Photoshop (Adobe Labs Tutorial Builder, 2012). However, this past work did not address how users would access these tutorials, and did not examine the impact of this kind of tutorial on learning and performing unfamiliar tasks, which is the focus of this paper.

Finally, the HCI community has explored tutorial systems that present instructional material in the context of an application's interface (Fernquist et al., 2011; Kelleher and Pausch, 2005). These have been shown to allow users to learn content from a tutorial faster (Kelleher and Pausch, 2005), and to better communicate domain knowledge (Fernquist et al., 2011). However, the focus of these techniques is on guiding the user through an artificial task for the purpose of learning. In contrast, task-centric interfaces are a mechanism to assist the user with completing their own tasks.

In-Application Task Assistance

Existing work on task assistance mechanisms has focused primarily on automation. The most widespread example of a task assistance mechanism is the wizard—a linear dialog composed of multiple steps that prompts the user for input, and then performs the task for the user. Along similar lines, DocWizards (Bergman et al., 2005) and CoScripter (Leshed et al., 2008) allow users to create macros that may include steps that prompt the user to take an action. Numerous techniques have also been investigated to completely automate tasks through macros (Bergman et al., 2005; Berthouzoz et al., 2011), machine learning (Berthouzoz et al., 2011; Grabler et al., 2009; Yeh et al., 2009), automatically personalizing scripts for the current user (Leshed et al., 2008), or using the crowd (human workers) to complete a task (Bernstein et al., 2010).

While these approaches can expand the set of operations that can be completed with minimal input from the user, there will always be tasks that cannot be completely offloaded to an automated system. Most prominently, ill-defined problems require a human to judge the results of individual operations so they can adjust their actions accordingly (Schön, 1983). Our task-centric interface design provides a general-purpose approach to task assistance that can accommodate all tasks, even those with ill-defined components.

Personalizable Interfaces

Finally, our task-centric interface design can be considered an *interface personalization* mechanism, as it provides a way to change the interface to suit the user's needs. Existing personalization mechanisms include *layered interfaces* that offer the user access to a small number of predefined feature subsets (Shneiderman, 2002); *adaptable interfaces* that give the user tools to customize the application to suit their needs (Mackay, 1991; McGrenere et al., 2007); *adaptive interfaces* that model the user's interests, preferences, and usage to automatically tailor the interface to the user (Findlater et al., 2009); and *mixed-initiative* approaches that combine these various strategies (Bunt et al., 2007).

In contrast to existing work on interface personalization, our design tailors the interface to support users in performing specific *unfamiliar tasks*. Existing personalization

mechanisms are not well suited to this use because they require either the user to be familiar with the application's functionality (so the user can customize the interface), or the system to model past behavior (so the system can perform reasonable customizations).

WORKFLOWS – A PROTOTYPE TASK-CENTRIC UI

Workflows was implemented as a modification of the open source GNU Image Manipulation Program (GIMP), though its design could be readily applied in other applications as well. The primary, visible modification to the interface is an additional pane displayed alongside the traditional toolbox (Figure 1(a)). This pane allows the user to enter keywords describing their goal, and receive a customization of the interface catered to that specific goal. We describe these features in greater detail next.

Keyword Search

When a user has difficulty determining how to complete a task, a common practice is to use web-based search engines to find relevant help resources, such as web-based tutorials (Ekstrand et al., 2011; Kong et al., 2012; Lafreniere et al., 2013; Grabler et al., 2009). This strategy is compelling because the user can type their high-level goal using natural language, then (in the ideal) is directed to a web page that describes how to achieve the desired result, step-by-step.

Inspired by this existing practice, task-centric customizations are accessed using a search bar at the top of the Workflows panel (Figure 1(c)). The user enters keywords into the search box, and relevant customizations are shown in a list displayed immediately below (Figure 1(b)).

For searches that return one or more workflows, the search keywords are added to a list of "Recent Searches" (Figure 1(d)) that can be clicked to return to a search. This is intended to make it easy for the user to return to a previously used customization, or to quickly switch between customizations during a complex task.

There are several features of keyword search that make it particularly well-suited as a mechanism for accessing task-centric customizations. First, it naturally handles providing access to a large number of items. This is important, because the number of tasks that could be performed in a given application is potentially unbounded.

Second, keyword search doesn't require an overarching taxonomical organization scheme to be imposed on customizations (such as those used to organize commands in hierarchical menu systems). This is desirable, because it is unclear how one would go about classifying all the tasks that users may wish to perform in an application.

Finally, users performing an unfamiliar task may not know the domain-specific terminology associated with that task. Keyword search allows the user to express their intent in language that is natural to them, with the system producing the workflows that it believes to be most relevant. In this way, keyword search helps with crossing the gulf-of-execution between high-level goals and the low-level functionality and procedures to achieve them (Norman and Draper, 1986).

Task-Centric Customizations

In Workflows, customizations consist of a title, a short summary describing their intent, and a series of step-by-step text instructions (Figure 1(b)). References to commands, tools, or dialogs in steps are actionable—they can be clicked to execute the corresponding action, or display the referenced dialog. If a dialog is already visible, the system will highlight it by briefly flashing its border. With this design, the interface directly describes how to perform a task using each of the necessary tools.

Our design for task-centric customizations takes inspiration from web-based tutorials, though it deviates from them in several important ways. First, customizations are presented *in situ* within the application. This is significant because presenting instructions in a separate window forces the user to read a step, remember it, and then apply the instructions in the application—a process that is known to lead to errors (Knabe, 1995). Second, references to commands are actionable, so users do not need to locate referenced functionality. Finally, we have standardized steps to present short, succinct instructions. In contrast, web-based resources vary greatly in their style of presentation and the amount of detail, expository text, and visual aids that they provide.

Authoring Customizations

In our prototype system, customizations are authored by creating an HTML file and adding it to a shared file repository, which is then synced to individual installations of Workflows. This approach served our needs for prototyping and testing, but more sophisticated authoring mechanisms would be required for a deployed system.

For the purposes of our user study, we manually created a set of customizations based on popular search queries for GIMP, using a method that we explain in the next section.

USER STUDY

In this section, we report on a user study we conducted to understand how a task-centric interface design impacts users' problem-solving strategies for performing unfamiliar tasks. Our study compares use of Workflows with existing practices, namely the use of the unmodified GIMP application combined with web search.

Study Design

Our study employed a within-subjects design with two conditions. In the control condition, participants were provided with GIMP 2.8 and a web browser loaded to the Google search page. Participants were instructed to use whatever strategies they would typically use (including web search if desired) to determine how to complete the requested task. In the experimental condition, participants were provided with Workflows and asked to use it as their primary method for completing the task. They were *not* permitted to use the web in the experimental condition. We imposed these restrictions because our goal was to learn how the task-centric interface would influence participants' problem-solving strategies.

To simulate both the scenario of performing an unfamiliar task for the first time, and returning occasionally to re-perform a task, the study consisted of two sessions for each participant. In the first session, the participant performed

four tasks that were new to them (two per condition). In a second session, held at least two weeks later (median 21 days, min 14, max 29)[1], the participant returned and performed the same tasks again. The mapping of the two sets of tasks to conditions; the order in which participants experienced the two conditions; and the order of tasks within each set were fully counterbalanced across participants. Each participant experienced the same task and condition ordering, and task-to-condition mapping, in both of their sessions.

Tasks and Procedure

Each session lasted approximately one hour. The first session began with a short demographic questionnaire. The experimenter then gave participants a brief overview of the major parts of the GIMP interface (the document, toolbox, tool settings, color picker, layers, and undo command). In the first session, an overview of the features of Workflows was provided immediately before the participant started this condition. In both sessions, before each condition, participants completed a short practice task to (re)familiarize them with the resources available in that condition.

In each session, participants completed the four tasks shown in Figure 2. Each task requires use of some direct manipulation operations, and so would be difficult to automate (making them representative of the type of tasks for which we imagine task-centric interfaces would be particularly well suited). These particular tasks were also selected to have minimal overlap in commands and tools, to mitigate learning effects between tasks.

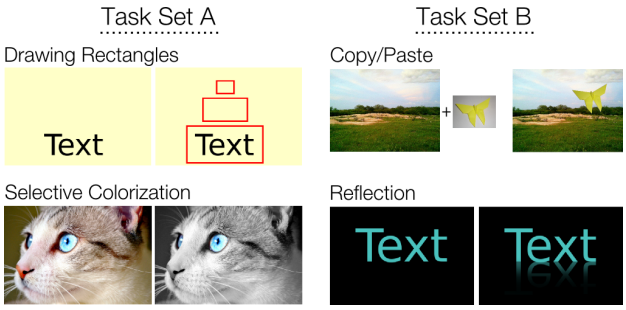

Figure 2. Before/after images for the four study tasks.

Each task was presented to the user as a before/after image displayed on a second monitor, without any text describing the task to be performed. We opted for this visual presentation of the task to avoid biasing the participant with terminology that they might use when searching for tools in the interface, or when performing keyword searches on the web or in the Workflows panel. Following each task, the participant filled out a NASA Task-Load Index (TLX) questionnaire to measure cognitive load (Hart and Staveland, 1988). At the end of the session, the participant filled out a post-session questionnaire.

Participants performed tasks on a computer that we provided, within a virtual machine that could preserve state. At the end of the first session, we preserved a snapshot of each participant's virtual machine to ensure that their search histories for both Workflows and the web browser were available in their second session.

The procedure for the second session was similar to the first, except that participants were not given a tour of the interface before each condition. Additionally, a semi-structured interview was conducted at the end of the second session to elicit participants' impressions of both conditions across the two study sessions.

An experimenter was present throughout the study to make observations, and to judge when the participant had completed a task and was ready to move on to the next task. If the participant appeared stuck, or declared that they had completed the task when their current document differed significantly from the goal image, the experimenter would ask "Is there anything else you could try?" or "Is there any way you could make it look more like [the goal image]?" The experimenter capped each task at 12 minutes.

Study Tasks and Available Customizations

To minimize bias that could arise due to our choice of tasks or our means of authoring customizations for the experimental condition, we developed the following procedure to choose the study tasks and create customizations.

First, we identified common web-search queries for GIMP using the CUTS technique (Fourney et al., 2011a), and from these, selected the set of tasks for the study (choosing a set of tasks with minimal tool/command overlap, and a reasonable length for a lab study). Next, to create the customizations, we executed each task's associated search query and selected the highest ranked web page that contained step-by-step instructions for completing that task. We then used a set of predefined templates and heuristics to create a customization with the same procedure as was documented on the selected page (see Appendix for details). This process was intended to ensure a reasonable degree of equivalence in the procedures available in the two study conditions, despite the differences in form between web-based tutorials and the customizations served in Workflows.

In addition to the four customizations created for the study tasks, we included one customization for a practice task and eight additional customizations for other tasks not tested in the study. These additional tasks simulated expected real-world conditions where the user would need to locate a relevant customization amongst many.

Participants

We recruited 16 participants from a university campus (10 male, 6 female) with ages ranging from 21–31 (median 24). Participants were recruited via postings on an email mailing list targeted toward graduate students. Participants were screened to ensure that they (1) had minimal experience with image editing software, and (2) were native English speakers (to control for variability in strategies that may depend on ability to formulate search queries). All but one participant took part in both study sessions. In appreciation for their time, participants were given a $10 gift card for an online retailer for the first session, and a $15 gift card for the second session.

[1] There were no significant differences in the number of days between sessions for any of our counterbalancing factors ($p > 0.40$ for all factors).

RESULTS

We first discuss the impact of Workflows on performance and self-assessments of cognitive load, then provide a detailed qualitative and observational analysis of participants' problem-solving strategies. We close with a discussion of participants' reactions to Workflows.

Task Times and Cognitive Load

In the first session, users completed 41 of 64 study tasks within the time allotted, with more tasks completed in the Workflows condition than in the control (23 vs. 18). In the second session, users completed 48 of 60 study tasks within the time allotted, with more tasks completed in the Workflows condition than in the control (26 vs. 22). Neither of these differences in number of tasks completed was found to be statistically significant. In our analysis of task times presented below, we include all tasks.

To compare task completion times, for each participant we averaged the time they spent on the two tasks for each condition/session (Figure 3). For both the first and second sessions, we observed significantly lower average task completion times in the Workflows condition.

For Session 1, the median avg. task time using Workflows was 432 seconds (IQR 187), versus 538 seconds (IQR 210) for the control condition. A Wilcoxon Signed-Rank test showed this difference to be significant ($W=11$, $Z=-2.783$, $p < .01$, $r=0.49$). This difference persisted in the second session (where users were asked to repeat the same tasks), in which the median avg. task time for Workflows was 211 seconds (IQR 205) versus 420 seconds (IQR 322) for the control condition ($W=13$, $Z=-2.669$, $p < .01$, $r=0.49$).

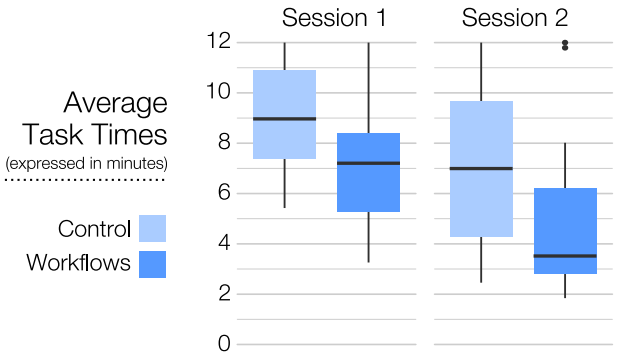

Figure 3. Average per-participant task times

Our findings for perceived cognitive load mirrored the findings for task completion times. The results of the NASA-TLX, broken down by component subscales and session are shown in Figure 4. For Session 1, a paired t-test found a significant difference in the average cognitive load for condition ($t(15)=-2.7462$, $p < .05$, Cohen's $d=0.69$), with the experimental condition showing lower cognitive load than the control condition. This result was mirrored in the second session, ($t(14)=-2.1998$, $p < .05$, Cohen's $d=0.57$). In terms of the individual subscales, Workflows performed better than or was equivalent to the control condition in all cases.

The results for task time and cognitive load indicate that a task-centric interface design can significantly improve performance and lower cognitive load as compared to current strategies for performing unfamiliar tasks.

Our qualitative analysis provides a more detailed account of where participants saved time when using Workflows, and the sources of difficulty and frustration in the control condition.

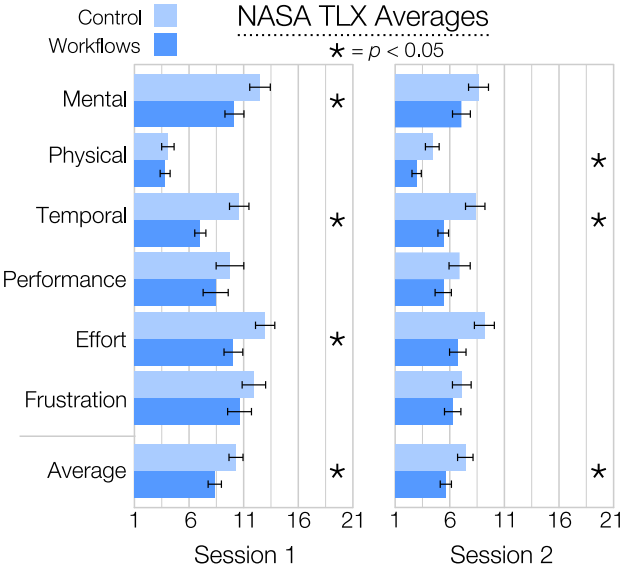

Figure 4. Average cognitive load for the six axes of the NASA-TLX by condition and session.

Problem-Solving Strategies

We chose to analyze two of the study tasks, Selective Colorization and Reflection, in greater detail to gain insights into the problem-solving strategies employed by participants. These two tasks were selected because they are the more complicated tasks in each of the task sets, and thus should reveal more about participants' problem-solving strategies.

The first author of this paper reviewed the video recordings of the study sessions, making observations and assigning qualitative codes to segments of time spent on various activities. Qualitative codes were developed using an open coding approach that was initially seeded with a coding scheme developed on a small set of study videos. During coding, the initial coding scheme evolved in minor ways. When this occurred, the codes for earlier videos were revised based on the updated scheme.

A summary of the resulting codes by study condition is shown in Table 1. Note that some of the codes are specific to one condition or the other (e.g., users in the Workflows condition sometimes spent time performing searches that returned no results, but this did not occur with web search in the control condition). In sections that follow, we reference these results to ground our discussion.

High-Level Problem-Solving Strategies

Participants exhibited different overall strategies in the two conditions. In the Workflows condition, participants almost universally started by using search to find a relevant workflow, and then attempted to apply the workflow instructions to complete the task. We term this the *guided-and-constrained* strategy—"guided" because participants used the workflow instructions as a framework for completing the tasks, and "constrained" because we found that participants spent less time using the interface outside of the Workflows panel.

W %	C %	Observation
41	--	Following instructions from a workflow.
17	27	Experimenting with tools or settings.
13	22	Looking for relevant functionality in menus, toolboxes, settings dialogs, etc.
9	21	Using the full interface to perform an action (excluding experimentation).
--	18	Viewing a page in the web browser (control condition only)
6	--	Keyword searches in the Workflows panel that returned no results.
5	6	Keyword searches in the Workflows panel / web that return results.
4	1	Other (not covered by other categories, e.g. adjusting size of a window, switching between open images, reading text in the interface).
2	1	Interacting with experimenter (e.g. experimenter encouraging participant to continue with task.)
2	3	Time spent on error recovery using Undo (excluding experimentation.)
2	--	Browsing workflows using the Recent Searches bar.
<1	<1	No obvious action being performed.
<1	<1	Using GIMP's built-in help system.

Table 1. Summary of activities for the two study conditions (W=Workflows, C=Control). Numeric values indicate the percentage of time coded with that activity across all users.

In the control condition, participants typically attempted to complete the task with minimal assistance from external help or documentation, even though a web browser with Google Search was open and available to them. Instead, they would often search through the interface trying to find relevant functionality, or experiment with tool and commands. We term this the *interface-exploration* strategy.

These two strategies are reflected in our qualitative coding. In the Workflows condition, participants spent the largest share of their time following instructions in the Workflows panel (41% of time)[1]. If we also include time spent issuing keyword searches (both successful and not), participants spend 54% of their time interacting with the system primarily through the Workflows panel. In contrast, the two most common activities in the control condition were experimenting with tools and settings (27%), and looking through menus, toolboxes, and dialogs (22%). These results indicate that the main activities in the Workflows condition were related to seeking and receiving guidance, whereas the main activities in the control condition were related to exploring the interface. Furthermore, in the Workflows condition participants spent significantly less time looking through the interface for functionality (Wilcoxon Signed-Rank test, $W=70$, $Z=-3.029$, $p < .01$, $r=0.38$); experimenting with tools and settings ($W=84.5$, $Z=-2.876$, $p < .01$, $r=0.37$); and using the full interface of the application ($W=31$, $Z=-4.253$, $p < .01$, $r=0.54$).

The difference in problem-solving strategies between the two conditions demonstrates that Workflows provides an alternative to self-guided exploration of the application's interface, and that task-centric customizations provide guidance and scaffolding for the user's efforts to perform unfamiliar tasks.

Sources of Difficulty – Control Condition

We observed that the interface-exploration strategy was associated with several common sources of difficulty in the control condition, which we discuss below.

Cognitive Overload: Participants following an interface-exploration strategy would appear to rapidly switch between: trying to find relevant functionality; experimenting with functionality that they found to learn how tools and commands function; and devising, evaluating, and reevaluating strategies for progressing toward the task goal. It's difficult to imagine how juggling all of these concerns *wouldn't* impose a high amount of cognitive load on the user. This is consistent with the finding of higher self-reported cognitive load discussed in the previous section.

Red-Herring Commands: Participants would sometimes find commands that *appeared* to be relevant to the current task, but were not appropriate. For example, a common solution for the Reflection task is to make a copy of the text layer, and then flip it vertically with either the Flip tool or the *Layer>Transform>Flip Vertically* command. However, many participants' explorations led them to instead find the *Image>Transform>Flip Vertically* command. This command flips the entire image, including all layers, and was a source of great frustration for participants in this task, who desired to flip only the currently selected layer. Red-herring commands are an instance of the concept of *false affordances* (Gaver, 1991).

Setting State Problems: After experimenting with tools and settings, participants would seldom reset settings to their defaults, which would leave the system in a non-typical state. This was a source of difficulty and frustration when the side effects of previous explorations would cause difficulty and unexpected results later on.

The task-centric interface design in Workflows addresses these three observed sources of difficulty by indicating relevant functionality and providing guidance in the interface, thus leading to less unguided interface exploration.

Sources of Difficulty – Workflows Condition

In addition to the issues identified in the control condition, we observed a number of common difficulties in the Workflows condition, which indicate areas where the task-centric interface design could be improved.

Formulating Search Queries: In the Workflows condition, a number of participants had difficulty formulating keyword searches to find a customization for their task. Because the pool of customizations available in the study was small, this often resulted in searches that would return

[1] This includes time spent reading and interacting with the customization, but also includes interactions outside of the Workflows panel if the participant is following an instruction from a customization (e.g. "Select the greyscale layer in the Layers dialog." requires the user to click a layer in the indicated dialog). This was typically easy to determine because participants would follow instructions sequentially.

no results, which was a source of frustration. Conversely, in the control condition we observed participants making extensive use of Google's query completion feature; the participant would type in search terms and carefully consider the list of suggestions that was displayed, and then either select a suggested query, or refine their query based on terminology from the list of suggestions.

These findings suggest the value of adopting common search engine features to help users to bridge differences in vocabulary and find relevant customizations faster. For example, one possibility is to adopt an autosuggest mechanism, further enhanced for the particular domain of the application. For example, in the visual domain of image editing, autosuggest could display preview images alongside terminology; this could help users to recognize relevant queries while also helping them to build an understanding of domain terminology as they used the system.

Skipping Text Instructions: A source of errors in the Workflows condition came from participants' skipping over text instructions, such as "switch to the greyscale image", in customizations. In extreme cases, participants treated the customization like it was a macro, where they could simply click through each command to complete the task, as can be seen in the following quote:

> *My first instinct is to go through really quick, you know what I mean, and I'm not necessarily reading all the text. So I click on the tool and I just think "I don't have to read what's in between" and I click the next tool. And that's me just being kind of lazy, or whatever, but that was kind of the way that I wanted to do it. But some of those tools, to make them work properly, or fully understand what I had to do, I had to read some of the text a little bit more, and that wasn't necessarily how I was going, just right out of the gate. [P10]*

We suspect that this is the result of the higher visual weight and interactivity of the actionable commands, which leads users to focus on them more than the included text-based instructions.

A potential design to address this issue would be to include *human action buttons* for important text instructions in customizations. When clicked, these buttons could pop up text or animated instructions describing what the user should do next. This would ensure that important manual actions have the same weight as actionable commands.

Understanding the Dynamics of Tools and Steps: Finally, we observed that participants sometimes had difficulty with understanding the dynamics of how to use a tool, or how to perform a step in a customization. We found this was especially problematic for steps that involved use of direct manipulation tools.

This suggests that our text-only instructions are too limited to fully communicate the dynamics of some tools and operations, which is consistent with findings from previous work (Chi et al., 2012; Grabler et al., 2009). The inclusion of images or animated demonstrations accessible on-demand (e.g., ToolClips (Grossman and Fitzmaurice, 2010)), could help address this problem, while still keeping instructions succinct.

In summary, the difference in common sources of difficulty between the two conditions serves to further reinforce our observation that Workflows enables an alternative problem-solving strategy. Moreover, the common sources of difficulty for the Workflows condition suggest potential improvements to the design of the form of customizations and mechanisms for accessing them.

Learning in Session 1

Our two-session study design allowed us to also gain insights into how task-centric interfaces can support learning and re-performing of tasks.

Overall, we found evidence to suggest that in both conditions participants retained knowledge of relevant functionality in the second session. For the tasks we qualitatively coded, participants spent significantly less time in the second session looking through the interface for functionality (Wilcoxon Signed-rank test, $W=289.5$, $Z=2.896$, $p < .01$, $r=0.37$) and experimenting with tools and settings ($W=271$, $Z=2.920$, $p < .01$, $r=0.38$).

Among participants who used search in both sessions (14 in the Workflows condition, 7 in the control), significantly less time was spent on search in the second session as well ($W=159$, $Z=2.016$, $p < .05$, $r = 0.31$). This suggests that participants may have used more refined search strategies in the second session, or simply remembered terms from the first session. We found evidence for this in the post-study interviews, where five of the 15 participants mentioned that remembering terminology from the first session helped them to return to resources in the second session. Some example quotes include:

> *I knew, oh last time I used "greyscale", and that worked, so I can just put "greyscale" and then boom. And I knew that worked. [P4]*

> *I think, for me what made the difference, is I knew… having used it in the previous session I had a better idea of what words to use and what to look for, I guess, the terminology. [P5]*

We found evidence of this strategy for use of the web in the control condition as well, as in the following quote:

> *I use the Internet to do everything in my life that way. So it's almost like, you think, "Oh I don't really need to remember it because it's on the Internet." [P10]*

This suggests that, in addition to learning relevant functionality, participants also learn how to return to task-centric help resources. We term this *keyword learning*.

Though we observed this phenomenon in both conditions, it is particularly significant for task-centric interface designs, in which the dominant problem-solving strategy is based around search. In contrast, users in the control condition seemed hesitant to go to the web for help. For example, P10 in the Reflect task started the second session by saying "oh" quietly to himself as if he was trying to remember something. As he did so, twice he moved his mouse pointer to the icon for the open web-browser in the task bar, as if he was going to click, but then returned the mouse to the interface instead. This internal debate lasted for 10 seconds, after which he adopted an interface-explo-

ration strategy, searching for commands. Only after a minute more of exploring did he switch to the web browser window and re-find a page from the first session. While this is only one specific example, the lower number of participants who searched in both sessions in the control condition suggests a hesitance to go to the web for help, and a corresponding overestimation by individuals of how much progress they can make unaided.

Participant Reactions

In post-study interviews, we asked participants to discuss the relative advantages and disadvantages of Workflows as compared to the control condition.

The most appreciated feature of the Workflows system was the actionable buttons (mentioned by 11 of the 15 participants), as the following quotes demonstrate:

> *Sometimes when I was working on my own, I had to scroll over everything to find exactly what they meant. Whereas this says, okay 'Crop' and then it gives you the actual tool, so you're not having to look. [P7]*

> *It saves you having to look for [the commands] after you have the instructions, you know what I mean. [P10]*

> *(...) it gave step-by-step instructions on how to complete a task, as well as even providing the buttons right there, if I couldn't find where the buttons actually were in the program. Compared to the web, I mean you type in something and it doesn't tell you exactly where the tool is, sometimes it just explains what the process is without actually, I guess, guiding you through it. [P15]*

As suggested by the quotes above, a key reason cited for appreciating the actionable buttons was that they saved the effort of locating commands in the interface (mentioned by 8 of those 11 participants). This suggests that a source of gains for the Workflows condition is saving the time and effort required to locate required commands in the application's interface.

Participants also expressed appreciation for having relevant information and functionality presented in the application itself:

> *One [advantage] is that it's all in the same environment, so you don't have to open some other program to access this workflow thing. Like, you don't have to alt-tab. Like if you only have one monitor, then you're constantly switching back and forth, it's a hassle. [P11]*

The succinct presentation of instructions in Workflows was cited as an advantage as well:

> *[The Workflows panel] was pretty verbose, but not to the point where, say, it was one of these [referring to a web tutorial on screen] where all this could be easily said within like two instructions within the workflows panel. This is what I'm looking for, but there's a whole bunch of [extra] stuff on the webpage. [P14]*

The main disadvantage that participants cited for Workflows was the quality of the search, as in this quote:

> *I think I mentioned it before, but sometimes no results would pop up at all. Like when I was working on the cat one, I thought a very basic way of finding it would be writing "one color", for example, and nothing showed up! [P15]*

This is consistent with our observation that participants had difficulty formulating search queries, and further suggests that refinements to the search facility, such as auto-suggest mechanisms, could be beneficial.

DISCUSSION AND FUTURE WORK

Our study results suggest that task-centric interfaces support a viable and efficient alternative to the interface-exploration strategy typically adopted by users of feature-rich software. In particular, by indicating relevant commands, providing guidance on how commands can be used together for a task, and saving users the effort to locate commands in the interface, the system enabled participants to complete unfamiliar tasks faster and with reduced cognitive load as compared to current practices. We also found evidence that search-based interfaces enable users to learn keywords to return to task-centric customizations, as an alternative to learning the lower-level commands and procedures themselves.

In this section we discuss the implications of our results in greater detail.

An Alternative Learning Model

One way to view our study findings is that task-centric interfaces realize gains by adjusting what needs to be learned in order to carry out an unfamiliar task. Current interfaces require the user to learn the locations of individual commands and general knowledge about how individual commands work, and then to synthesize this general knowledge into a plan for how to carry out their task. This learning model makes sense when the goal is to support regular use of the software by expert users, who could be expected to learn this general knowledge during an initial training period. However, this approach imposes a great deal of cognitive load and potential frustration on the user who may simply desire to quickly reach a particular end goal.

Task-centric interfaces invert this learning model. Instead of learning general knowledge about individual commands, the interface supports learning task-specific knowledge (e.g., the keyword searches representing a task; how individual commands operate in the context of a task; and the conceptual relationships between commands for particular tasks). In the short term, this knowledge allows the user to quickly reach specific end goals. In the long term, this knowledge could be synthesized into more general knowledge about individual commands and the system as a whole.

Keyword Learning

Our observation that users learn keyword searches to return to task-centric help resources also suggests an alternative model for learning in feature-rich software. Previous work has shown that users re-find information using keyword search on the web (e.g., (Aula et al., 2005)), but this finding is particularly significant for feature-rich software for two reasons. First, by learning a small set of keywords, a user can return to an arbitrarily large set and sequence of commands to help them complete a task. In this way, keyword learning has the potential to scale more gracefully than learning the precise details of how to perform each

task. For performing unfamiliar or occasional tasks in particular, this strategy appears to have clear benefits over learning commands and procedures.

Second, keyword learning has the advantage that the keywords themselves act as a kind of description of the task to be performed. In contrast, the commands necessary to complete a task may have generic names, or may use domain-specific terminology that is foreign to the user, making them more difficult to remember and recall. Because keywords are intrinsically descriptive of the task to be performed, they are likely to be more memorable.

Supporting À La Carte Usage

The alternative learning model supported by task-centric interfaces is particularly well-suited to the growing number of sophisticated software applications that are available for free to anyone who wants to use them (e.g., open source applications such as GIMP, Inkscape, Blender, and Audacity, or free-to-use applications such as TinkerCAD, 123D Design, Pixlr, and Google's Drive suite). The ease of accessing these applications naturally supports an *à la carte* usage scenario, in which a sophisticated application is used to perform only one or a small number of tasks, using only a small percentage of its full functionality (Lafreniere et al., 2010). In this scenario, supporting the user in quickly performing unfamiliar tasks is arguably more important than supporting more general learning of the application, which may not provide enough benefit to justify the required time and effort.

Extending the work presented in this paper, it would be interesting to look at how the task-centric interface approach could be augmented to better support à la carte usage. One avenue for future work would be to examine how support could be provided *before an application has been installed*, to support users who have a goal in mind, but do not yet know whether an application exists that is appropriate for their task. One could imagine a kind of task search engine on the web that would help users to quickly locate and install an application, and then provide a task-centric interface within the application to guide the user through performing the task.

It would also be interesting to look at how to support a transition from à la carte usage of an application to more general use. As an example, in Workflows this might be achieved by fading away or hiding the step-by-step instructions over time, so a customization for a commonly used task gradually becomes a succinct toolbar of commands. Supporting a transition to more efficient interaction techniques has previously been explored for command invocation (Kurtenbach et al., 1994), but we are unaware of any work that has examined this problem for higher-level tasks.

Limitations and Areas for Future Work

In this work, we have focused on the scenario where the user wishes to perform a task, and a single appropriate customization is available in the task-centric interface. Having established benefits in this initial scenario, there is an opportunity to examine how to best support situations where the user must draw upon multiple customizations to complete a task, or when the available customizations cannot provide ideal support for a task.

We have also not examined how a comprehensive collection of customizations could be created for a task-centric interface. For applications with large, established user bases, rich collections of tutorials already exist on the web, which could potentially be converted into task-centric interfaces. It may be possible to automate some aspects of this conversion; recent research has demonstrated techniques for automatically identifying command-task relationships from web search query logs (Fourney et al., 2011b), which could act as a starting point for more detailed task modeling. Crowdsourcing conversion of tutorials is another possibility that could be explored. Finally, past work has proposed the idea that a community of users of an application could collectively create, document, and refine a set of interface customizations over time (Lafreniere et al., 2011).

SUMMARY AND CONCLUSIONS

This paper has presented and evaluated an alternative task-centric interface design for feature-rich software, which provides keyword search access to task-specific interface customizations. A study with two sessions spanning at least two weeks indicates that this design can enable qualitatively different problem-solving strategies for performing new and infrequently performed tasks, with significant gains in performance and reductions in cognitive load, as well as insights into how a task-centric interface change learning in feature-rich software.

APPENDIX

The templates in Table 2 were used to create customizations based on web tutorials, while preserving the overall procedure of the tutorial. In addition, vague settings in the tutorial were made explicit (e.g. "Choose a large brush" was converted to "Choose a 150px brush"), and an "(Optional)" flag was added to steps or procedures that weren't strictly necessary.

Template	Example Text
Switch to <Short Description> image.	Switch to the greyscale image.
Select the <Name> tool.	Select the **Crop** tool.
Click <Command or Button> [to <Desired Effect>].	Click **Duplicate** to create a copy of the original image.
[Hold the <Ctrl-, Shift-> key and] [<left, right>] [Click][and drag] to <Desired Effect>.	Hold the **Shift** key and left click where you want the line to end.
Use the following settings: <Setting: Value pairs>	Use the following settings: · Width: *200 percent* · Height: *200 percent*
Set the <FG,BG> color to <Color> [and the <FG,BG> color to <Color>].	Set the foreground color to *Black*.
Use one or more selection tools to <Effect>. <All selection tools>.	Use one or more selection tools to create a selection in the desired shape. **Rect Select**, **Ellipse Select**, **Free Select**, ...
In the <Name> dialog <Do Action>.	In the **Tool Options** dialog, use the following settings: …
In the Layers dialog, select the <Short Description> layer.	In the **Layers** dialog, select the greyscale layer.
Use the <Name> tool to <Desired Effect> [by <Method>].	Use the **Text** tool to create some text.

Table 2. Templates used to create customizations from web tutorials. Bold text indicates actionable buttons.

ACKNOWLEDGMENTS

This work was supported by the Graphics, Animation, and New Media research network (GRAND/NCE) and by the Natural Sciences and Engineering Research Council of Canada (NSERC).

REFERENCES

Andrade, O.D., Bean, N., and Novick, D.G. The macro-structure of use of help. Proc. SIGDOC '09, ACM, (2009), 143–150.

Aula, A., Jhaveri, N., and Käki, M. Information Search and Re-access Strategies of Experienced Web Users. Proc. WWW '05, ACM, (2005), 583–592.

Bergman, L., Castelli, V., Lau, T., and Oblinger, D. DocWizards: a system for authoring follow-me documentation wizards. Proc. UIST '05, ACM, (2005), 191–200.

Bernstein, M.S., Little, G., Miller, R.C., et al. Soylent: a word processor with a crowd inside. Proc. UIST '10, ACM, (2010), 313–322.

Berthouzoz, F., Li, W., Dontcheva, M., and Agrawala, M. A Framework for content-adaptive photo manipulation macros: Application to face, landscape, and global manipulations. ACM Trans. Graph. 30, 5 (2011), 120:1–120:14.

Bunt, A., Conati, C., and McGrenere, J. Supporting interface customization using a mixed-initiative approach. Proc. IUI '07, ACM, (2007), 92–101.

Carroll, J.M. *The Nurnberg funnel: designing minimalist instruction for practical computer skill.* MIT Press, 1990.

Carroll, J.M. and Rosson, M.B. Paradox of the active user. In *Interfacing Thought: Cognitive Aspects of Human-Computer Interaction*. MIT Press, 1987, 80–111.

Chi, P.-Y., Ahn, S., Ren, A., Dontcheva, M., Li, W., and Hartmann, B. MixT: Automatic generation of step-by-step mixed media tutorials. Proc. UIST '12, ACM, (2012), 93–102.

Ekstrand, M., Li, W., Grossman, T., Matejka, J., and Fitzmaurice, G. Searching for software learning resources using application context. Proc. UIST '11, ACM, (2011), 195–204.

Fernquist, J., Grossman, T., and Fitzmaurice, G. Sketch-sketch revolution: an engaging tutorial system for guided sketching and application learning. Proc. UIST '11, ACM, (2011), 373–382.

Findlater, L., Moffatt, K., McGrenere, J., and Dawson, J. Ephemeral adaptation: the use of gradual onset to improve menu selection performance. Proc. CHI '09, ACM, (2009), 1655–1664.

Fourney, A., Mann, R., and Terry, M. Characterizing the usability of interactive applications through query log analysis. Proc. CHI '11, ACM, (2011a), 1817–1826.

Fourney, A., Mann, R., and Terry, M. Query-feature graphs: bridging user vocabulary and system functionality. Proc. UIST '11, ACM, (2011b), 207–216.

Gaver, W.W. Technology Affordances. Proc. CHI '91, ACM, (1991), 79–84.

Grabler, F., Agrawala, M., Li, W., Dontcheva, M., and Igarashi, T. Generating photo manipulation tutorials by demonstration. ACM Trans. Graph. 28, 3 (2009), 66:1–66:9.

Grossman, T. and Fitzmaurice, G. ToolClips: An investigation of contextual video assistance for functionality understanding. Proc. CHI '10, ACM, (2010), 1515–1524.

Hart, S. and Stavenland, L. Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. In P. Hancock and N. Meshkati, eds., *Human Mental Workload*. Elsevier, 1988, 139–183.

Kelleher, C. and Pausch, R. Stencils-based tutorials: Design and evaluation. Proc. CHI '05, ACM, (2005), 541–550.

Knabe, K. Apple guide: a case study in user-aided design of online help. Proc. CHI '95, ACM, (1995), 286–287.

Kong, N., Grossman, T., Hartmann, B., Agrawala, M., and Fitzmaurice, G. Delta: a tool for representing and comparing workflows. Proc. CHI '12, ACM, (2012), 1027–1036.

Kurtenbach, G., Moran, T.P., and Buxton, W. Contextual Animation of Gestural Commands. Computer Graphics Forum 13, 5 (1994), 305–314.

Lafreniere, B., Bunt, A., Lount, M., Krynicki, F., and Terry, M.A. AdaptableGIMP: Designing a socially-adaptable interface. Proc. UIST '11 Adjunct, ACM, (2011), 89–90.

Lafreniere, B., Bunt, A., Lount, M., and Terry, M. Understanding the roles and uses of web tutorials. Proc. ICWSM '13, AAAI, (2013), 8 pages.

Lafreniere, B., Bunt, A., Whissell, J., Clarke, C.L.A., and Terry, M. Characterizing large-scale use of a direct manipulation application in the wild. Proc. GI '10, Canadian Information Processing Society, (2010), 11–18.

Leshed, G., Haber, E.M., Matthews, T., and Lau, T. CoScripter: automating & sharing how-to knowledge in the enterprise. Proc. CHI '08, ACM, (2008), 1719–1728.

Mackay, W.E. Triggers and barriers to customizing software. Proc. CHI '91, ACM, (1991), 153–160.

McGrenere, J., Baecker, R.M., and Booth, K.S. A field evaluation of an adaptable two-interface design for feature-rich software. ACM Trans. Comput.-Hum. Interact. 14, 1 (2007).

Norman, D.A. and Draper, S.W. *User Centered System Design; New Perspectives on Human-Computer Interaction.* L. Erlbaum Associates Inc., 1986.

Novick, D.G., Andrade, O.D., and Bean, N. The micro-structure of use of help. Proc. SIGDOC '09, ACM, (2009), 97–104.

Rettig, M. Nobody reads documentation. Commun. ACM 34, 7 (1991), 19–24.

Rieman, J. A field study of exploratory learning strategies. ACM Trans. Comput.-Hum. Interact. 3, 3 (1996), 189–218.

Schön, D.A. *The reflective practitioner: how professionals think in action.* Basic Books, New York, 1983.

Shneiderman, B. Promoting universal usability with multi-layer interface design. SIGCAPH Comput. Phys. Handicap., 73-74 (2002), 1–8.

Yeh, T., Chang, T.-H., and Miller, R.C. Sikuli: using GUI screenshots for search and automation. Proc. UIST '09, ACM, (2009), 183–192.

Adobe Labs Tutorial Builder. 2012. http://labs.adobe.com/technologies/tutorialbuilder/.