Evaluating Reduced-Functionality Interfaces According to Feature Findability and Awareness

Leah Findlater and Joanna McGrenere

Department of Computer Science
University of British Columbia
Vancouver, Canada
{lkf, joanna}@cs.ubc.ca

Abstract. Many software applications continue to grow in terms of the number of features they offer. Reduced-functionality interfaces have been proposed as a solution by several researchers, but evaluations have been limited in number and scope. We argue that traditional performance measures are not sufficient for these interfaces, so we introduce and distinguish feature findability and feature awareness measures. We have conducted a controlled study that demonstrates the tradeoff between these two measures: findability in a minimal layered approach was better than in the full interface alone, but subjects were more aware of advanced features if they used the full interface from the outset. A marked layered approach was also evaluated, but provided little benefit over the other interfaces. Ours is also the first experiment comparing more than one multi-layer approach to a control interface.

1 Introduction

Feature-rich user interfaces can provide necessary features for users, yet can also be overwhelming for many, especially novices. Even experts tend to use only a subset of features and can be plagued by the array of options in the menus and toolbars [13, 14]. As a solution, reduced-functionality interfaces, such as the layered interfaces approach [17], have been proposed by several researchers, either for regular use or for an initial training period. Despite these advances, evaluations been limited in number and have focused largely on the benefits of reducing functionality. Our goals are: (1) to introduce new evaluation metrics, findability and awareness, that reveal a more comprehensive understanding of the impact of reducing functionality; and (2) to apply these metrics to compare two 2-layer interfaces to a control interface.

Several methods for reducing functionality have appeared in the research literature and in commercial applications. The layered interfaces approach, for example, gradually introduces new functionality to the user by starting with a simple interface containing a core set of features, and allowing the user to control his transition to increasingly feature-rich interface layers [17]. In contrast to many methods for personalizing menus and toolbars that block individual features (e.g., [10, 15]), layered interfaces offer a relatively coarse-grained approach; that is, relatively large sets of functionality are grouped in layers. Examples to date allow the user to transition from between 2 to 8 layers, and evaluation has been mainly qualitative [7, 8, 16, 17]. A related approach to reducing functionality is the multiple interfaces approach, which offers users a “personal” interface in addition to the full interface of the application [15]. The user can easily switch between the two interfaces and specify the functions contained in his personal interface. Another earlier reduced-functionality approach is the training wheels interface, which blocks the use of advanced functionality for novice users but does not remove it from the interface [5]. In contrast to these, adaptive mechanisms can also be used to automatically reduce functionality; for example, Microsoft Office 2003 provides personal menus that contain only an automatically-generated subset of features when they are initially opened. Mixed-initiative mechanisms that combine both system and user control have also been proposed, most commonly through the use of adaptive suggestions to support the user's customization (e.g., [4]).

Evaluations of these reduced-functionality approaches have shown that they can make novice users faster, more accurate and more satisfied with the interface [5], and that such approaches can be preferred by a large proportion of intermediate and advanced users [15]. However, we argue that satisfaction and initial speed only reflect part of the impact of functionality reduction. When features are removed from the interface, the user's level of awareness of the full feature set is also affected. A severely feature-reduced interface may promote ease of accessing functions, but likely impedes awareness of those functions only available in the full application interface. A survey of 53 users of Microsoft Word 97 reflects this tension: while many of the users requested that their unused functions be “tucked away”, many also indicated that it was important that they be continually able to discover new functions [14].

To address this problem, we distinguish two new evaluation measures: *findability* and *awareness*. Both of these can impact performance. Generally speaking, findability measures the speed with which users can find known functions, and awareness measures the degree to which users are conscious of the full set of available functions. While several previous evaluations have measured performance that maps to our findability measure (in some cases findability related to transferring from a simpler to a more complex interface) [2, 5, 6, 11], none have included a measure of awareness.

We have conducted an experiment to empirically validate the tradeoff between findability and awareness and to provide the first controlled comparison of more than one multi-layer interface. Our study compares two 2-layer interfaces, Minimal and Marked, to a control condition and shows that findability in the Minimal approach is significantly better than in the Control, but that subjects are more aware of features in the Control. The Marked approach provided little benefit over the other two.

2 Findability and Awareness Definitions

The distinction between findability and awareness allows for a more nuanced evaluation, which is particularly important for reduced-functionality designs, where the potential impact on awareness may be greater than in more traditional approaches.

– **Findability** is the speed with which the user can locate a function she knows exists. The set of findable functions includes those the user has already used, those the user has heard about from others or from documentation, those the user has used in a previous version or reduced-functionality version (or layer) of the same application, and those the user has a strong basis for believing exist (e.g., `Save` is found in most document-centric applications).

– **Awareness** is the degree to which the user notices (through using an application) and can recall functions which do not fall into the findable function set above.

Findability is essentially the speed of accessing a specific set of functions. By contrast, the speed to complete a task as a whole is a more compound measure, impacted potentially by both findability and awareness. For example, the speed with which a user completes a task is related to the time it takes her to do the steps she is familiar with (using the set of findable functions) and those steps that she needs to "discover" how to complete. The time to complete the latter steps will be in part related to her prior awareness (of the set of aware functions). We propose specific techniques for measuring findability and awareness in our experiment, described later.

3 Study Motivation

Findability and awareness are likely impacted by many design factors and we focus on two of these in our study. The first is *change direction*. Most approaches to reducing functionality begin with a small feature set which increases over time. For example, layered interfaces [17], multiple interfaces [15], training wheels interfaces [5], incremental interfaces [3], and the MS Office adaptive menus all fit within this category. An approach that moves from less to more functionality should maximize the findability of those functions in the initial interface states, but the awareness of potentially useful advanced functionality is surely compromised. Moving in the opposite direction, from more to less functionality, should allow for improved initial awareness of the full functionality set of an application but initial findability suffers. Our study compares the effect of using a reduced-functionality design that moves from less to more functionality versus working in the full application interface. Since some related evaluations have used metrics that are similar to findability, we use these to inform the findability hypotheses of our study (given later in Section 4.7). In particular, related work has shown that: (1) novice users were faster with a training wheels version of an interface than the full interface [5], and (2) users who initially used the training wheels interface performed no differently on a follow-up similar task in the full interface than those who had used the full interface from the outset [6].

The second design factor incorporated into our reduced-functionality designs is *visibility of change*. When functionality is blocked in an interface, it can remain visually unchanged, be visually marked, or be completely removed from the visual interface. Removing the visual affordance associated with blocked functions (e.g., [15, 17]) should emphasize the findability of the remaining functions. On the other hand, the training wheels approach, which blocks functions but leaves them visually unchanged [5], should allow users to develop an awareness of the more advanced features available in the full interface. Visually marking, yet not fully hiding a widget, may offer a compromise between the two extremes. To understand whether this is the case, our study compares a design that visually marks blocked features to one that completely hides blocked features.

Previous subjective results are conflicting and highlight the importance of including both subjective and objective measures in our evaluation. For example, the original training wheels approach was shown to be both more efficient and preferred by novice users [5], but a more recent, less tightly-controlled study suggests that users may not see the value of a training wheels approach in a graphical user interface [2].

4 Experimental Design

This study compares two 2-layer interfaces to a control condition (a default full interface) based on findability and awareness. We chose Microsoft PowerPoint 2003 as our experimental application. Though not as complex as some applications (e.g., animation software), PowerPoint does not require specialized domain expertise, easing subject recruitment, and the menus and toolbars are highly programmable.

4.1 Interviews to Define Command Sets

To inform the design of the experimental conditions and tasks, we interviewed 10 frequent users of PowerPoint XP and 2003 from varying backgrounds (academic, business and medical). For each of the 361 selectable items (commands) found in the pull-down menus and on the 12 main toolbars, we asked users to specify their frequency of use (never, irregular or regular). From this, we defined a baseline interface, composed of menus and toolbars that were used by at least half of the 10 users. This included all the menus and the Standard, Formatting, Drawing, and Picture toolbars, a total of 260 commands; it did not include several context-based toolbars that are not visible by default. (Two duplicate commands were also removed: Slide Show appeared twice in the menus, and Font Color appeared twice in the toolbars.)

We then categorized the commands in our baseline interface according to two independent dimensions: (1) **basic** commands are used by at least 8/10 of our users, while the remaining are **advanced**; and (2) **generic** commands are common to the default layout of other MS Office or MS Windows applications (such as Save) while the remaining are considered **specific** to PowerPoint. These categorizations impacted our interface and task designs, and we often refer to the intersections of the sets, for example, generic-basic commands. The relative frequencies are shown in Table 1. We note that our data and categorization offer only an approximation of PowerPoint commands and their usage. This is sufficient for our experimental setup, but more accurate usage data and categorization adjustment would be needed for a deployable reduced interface for PowerPoint.

Table 1. Breakdown of baseline command set (from the menu bar and the Standard, Formatting, Drawing, and Picture toolbars).

	Basic	Advanced	Total
Specific	12 (5%)	123 (47%)	135 (52%)
Generic	32 (12%)	93 (36%)	125 (48%)
Total	44 (17%)	216 (83%)	260 (100%)

4.2 Conditions

We evaluated three conditions: two 2-layer conditions (Minimal and Marked), and a control condition (Control). In the layered conditions, subjects completed a simple task in the initial interface layer of the respective condition, then transitioned to a full interface layer for a second, more complex task. This simulated the predicted use of a layered interface, where users start in a reduced layer to complete easier tasks, then transition to more complex layers for advanced tasks [17]. In the Control condition, subjects completed both the simple and complex task in the full interface layer. Each of the three layers is described below and Figure 1 shows samples of their menus and toolbars. Context menus and keyboard shortcuts are disabled in all conditions to constrain the interaction and to focus on the persistent visual complexity that comes from the menus and toolbars. The MS Office adaptive menus are also turned off.

Full Interface Layer: The baseline interface, which contains 260 commands, as described above.

Minimal Interface Layer: Contains only the 44 basic commands (both general and specific). Since the Tools menu, Window menu, and Picture toolbar contained no basic commands they do not appear in this layer.

Marked Interface Layer: Extends Carroll and Carrithers' training wheels approach [5] by visually marking as well as blocking access to all advanced commands, leaving only the 44 basic commands accessible. Marking is achieved by fading a command's icon (if it has one) and adding a small ‘x’ (see Figure 1, *B* and *E*); if selected, a dialog box informs the user that “This item is unavailable for this task”. Limitations in PowerPoint's API forced two secondary design decisions: (1) submenus with all blocked commands are completely removed, and their parent item is visually marked, which reduces the total command set to 210; and (2) only the icon is changed on blocked menu items (ideally the APIs would have allowed us to pilot test options for changing the background or text colour as well).

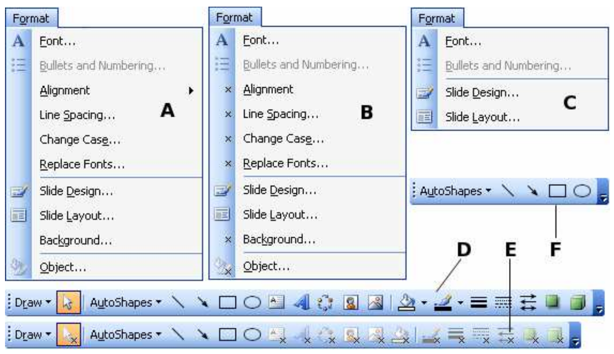


Figure 1: Sample menus and toolbars from the three experimental conditions: *A*, *B*, and *C* show the Format menu for the full, marked and minimal layers, respectively; *D*, *E*, and *F* show the Drawing toolbar for the full, marked and minimal layers, respectively. (The marked toolbar is narrower than the full one because the drop-down arrows on some commands could not be replicated for blocked functions.)

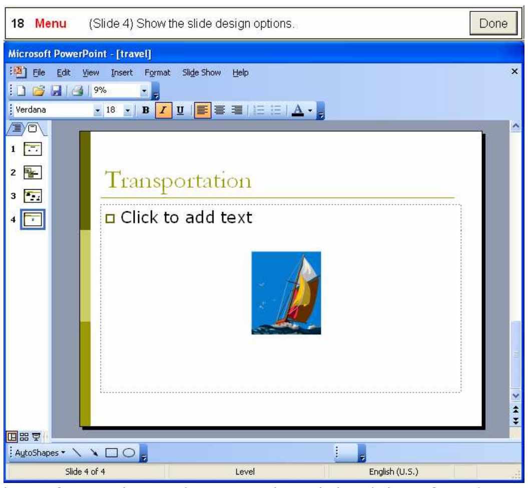

18 Menu (Slide 4) Show the slide design options.
Done
Microsoft PowerPoint - [travel]
File Edit View Insert Format Slide Show Help
Verdana
18
Transportation
Click to add text
AutoShapes
Slide 4 of 4
Level
English (U.S.)

Figure 2. Screenshot of experimental system in minimal interface layer; at the top of the screen, the current instruction specifies that a menu item is required.

The Minimal and Marked conditions both reduce functionality by providing a small feature set which increases over time. Comparing Minimal to Marked also provides insight into the impact of different types of visibility of blocked functions: as discussed in Section 3, we anticipated that visually distinguishing, but not removing blocked functions could offer a compromise between findability and awareness.

4.3 Task

We designed two tasks (simple and complex), each consisting of a list of step-by-step instructions to modify a pre-existing short presentation. On each step, the user performed one interface operation: steps requiring specific-basic or specific-advanced commands were interspersed among generic-basic commands, navigation, and text entry steps to create a realistic task flow. The instruction indicated when to use a menu or toolbar, but did not specify the exact name of the command. For example, Figure 2 shows a screenshot of the experimental system with an instruction that specifies that the subject should use a menu command to show the slide design options (which maps to Slide Design command in the Format menu).

The interested reader can see the command distribution by task in Figure 3. The tasks were as follows:

- **Simple task:** This relatively short task could be completed in any of the three interface layers. It included all 12 specific-basic commands, 6 of which were repeated twice to increase task length (18 specific-basic invocations in total); 12 generic-basic commands were also included.
- **Complex task:** This longer task could only be completed in the full interface layer, and introduced advanced functionality, such as animation. It included 18 specific-advanced commands in addition to the exact same set of commands used in the simple task.

4.4 Design, Subjects and Apparatus

A between-subjects single factor (interface) design was chosen to prevent any learning confounds. Thirty subjects (17 females) between the ages of 19-55 were randomly assigned to each of the three conditions (10 per condition). Subjects were students and community members recruited through campus advertising and a local subject pool. They were screened so that they had either never used PowerPoint or had at most infrequently loaded and viewed a presentation created by others. Each subject was paid $20 to participate.

The experiment used a 1.1 GHz Pentium PC with 640 MB of RAM, 18” LCD monitor, and running MS Windows XP and Office 2003. The experimental versions of PowerPoint, one of which is shown in Figure 2, were coded in Visual Basic for Applications 6.0. Instructions were given one at a time at the top of the screen. When the subject believed she had correctly completed a step, she clicked the “Done” button and the next instruction appeared. The system recorded all timing data.

4.5 Procedure

The experiment fit in a two hour session. Subjects first completed a background questionnaire and were given a brief introduction to the experimental system using the initial layer of their condition (Minimal, Marked or Control). The introduction covered where the menus and toolbars were located, how to navigate to a particular slide, and, for the Marked condition, the behaviour of a marked function. Subjects then completed the simple task in their assigned interface, followed by a short questionnaire, and then a ten minute break with a distractor activity. Next, all subjects used the full interface to complete the complex task, followed by another short

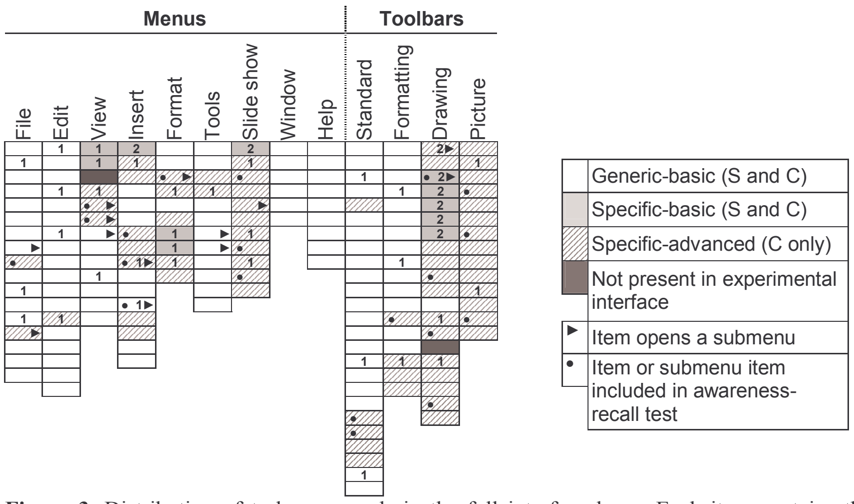


Figure 3. Distribution of task commands in the full interface layer. Each item contains the number of times it was used in each task (S = simple, C = complex). Items that open submenus are shaded based on their submenu’s most advanced item.

questionnaire. Finally, subjects were given an awareness-recall test, described below, followed by a five minute interview and discussion period to elicit additional subjective feedback.

During the tasks, subjects were told that the goal was twofold: to complete the steps in a timely manner and to familiarize themselves with the interface while doing so. They were told they could not ask for help on completing steps. If a subject had particular difficulty with a step, the system timed out after two minutes and the experimenter showed the subject how to complete the step so the task could continue. Subjects were allowed to make errors, but if an error was critical to completing subsequent steps, the experimenter corrected the situation.

4.6 Measures

Each step was measured as the time elapsed until the user clicked the “Done” button.

Findability. We differentiate 3 types of findability:

1. *Used-before findability:* Functions a user has used before in the same interface. By design, all our subjects were new to PowerPoint, which means we were not able to measure the used-before findability. (At first glance it may appear that the generic-basic steps could be used for this measure; variation in previous MS Office experience, however, would have confounded the result.)
2. *Heard-about findability*: Functions a user has heard exist (from others or documentation), but has never used. Time to complete the 18 specific-basic steps in the simple task approximates heard-about findability, because the instructions loosely simulate having heard from a colleague that such a function exists.
3. *Transfer findability:* Functions a user has used before in a previous or reduced-functionality version (or layer) of the same application. Transfer findability is measured as the time to do the 18 specific-basic steps in the complex task.

Awareness. We measured awareness using two methods:

1. *Direct awareness:* As a direct measure of awareness we administered a recall test. This was a questionnaire that listed 20 of the specific-advanced functions that were present in the full interface layer but were not used for either of the two tasks, and five distractor functions (commands that do not exist in PowerPoint, but could be believed by a novice to exist; e.g., Assign Slide Manager). Icons were also provided for those commands that had one. Half of the valid commands were menu items and half were toolbar items.

   The distribution of commands tested is shown with a ‘•’ in Figure 3. For each item, subjects noted if they definitely remembered it. We then calculate the *corrected recognition rate*, a commonly-applied method in psychology to account for individual variation in the amount of caution a subject applies when responding to a memory test; it is simply the percentage of targets correctly remembered minus the percentage of distractors incorrectly chosen [1]. When an individual user's corrected score was negative, we assigned her a score of zero.

2. *Indirect awareness*: We used time to complete the 18 specific-advanced commands in the complex task as an indirect measure of awareness. These commands are used for the first time in the second task, which all subjects complete using the full interface layer. A difference in access time for these commands should be a result of different levels of awareness gained during the simple task.

Secondary objective measures. Our secondary objective measures included timeouts, errors, and exploration. Errors only included incorrectly completed steps not already counted in timeouts. Exploration was defined as the number of toolbar or menu items that a subject selected before selecting the target item, or in the case of incorrectly-completed steps, before clicking "Done."

Subjective measures. We report six subjective measures. After each task, all subjects ranked on a 5-point Likert scale how overwhelmed they felt by the amount of "stuff" in the menus and toolbars, and how difficult it was to navigate through them. Additionally, after completing the second task, Minimal and Marked subjects ranked on a 5-point Likert scale how easy they found it to transition from the menus and toolbars in the first task to those in the second, and whether they preferred those in the first task to those in the second. In follow-up interviews, these subjects were also asked which version they would prefer to use in the future, and whether or not they could see themselves switching between the two versions for different tasks.

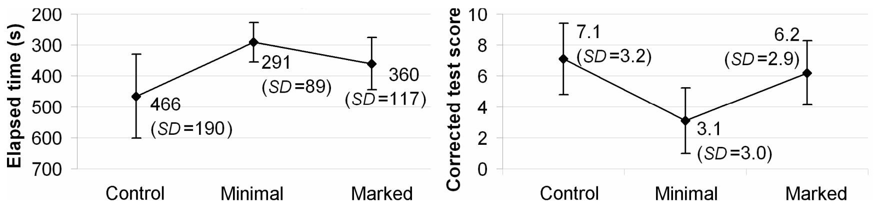
4.7 Hypotheses

Our main hypotheses, based on discussion in Section 3, were as follows:
H1. *Heard-about findability:* Minimal is faster than Marked, and Marked is faster than Control.
H2. *Transfer findability:* No difference between conditions.
H3. *Direct and indirect awareness:* Control is better than Marked, and Marked is better than Minimal.

5 Results

We performed one-way ANOVAs on each dependent measure, except where noted. All pairwise comparisons were protected against Type I error using a Bonferroni adjustment. We report measures which were significant ($p < .05$) or represent a possible trend ($p < .10$). Along with statistical significance, we report partial eta-squared (η^2), a measure of effect size, which is often more informative than statistical significance in applied human-computer interaction research [12]. To interpret this value, .01 is a small effect size, .06 is medium, and .14 is large [9].

On average across all conditions, the simple task took 15.5 minutes (*SD* = 6.3) while the complex task took 26.1 minutes (*SD* = 5.3).



(a) Heard-about findability (mean) **(b) Direct awareness (mean)**

Figure 4. Means of findability for the simple task and direct awareness, by condition; findability is displayed using a reverse scale for time, thus higher scores are better in both graphs ($N = 30$).

5.1 Findability

As expected, the conditions did impact heard-about findability significantly differently ($F(2,27) = 4.03$, $p = .029$, $\eta^2 = .230$). The means and standard deviations are shown in Figure 4(a). Pairwise comparisons showed that Minimal subjects were significantly faster on this measure than Control subjects ($p = .027$), but no other comparisons were significant.

Also as expected, no significant effect of interface was found on transfer findability ($F(2,27) = .708$, $p = .501$ $\eta^2 = .050$). The overall average to complete the specific-basic steps in the complex task was 211 seconds ($SD = 68$).

For completeness, we ran an ANOVA on the generic-basic steps in both tasks, and, not surprisingly, found no significant differences. This suggests that previous experience with MS Office dominated over interface condition.

5.2 Awareness

For direct awareness, there was a significant main effect of condition on corrected recognition rate ($F(2,27) = 4.81$, $p = .016$, $\eta^2 = .263$). Table 2 shows the hit rate, false alarm (distractor) rate and corrected recognition rate from the awareness-recall test. Control subjects had an average corrected recognition rate of 7.1 out of 20 items ($SD = 3.2$), which was significantly more than Minimal subjects ($p = .019$), who only remembered on average 3.1 items ($SD = 3.0$). A trend suggested that Marked subjects, scoring 6.2 ($SD = 2.9$) on average, were aware of more than Minimal subjects ($p = .090$). The trade-off between findability and direct awareness is shown graphically in Figure 4. High findability is matched with low awareness and vice versa.

Unexpectedly, no significant effect of interface condition was found on indirect

Table 2. Average awareness scores as percentage of items answered affirmatively ($N = 30$ subjects).

	Correct targets (%)	**Incorrect distractors (%)**	**Corrected recognition (%)**
Control	.46 (*SD* = .16)	.1 (*SD* = .17)	.36 (*SD* = .16)
Minimal	.36 (*SD* = .18)	.24 (*SD* = .30)	.16 (*SD* = .15)
Marked	.43 (*SD* = .19)	.12 (*SD* = .14)	.31 (*SD* = .14)

awareness ($F(2,27) = .172$, $p = .843$, $\eta^2 = .013$). Across all conditions, the specific-advanced steps took on average 780 seconds ($SD = 167$).

5.3 Timeouts, Exploration, and Errors

There was a significant main effect of condition on timeouts for the simple task ($F(2,27) = 4.18$, $p = .026$, $\eta^2 = .236$). The one significant pairwise comparison was between the Minimal and Control. Minimal subjects never timed out, which was significantly less than the average 1.6 timeouts ($SD = 2.1$) for Control subjects ($p = .030$). This result suggests that our heard-about findability measures favoring Minimal are conservative, since Control subjects' scores would have been worse without timeouts than they were with timeouts. In the complex task there were more timeouts (across conditions: $M = 2.3$, $SD = 1.6$), but not surprisingly no significant effect of condition was found.

There was a main effect of condition on exploration in the simple task ($F(2,27) = 4.79$, $p = .017$, $\eta^2 = .262$). Control subjects selected on average 18.6 items ($SD = 13.3$) while exploring, which pairwise comparisons showed to be significantly more than the average of 6.9 ($SD = 6.5$) for subjects in the Marked condition ($p = .023$). A trend also suggested that Control subjects explored more than the 8.7 average ($SD = 5.4$) of Minimal ($p = .066$). In the complex task, no significant differences were found.

Mean error rates were uniformly low (simple task: $M = 1.8$, $SD = 1.6$; complex task: $M = 1.7$, $SD = 1.6$), and no significant differences were found. The experimenter stepped in on average 0.9 times per participant to correct errors that would have affected further steps ($SD = 1.3$).

5.4 Subjective Responses: Questionnaires and Interviews

Questionnaire responses showed that there was a significant main effect of condition on the degree to which subjects felt overwhelmed by how much "stuff" there was in the menus and toolbars after the first task ($F(2,27) = 4.50$, $p = .021$, $\eta^2 = .250$). Pairwise comparisons showed that Marked subjects felt more overwhelmed than Minimal subjects ($p = .020$). In terms of navigation, a trend also suggested that interface condition may have had an effect on the degree to which subjects felt it was difficult to navigate through the toolbars in the complex task ($F(2,27) = 2.54$, $p = .098$, $\eta^2 = .158$).

Using one-tailed *t*-tests, we evaluated the Likert-scale questions completed by only Minimal and Marked subjects. Marked subjects felt more strongly than Minimal subjects that they were easily able to transition between the menus and toolbars used in the two tasks ($t(18) = 1.89$, $p = .038$). Minimal subjects preferred their initial interface to the full interface more than Marked subjects did ($t(18) = -2.76$, $p = .007$).

During interviews, participants were asked which interface they would prefer to continue using. Minimal subjects overwhelmingly chose the minimal layer over the full layer (9 subjects, $\chi^2(1,10) = 6.4$, $p = .011$) while Marked subjects were equally strong in their preference for the full layer (9 subjects, $\chi^2(1,10) = 6.4$, $p = .011$). This replicates recent subjective findings of training wheels on a modern word processor

[2]. Trends suggested that subjects who had used the minimal interface could see themselves switching between a minimal and full layer for different tasks (8 subjects, $\chi^2(1,10) = 3.6$, $p = .058$), whereas subjects who used the marked interface felt exactly the opposite about their interface (8 subjects, $\chi^2(1,10) = 3.6$, $p = .058$).

5.5 Summary

We summarize our findings in terms of our hypotheses:

H1 *Partially supported:* Minimal had significantly better heard-about findability than Control, but there were no other significant differences.

H2 *Supported:* No difference for transfer findability.

H3 *Partially supported:* In terms of direct awareness, Control was better than Minimal and a trend suggested that Marked was better than Minimal. There were no other significant differences for direct awareness, nor any for indirect awareness.

6 Discussion and Future Work

Awareness measure adds value. The comparison of the Control condition to the Minimal 2-layer condition shows that there is a measurable tradeoff between findability and awareness. Taken in isolation, the findability results replicate related work on training wheels interfaces [5], and could lead us to reach the straightforward conclusion that a minimal 2-layer interface is better than the full interface alone: it was faster in the first task and had no cost when transferring to the full interface layer for the second task. By teasing apart performance and demonstrating that improved findabilty can come at a cost of decreased awareness, we provide a richer understanding of the experience.

Two-layered minimal interface is promising. The qualitative and quantitative data together suggest that a two-layer minimal interface offers significant advantages over a default full functionality interface alone, although the findings need to be tempered by its reduced awareness. Eight out of 10 subjects indicated that they would prefer to have a minimal interface in conjunction with a full interface. Subjects had better findability in the simple task in the Minimal condition, and transfer findability in the complex task was no worse than in the Control condition. The Control condition did, however, have better direct awareness. We speculate that if users could freely switch between a minimal and full interface the impact on awareness could be smaller, but further research is required to substantiate that claim.

Two-layered marked interface is problematic. We theorized that visually marking, yet not removing, blocked functionality would offer a compromise between findability and awareness. This was not shown in our study. A trend-level result suggested the Marked condition improved direct awareness over the Minimal condition. Combining this with the means in Figure 4 suggests that the Marked condition may have a small positive effect on findability and awareness but we simply did not have sufficient power to detect this. However, the preference of 9 out of 10 subjects for the full interface over the marked one is a strong indicator that even if a 2-layer marked interface offers a small performance improvement over a full interface, it would not achieve widespread adoption. The marked interface, in its current form, is not a fruitful direction for further research. An alternative would be to try a different form of visual marking that allows users to filter out advanced items more easily, which could lessen the negative reaction. Further work would be needed to investigate this.

Measuring awareness is challenging. The two measures of awareness in our study produced inconsistent results. The direct measure, assessed by questionnaire, partially supported our hypotheses that the Minimal condition would have the least awareness, the Control the most, and the Marked condition would be in between. By contrast, the indirect measure, assessed by performance on the specific-advanced commands in the complex task, provided no support for our hypotheses. This is likely due to a lack of power for the indirect measure: the impact of awareness on the complex task could have been small relative to the overall difficulty and time needed to find specific-advanced features in the full interface. Beyond using a within-subjects design, one possibility to increase power is to use a less directed task that would encourage subjects to explore more during the simple task, thus magnifying any differences in awareness before starting the complex task. For example, leaving subjects to discover commands that will help them replicate a sample final presentation should encourage more exploration of the interface. Our challenge in achieving convergent evidence for awareness points to a need for further investigation of how it should be measured.

Need exists for a design and evaluation framework. Despite advances in research on reduced-functionality interfaces, there has never been an overall framing of the design and evaluation space. Our study focused exclusively on novice users, and explored the design factors of *change direction* and *visibility of change*. It provides a first step towards understanding how design decisions impact findability and awareness. Further work, however, is needed to identify and map out the set of major design factors that impact findability and awareness and to understand the nature of this impact. We must also strive to understand the maximum possible effect that a reduced-functionality interface can have on findability and awareness. Finally, our study protocol did not have users interacting with the mechanism to reduce features since the experimenter set the interface layer for each task. Although the goal of reduced-functionality designs is to reduce complexity, the very inclusion of a mechanism to do so adds some complexity to the interface. This impact needs to be outweighed by the beneficial effects of working within reduced functionality.

7 Conclusion

There is a strong tendency to add, rather than eliminate, features in new versions of software applications. The need for managing functionality is thus increasing, which underscores the motivation behind approaches to reducing functionality. We have introduced findability and awareness, two evaluation measures that offer a decomposition of more traditional performance measures. They allow for a more nuanced comparison of different designs. In a controlled laboratory study to evaluate layered interfaces, we have demonstrated a measurable tradeoff between findability and awareness: findability in a minimal layered interface approach is better than in the full interface alone, but subjects were more aware of advanced features if they used the full interface from the outset. Previous research on reduced-functionality designs had largely focused on the benefit of such approaches, including improved initial performance and reduced visual complexity. Our work reveals a more comprehensive understanding of the impact of reducing functionality.

Acknowledgements. We thank IBM Centers for Advanced Studies and NSERC for funding for this work, and Rebecca Hunt Newbury for helping to conduct the experimental sessions.

References

1. Baddeley, A. (1976). *The psychology of memory*. New York: Basic Books.
2. Bannert, M. (2000). The effects of training wheels and self-learning materials in software training. *Journal of Computer Assisted Learning*, 16(4), 336-346.
3. Brusilovsky, P. and Schwarz, E. (1997). User as student: Towards an adaptive interface for advanced web-based applications. *Proc. of the Sixth International Conference on User Modeling*, 177-188.
4. Bunt, A., Conati, C. and McGrenere, J. (2007). Supporting interface customization using a mixed-initiative approach. *Proc. Intelligent User Interfaces*, 92-101.
5. Carroll, J.M. and Carrithers, C. (1984). Training wheels in a user interface. *Communications of the ACM*, 27(8), 800-806.
6. Catrambone, R. and Carroll, J.M. (1987). Learning a word processing system with training wheels and guided exploration. *Proc. SIGCHI/GI,* 169–174.
7. Christiernin, G.L., Lindahl, F., and Torgersson, O. (2004). Designing a multi-layered image viewer. *Proc. NordiCHI '04*, 181–184.
8. Clark, B. and Matthews, J. (2005). Deciding layers: Adaptive composition of layers in a multi-layer user interface. *Proc. HCI International.*
9. Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd ed.). Hillsdale, NJ: Lawrence Erlbaum Associates.
10. Findlater, L. and McGrenere, J. (2004). A comparison of static, adaptive and adaptable menus. *Proc. ACM CHI 2004*, 89-96.
11. Franzke, M., Rieman, J. (1993). Natural training wheels: Learning and transfer between two versions of a computer application. *Proc. VCHCI 1993*, 317-328.
12. Landauer, T. (1997). Chapter 9: Behavioral research methods in human-computer interaction. In M. G. Helander, T. K. Landauer and P.V. Pranhu (Eds), *Handbook of Human-Computer Interaction* (2nd ed., pp. 203-227). Amsterdam: Elsevier Science B.V.
13. Linton, F., Joy, D., Schaefer, H.-P., & Charron, A. (2000). Owl: A recommender system for organization-wide learning. *Educational Technology & Society*, 3(1), 62-76.
14. McGrenere, J., and Moore, G. (2000). Are we all in the same "bloat"? *Proc. of GI 2000*, 187-196.
15. McGrenere, J., Baecker, R., and Booth, K. (2002). An evaluation of a multiple interface design solution for bloated software. *CHI Letters*. 4(1), 163–170.
16. Plaisant, C., Kang, H., and Shneiderman, B. (2003). Helping users get started with visual interfaces: Multi-layered interfaces, integrated initial guidance and video demonstrations. *Proc. HCI International 2003: Volume 4 Universal Access in HCI*, 790-794.
17. Shneiderman, B. (2003). Promoting universal usability with multi-layer interface design. *Proc. of the 2003 Conference on Universal Usability*, 1–8.