Parameter Selection in Keyboard-Based Dialog Boxes

Jeff Hendy, Juliette Link, Kellogg S. Booth & Joanna McGrenere
Department of Computer Science, University of British Columbia, Vancouver, BC, V6T 1Z4, Canada
{jchendy, jlink, joanna, ksbooth}@cs.ubc.ca

ABSTRACT
Recent keyboard-based alternatives to WIMP interfaces do not have good support for commands that require multiple parameters. We remedy this by extending a previous design and mimicking dialog boxes to provide good visual feedback while still keeping the advantages of keyboard input. A laboratory study showed the new technique to be competitive with dialog boxes on speed and error rate, but strongly preferred over dialog boxes by experienced command line users. This is a marked improvement over the previous design, which was also preferred by the target user group but did not compete with dialog boxes in terms of performance.

Author Keywords
Command-line interfaces, dialog boxes, WIMP

ACM Classification Keywords
H52 [Information interfaces and presentation]: User Interfaces – Graphical user interfaces.

General Terms
Design

INTRODUCTION
Many systems allow keyboard input to augment or replace standard WIMP interfaces. The goal of these keyboard-based systems is to increase satisfaction and performance for experienced command line users. Examples include Quicksilver [1], Enso [3], Inky [4], GEKA [2], and Ubiquity [5]. They provide well thought out ways to specify commands. Not as much has gone into how parameters are specified, especially for commands with multiple parameters. In WIMP interfaces, parameters are often specified through dialog boxes, especially multiple parameters. Evidence suggests experienced computer users strongly dislike dialog boxes [2]. Quicksilver and Enso support only one parameter. Ubiquity and Inky support multiple parameters, but use imprecise syntaxes that can make parameter entry confusing. They are designed specifically for web-based commands with smaller, simpler sets of parameters. In contrast, GEKA has a keyword-based parameter system in which any number of parameters can be precisely input in any order, but it showed lower performance compared to existing WIMP dialog boxes [2].

CHI 2011, May 7–12, 2011, Vancouver, BC, Canada.

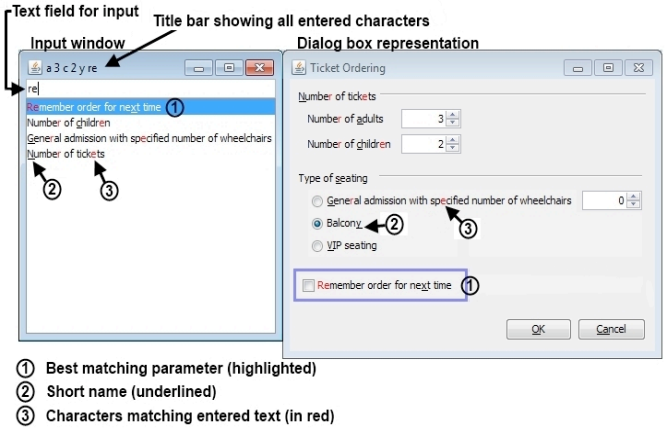

Figure 1: KDB graphical feedback. The characters "a 3 c 2 y re" have been entered. This sets the values of Number of adults (3), Number of children (2), and Type of seating (Balcony). The final input, "re", is selecting which parameter will be set next.

We describe a new method for specifying parameters, keyboard-based dialog boxes (KDB), that allows input of any number of parameters in any order. Our goal is to increase performance, improve experience, and minimize cognitive load. To accomplish this, KDB provides graphical feedback that looks and behaves very much like the dialog boxes users are familiar with, but with the speed of a command line interface. A lab experiment with experienced command line users showed our new method is preferred to and performs competitively with WIMP dialog boxes.

KDB DESIGN
KDB is based on GEKA. It makes minor changes to the GEKA command language but replaces its graphical feedback with a modified version of each command's existing dialog box. These changes should give KDB a strong performance advantage over GEKA and make it competitive with WIMP dialog boxes.

Multiple parameter entry utilizes auto-completion to quickly specify parameter name and value pairs. Example parameter specifications for the **`print`** command include:

`pages 2`
set the value of **`pages`** *to* **`2`**
`printer downstairs`
set the value of **`printer`** *to* **`downstairs`**
`downstairs`
because **`downstairs`** *can only be a value for* **`printer`***, this sets* **`printer`** *to* **`downstairs`**

For parameter name selection, a ranked list of possible

matches is generated after each character is typed. Typing more characters refines the list. Pressing **SPACE** accepts the top-ranked match and moves on to value entry. If a parameter has a discrete set of possible values, the value is selected using the same auto-complete mechanism. Otherwise, the value is typed in its entirety. Again pressing **SPACE** accepts the value and allows the user to specify another parameter. If the top-ranked parameter name has a Boolean value, pressing **SPACE** toggles the value and immediately moves on to another parameter.

Each parameter also has a short name, which is a short sequence of characters that unambiguously identifies the desired parameter name. Auto-completion uses our GEKA four-category algorithm [2] to order possible matches: *exact match*, *prefix match, substring match, subsequence match*. Matches within each category are sorted alphabetically. Up/down arrow keys scroll the ranked list.

There are two components to KDB graphical feedback, the input window, which closely resembles the graphical feedback in GEKA, and a new dialog box representation, which provides additional visual feedback. Figure 1 shows these two components. The input window has a text box where input is typed and a list of possible matches to the input is shown. The best match is highlighted and appears first. For all entries, the matching characters are in red. The short name is underlined. The dialog box representation provides graphical feedback identical to a command's WIMP dialog box. Because users are already familiar with this dialog box, we expect that displaying it will allow users to quickly identify parameter names they want to use.

The best matching parameter highlighted in the input window is shown with a blue box in the dialog box. The dialog box also has the short name for each parameter underlined and the characters that match the entered text are in red. This gives users the same information whether they look for feedback in the parameter list or the dialog box. Highlighting also allows users to quickly verify that the parameter they want is selected. If the best matching parameter is located in a sub-dialog box or a different tab within the dialog box, the feedback is automatically updated to show the location with that parameter.

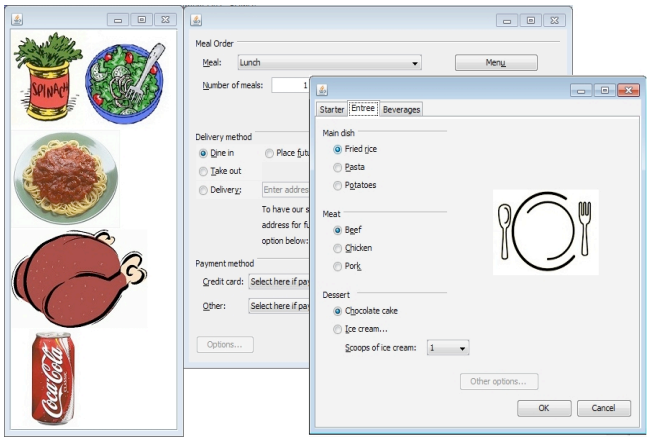

Figure 2: The dialog box condition showing the long task for order food. The user has already clicked "menu" to open a sub-dialog. The left window shows that "spinach salad," "pasta," "chicken," and "coke" must be selected. Salad and drink options are in different tabs. Parameters not listed in the left window use default values and are not set by participants.

LABORATORY EXPERIMENT

We conducted a laboratory experiment to evaluate KDB against the goals of high performance and preference over dialog boxes (DBs) by experienced command line users. We also included a GEKA condition to compare KDB to an existing keyboard-based interface.

The tasks were to specify parameters for one of four commands. This was done in three conditions: KDB, GEKA as implemented by Hendy et al. [2], and DBs. With DBs, participants could use any combination of mouse clicks, **TAB** key navigation, or keyboard mnemonics.

The experiment tested familiar commands from a standard application and unfamiliar commands that we invented. It also tested simple commands, specified in a single DB, and complex commands, which involved multiple tabs and sub-dialog boxes. The four commands are listed below.

	Familiar	Unfamiliar
Simple	`Insert table`	`Order tickets`
Complex	`Print`	`Order food`

For the two familiar commands we implemented replicas of Word 2003 DBs (including parameter names). All of our participants reported using these DBs in Word 2003. **`Order tickets`** was created for this experiment; it has the same number of parameters and a similar DB layout to **`insert table`**. **`Order food`** was similarly analogous to **`print`**. The two invented DBs are shown in Figures 1 & 2.

For each command in the experiment, we created a short task, in which only one parameter value was to be specified, and a long task, in which four values were to be specified. Thus, there were eight different task combinations. For the complex commands, both the short and long tasks required the use of at least one sub-dialog box. As in an earlier study of GEKA [2], all tasks were prompted using image-based descriptions to avoid biases introduced by using text descriptions. Figure 2 shows the prompts for the long **`order food`** task. For each task, the command was pre-selected in the interface, meaning that the dialog box or GEKA parameter pane was already open. The participant needed only to select the specified parameter values. This was done to isolate parameter selection times.

Participants

There were 12 participants (3 females). In a pre-screening questionnaire all reported command line experience and correctly answered at least two of three command line knowledge questions. Participants received $20. The top third fastest participants got a $5 bonus to motivate quick and accurate performance.

Procedure

Each participant completed a single two-hour session. A session began with an introduction. Participants then completed all trials in a particular condition before moving on to the next condition. Presentation order of the three conditions was counterbalanced. Each condition began with an introduction to the method used and a practice block in which the eights tasks were each completed once. During a practice block, participants could ask questions and refer to printouts of the task images that referred to each parameter. No aids were permitted during experimental blocks.

Presentation order of the four commands was randomized across participants but remained constant across the three conditions for each participant. The order of the two task lengths was similarly randomized across participants. During each condition, participants completed one trial for each of the two task lengths for a command and then repeated this pair four more times before moving on to the next command. Completion time was recorded for each trial from when the participant dismissed a begin-task prompt (by clicking the mouse or pressing a keyboard button) to when the task was successfully completed.

An error occurred if a participant selected **OK** in the DB condition or pressed **ENTER** in the GEKA or KDB conditions with an incorrect set of parameter values selected. If a participant made an error during a trial, a pop up notified the participant that an error was made and the task had to be repeated until it was successfully completed. After all trials for a command, participants took a 30 second break. Between conditions, participants took a 2 minute break. A questionnaire and interview completed the study.

Design

The experiment used a mixed factor design: 3 (interface) x 2 (command complexity) x 2 (command familiarity) x 2 (task length) x 5 (repetition) x 6 (presentation order). All factors were within-participant except for presentation order, which was a between-participant control variable. Bonferroni corrections were used for all pairwise comparisons. Greenhouse-Geisser corrections, identified by non-integer df, were used when sphericity was an issue.

Note on experimental design and participants

The focus of the KDB design and evaluation was on experienced command line users, but we were also interested in exploring how KDB would be received by users lacking command-line experience. Our initial experimental design thus included experience as a factor and we had 12 additional participants with no command line experience. We saw potential trends due to experience level, but there was very high variance in the data from inexperienced users. We would have needed to run many more participants to expose any significant differences that existed. We thus only report data from experienced users.

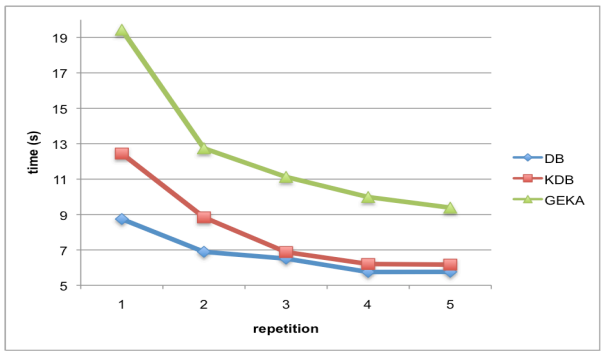

Figure 3: Mean completion times (N=12)

RESULTS

Most participants used both the keyboard and mouse (11/12), choosing the keyboard for different combinations of: inputting numbers (10/12), using a few mnemonics (4/12) and/or tabbing between adjacent parameters (7/12), and used the mouse for everything else.

Completion Time

A RM ANOVA indicated no main effect or interactions of presentation order, so it was dropped as a factor. Figure 3 shows that participants got faster over time: a main effect of repetition ($F(1.51, 16.58)=77.812$, $p<.000$, $\eta^2=.876$) with significant differences ($p<.05$) between all pairs of repetitions, except repetitions 4 and 5 ($p=.20$), indicating performance was plateauing. The RM ANOVA we report dropped repetition as a factor and used mean times for only repetitions 4 and 5 to eliminate the obvious learning affect.

Overall, KDB and DBs are not significantly different. There was a main effect of interface ($F(2, 22)=20.476$, $p<.001$, $\eta^2=.654$) with mean times for DBs, KDB, and GEKA being 5.76s, 6.19s, and 9.69s. Pairwise comparisons showed no significant difference between KDB and DBs ($p=1.0$). All other pairs had significant differences ($p<.05$).

For simple commands, DBs are fastest. The interfaces were impacted differently by command complexity (an interaction effect between interface and complexity, $F(2,22)=5.969$, $p=.008$, $\eta^2=.352$). For simple commands, a trend ($p=.072$) suggested DBs (4.09s) were faster than KDB (5.73s), but there was no difference for complex commands ($p=1.0$). GEKA was slower than DBs for both ($p<.003$).

For short tasks, KDB is fastest. An interaction between interface and task length ($F(2,22)=30.88$, $p=.000$, $\eta^2=.737$)

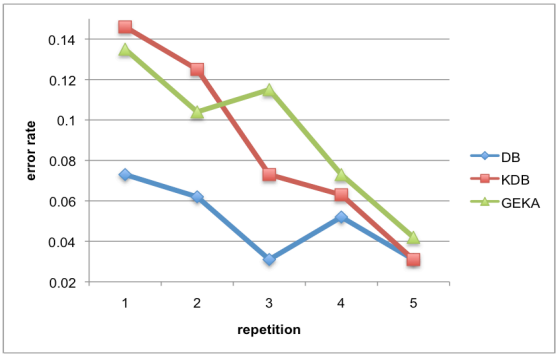

Figure 4: Error rates for interfaces in each repetition (N=12)

indicated that for short tasks, KDB (2.64s) was faster (p=.007) than DBs (3.74s). There was no difference between the two for long tasks (p=.355). GEKA was slower than KDB for both task lengths (p <.007).

There was no interaction between interface and command familiarity ($F(2,2)$=.867, p=.434, η^2=.073).

Errors

Figure 4 shows that participants made fewer errors as the study progressed. While there was a trend of interface ($F(2,22)$=3.01, p=.07, η^2=.215) across all repetitions, by repetition 5, errors had nearly disappeared for all three interfaces. In repetition 5, out of the 96 total trials for each interface DBs, KDB, and GEKA had 3, 3, and 4 errors, respectively, across all participants.

Questionnaire and interview

Overall, participants rated KDB the highest, with a mean of 17.00 on a scale of 0 ("I really dislike it") to 20 ("I really like it") vs. 10.50 for DBs and 12.25 for GEKA. A RM ANOVA showed no significant difference between DBs and GEKA (p=.392); all other pairs were significant (p<.004). When asked to explain why they preferred KDB, the most common reasons cited were being able to set parameter values without first having to specify the parameter name (7/12), liking the graphical feedback from the dialog box representation (6/12) (especially for verifying that the correct parameters were set (4/12)), and being able to only use the keyboard (5/12).

Participants said they were faster with KDB, with a mean of 18.17 on a scale of 0 ("very slow") to 20 ("very fast") vs. 11.75 for DBs and 13.25 for GEKA. There was no significant difference between DBs and GEKA (p=1.0); all other pairs were significant (p<.03). Participants explained they sometimes were able to skip specifying the parameter name (4/12) or that typing was faster than a mouse (3/12).

Participants said they made fewer errors with DBs, with a mean of 2.17 on a scale of 0 ("very few errors") to 20 ("many errors") vs. 6.58 for KDB and 9.08 for GEKA. The only significant difference was between DBs and GEKA (p=.002). While they did not necessarily like the mouse, they felt that using a mouse made errors less likely (5/12).

Ten of 12 participants said KDB was easier to learn than GEKA and gave reasons similar to why they had preferred KDB overall. However, DBs were felt to be the easiest to learn because participants were already familiar with them.

LIMITATIONS

Because we made improvements to both the command language (the changes only ever decrease the required keystrokes) and the graphical feedback, it is not clear how much of the improvement over GEKA can be attributed to each. Because the reduction in keystrokes is fairly small and the graphical feedback is radically different, we believe that most of the gains are a result of the graphical feedback.

While we made an effort to evaluate diverse tasks by using simple, complex, familiar, and unfamiliar commands with short and long tasks, our experiment still used only a small number of commands and tasks compared to what users experience in actual scenarios. Furthermore, while we saw performance appear to plateau by the end of our experiment, we cannot know how performance would change with prolonged usage. A longitudinal field study would help address these issues. Additional future work includes a study with more focus on non-experienced users and one that examines cognitive load.

DISCUSSION AND CONCLUSIONS

KDB showed speed and error rates in the final repetitions nearly identical to DBs. Participants reported a very strong preference for KDB. This suggests KDB should be an option for parameter specification: it makes experienced users more satisfied without impeding performance.

Despite the very similar speeds between dialog boxes and KDB, participants felt that they were much faster with KDB. We found a similar contradiction while evaluating GEKA [2]. This is an interesting finding about users' perceptions that bears further investigation.

KDB significantly outperforms GEKA in speed under almost all conditions. This suggests that the dialog box-like graphical feedback of KDB is a major improvement over the feedback in GEKA. Another major improvement over GEKA is that KDB can support a wide variety of parameter types. Earlier work [2] lists several types of commands that GEKA is not able to accommodate because of its restrictive graphical feedback. KDB overcomes most limitations.

We previously showed GEKA to be an effective general interaction method for specifying many types of commands, but it fell short compared to dialog boxes. We have now shown that KDB outperforms GEKA for parameter selection and is highly competitive with dialog boxes. Application designers looking to improve interfaces for experienced command line users could incorporate KDB-style parameter selection into any keyboard-based command selection scheme, including but not limited to GEKA. This would give users a well-liked way to select all commands and parameters using only a keyboard.

REFERENCES

[1] Blacktree (2009). Retrieved March 31. http://www.blacktree.com

[2] Hendy, J., Booth, K.S., McGrenere, J. (2010). Graphically Enhanced Keyboard Accelerators for GUIs. In *Proc. Graphics Interface 2010*, 3-10.

[3] Humanized (2009). Retrieved March 31. http://humanized.com

[4] Miller, R. C., Chou, V. H., Bernstein, M., Little, G., Van Kleek, M., Karger, D., and schraefel, m. (2008). Inky: a sloppy command line for the web with rich visual feedback. In *Proc. UIST 2008*, 131-140.

[5] Mozilla Labs. (2009). Ubiquity. Retrieved December 12. https://mozillalabs.com/ubiquity/