Towards Demystification of Direct Manipulation: Cognitive Modeling Charts the Gulf of Execution

David Kieras
Electrical Engineering and Computer Science Department
University of Michigan
Ann Arbor, MI 48109
kieras@eecs.umich.edu

David Meyer
Psychology Department
University of Michigan
Ann Arbor, MI 48109
demeyer@umich.edu

James Ballas
Naval Research Laboratory
Washington, D.C. 20375-5337
ballas@itd.nrl.navy.mil

ABSTRACT

Direct manipulation involves a large number of interacting psychological mechanisms that make the performance of a given interface hard to predict on intuitive or informal grounds. This paper applies cognitive modeling to explain the subtle effects produced by using a keypad versus a touchscreen in a performance-critical laboratory task.

Keywords

Direct manipulation, cognitive modeling

INTRODUCTION

Direct manipulation is an interface design concept in which the user manipulates or selects objects on a screen with a pointing device rather than types in syntactically structured commands. Although almost everybody believes that direct manipulation interfaces are superior, the psychological theory involved has not been systematically developed (see [4] for a recent review). Perhaps the most common presentation is in terms of Norman's [5] concepts of the "gulf of evaluation" (the user must interpret the display) and the "gulf of execution" (the user must determine how to act on the system). Direct manipulation interfaces are thought to reduce these two "gulfs", meaning that the user can more easily understand the system state revealed on the display and more easily figure out how to act on the system to achieve the desired result.

But from the psychological point of view, direct manipulation interfaces confound several factors. For example, they usually involve recognition rather than recall; visual search instead of verbal memory; concrete rather than abstract metaphors; simpler, more consistent, procedures (e.g. as revealed by GOMS analysis); and most interestingly, they stress different perceptual and motor capabilities, such as mouse movements instead of keyboard entry. Since we lack a well-articulated psychological theory of how the different interface paradigms operate at the detailed cognitive, perceptual, and motor levels, we do not know how these differences will interact in determining the quality of actual interfaces.

SIGCHI'01, March 31-April 4, 2001, Seattle, WA, USA.

This paper presents an initial theoretical treatment of the nature of direct manipulation, based on some data previously collected on this topic [1, 2], that uses a complex performance-critical task to compare a keyboard-operated interface with a touchscreen interface. Since the display is the same, these results address how the "gulf of execution" is affected by direct manipulation. Due to limitations on space, this is necessarily only a subset of a large and complex set of results; more complete treatments will be presented elsewhere.

THE TASK AND THE INTERFACES

The task was developed by Ballas, Heitmeyer, & Perez [1, 2] to resemble a class of multiple tasks performed in combat aircraft in which the user must both perform a task such as tracking a target, and at the same time keep up with the tactical situation using sensors such as radar, with partial automation support by an on-board computer. Figure 1 shows a sketch of the display as it appears for the Keypad interface. The right hand box contains a pursuit tracking task in which the cross-hairs must be kept on the target with a joystick operated with the right hand. The left-hand box is a radar-like display that contains a tactical decision task in which objects ("tracks") must be classified as hostile or neutral based on their behavior, and the results entered by means of a keypad under the left hand. These objects appear as icons that represent fighter aircraft, cargo

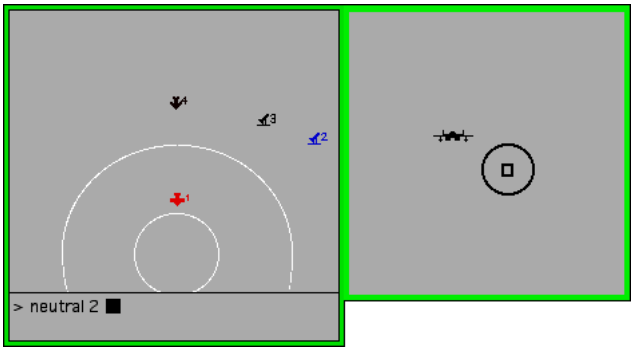

Fig. 1. Screenshot of the Keypad interface display.

airplanes, and SAM sites. A number identifies each object on the display. To avoid the overloaded term "object" or the military jargon of "track", the term *blip* will be used to refer to the objects on the radar display. Similarly, "user" will refer to the experimental participants.

The blips appear near the top of the display, and then move down. The fictitious on-board computer attempts to classify each blip, indicating the outcome after some time by changing the blip color from black to red, blue, or amber. If the blip changes to red (hostile) or blue (neutral), the user must simply confirm the computer's classification by typing a code key for the hostile/neutral designation followed by the key for the blip number. If the blip changes to amber, the subject must observe the behavior of the blip and classify it based on a set of rules, and then type the hostility designation and blip number. After the response, the blip changes color to white, and then disappears from the display some time later. The basic dependent variable is the reaction time to the *events*, measured from when a blip changes color to when each of the two keystrokes are made in response.

Ballas et al. varied the format of the tactical display and the response. The above description is for the *graphical keypad* interface; the other combinations consisted of using a tabular display instead of the graphical radar-like display, and a touchscreen response procedure instead of the keypad. This work concerns only the graphical display with both the keypad and touchscreen responses. In the Touchscreen interface, the first response is to identify the track by touching the blip on the screen, and the second response is to designate the hostility by touching a wide color-coded bar at the side of the tactical task display; a left-hand red bar corresponded to hostile and a right-hand blue bar to neutral. Notice that the order of the responses in logical terms was reversed for the Touchscreen interface compared to the Keypad interface.

Ballas et al. also studied the effects of *adaptive automation*. From time to time during the task, the tracking task would become difficult, and the on-board computer would take over the tactical task, signaling when it did so. The computer would then generate the correct responses to each blip at the appropriate time, with the color changes showing on the display as in the manual version of the task. Later, the tracking would become easy again, and the computer would signal with a loud buzzer tone and then return the tactical task to the user to perform. How users dealt with the transition was measured by recording the time required to respond to the individual events, counting from when they had to resume the tactical task.

THE PUZZLES IN THE DATA

Figure 2 shows these results in terms of the reaction time (RT) for the second response, which represents the total time to respond to an event, for the two interfaces. The

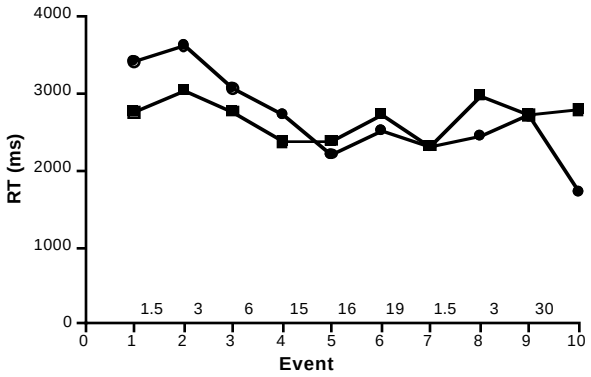

Fig. 2. Observed reaction times for Keypad (circles) and Touchscreen (squares) as function of event since task resumption. Inter-event intervals (sec) are shown above the x-axis. Note the rough equivalence of the two interfaces.

horizontal axis corresponds to each event following resumption of the manual tactical task; i.e., Event 1 is the first color-change event, Event 2 is the second, and so forth. Above the *x*-axis is shown the time interval between events in seconds. Events 1, 2, 3, and 4 were set to appear at closely spaced increasing fixed intervals, as are Events 7 and 8. The other events are widely spaced at randomly chosen intervals whose mean values are shown. Thus there is a high workload at the beginning of task resumption, a low-workload period, followed by another high-workload peak with the same event types and spacing, and a final low-workload period. In all of the graphs in this paper, the Keypad interface is represented with circular plotting points, the Touchscreen with square points.

The most puzzling result in Figure 2 is that the time to complete the processing for each event is roughly the same for the two interfaces despite the apparent "naturalness" or "directness" of the Touchscreen interface compared the Keypad. Even more puzzling is the fact that sometimes the Keypad interface is faster that the Touchscreen. In fact, the Touchscreen is superior to the Keypad only during first few events. As noted by Ballas et al. [1, 2], there is an *automation deficit* effect, in which for a time after resuming the tactical task, users produced longer response times in the tactical task compared to their normal steady-state manual performance. This effect represents some of the serious concerns about possible negative effects of automation in combat situations; if the automation fails, the operator can lack situation awareness, and it might take a long time to "catch up."

The Keypad interface shows a strong automation deficit effect, where the times for Events 1, 2, and 3 are longer than the matched Events 7, 8, and 9, producing an overall descending shape to the RT profile. In contrast, the Touchscreen interface is fairly flat, showing little or no automation deficit effect.

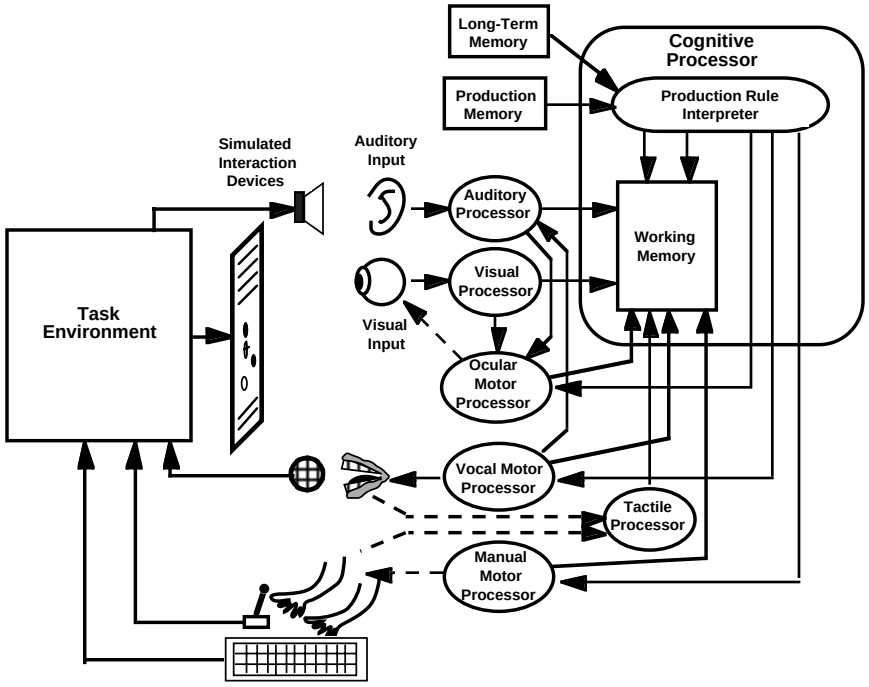

Fig. 3. The overall structure of the EPIC architecture. Perceptual-motor peripherals surround a cognitive processor.

While lesser automation deficit accords with the assumed superiority of direct manipulation, one would surely expect the direct-manipulation Touchscreen interface to be superior to the clumsy Keypad interface across the board. But the the advantage seems to be limited to the relief of automation deficit. These results are puzzling for a simple view of direct manipulation.

Explaining these phenomena requires much more detailed analysis than the usual generalizations about the virtues of direct manipulation. We would argue that in fact unaided intuition or conventional wisdom about fundamental interface design choices lacks enough precision to be useful to designers trying to arrive at an efficient interface for a demanding task. In the remainder of this paper, we will present computational models that explain these phenomena using a comprehensive architecture for human cognition and performance. These models explain in detail how the Touchscreen interface does indeed "narrow the gulf of execution" in a sense, but this has more to do with the different ordering of the two responses, rather than the "directness" of the interface. The Touchscreen interface also differs from the Keypad interface over how the eye can be moved, meaning that changes that affect the "gulf of execution" can also affect "the gulf of evaluation."

MODELING THE INTERFACE TASKS

The EPIC Cognitive Architecture

The models for the two interfaces were constructed using the EPIC architecture for human cognition and performance, which provides a general framework for simulating a human interacting with an environment to accomplish a task. Due to lack of space, EPIC cannot be described in full detail here. A more thorough description is presented in [6]. In brief, EPIC resembles the Model Human Processor [3], but differs in that EPIC is an implemented computational modeling system and incorporates more specific constraints synthesized from human performance literature. Figure 3 provides an overview of the architecture, showing perceptual and motor processor peripherals surrounding a cognitive processor; all of the processors run in parallel with each other. To model human performance of a task, the cognitive processor is programmed with production rules that implement a strategy for performing the task. When the simulation is run, the architecture generates the specific sequence of perceptual, cognitive, and motor events required to perform the task, within the constraints determined by the architecture and the interface. For example, the current orientation of the eye determines how visual events are detected and recognized, and movement times are governed by relationships such as Fitts' Law.

A few key features of EPIC can be summarized; additional ones will be introduced as needed. The perceptual and cognitive processors are always operating in parallel, but it is up to the task strategy (the production rule programming) to take advantage of the parallel capabilities of the motor processors. Cognitive processing is multi-threaded, in that multiple sequences of production-rule execution can be underway simultaneously, which enables sophisticated models of complex multiple-task situations [9, 11].

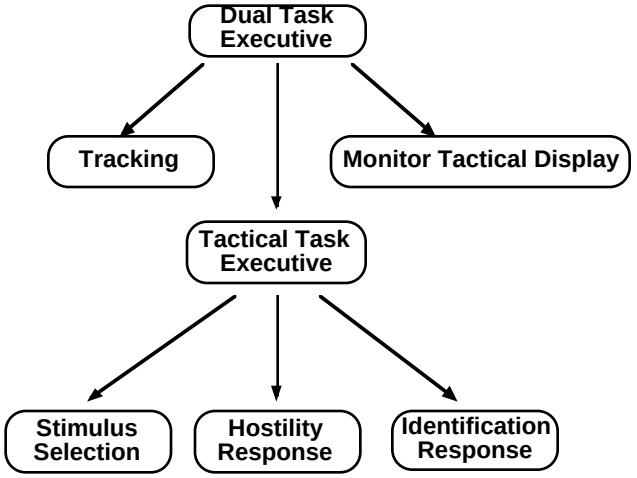

Fig. 4. The structure of the production-rule task strategy for both interfaces.

Task Strategy

To model the present task, a set of production rules were written for each interface, organized in a hierarchy shown in Figure 4 reflecting the overall structure of the task. Since EPIC allows fully multithreaded cognitive processing, many of the tasks shown in the figure execute in parallel. The top level is a dual-task executive process that controls execution of the tracking task, the tactical task, and a task that monitors the tactical display for color changes and other relevant events. When the tactical task is automated, the dual-task executive runs only the tracking task, effectively ignoring the tactical display entirely. When the auditory signal to resume the tactical task is recognized, the executive shuts down the tracking task, starts the monitoring process and initiates the tactical task process to handle the first event. Notice that the tactical task is controlled by its own sub-executive process. When the tactical task no longer has events to process, it terminates, and the dual-task executive restarts the tracking task. If the monitoring task then detects events such as a color change in peripheral vision, the dual-task executive will shut down the tracking task and restart the tactical task to handle the event. When the signal is made to return to automated mode, the executive shuts down everything except the tracking task, and waits for the next resumption signal.

Tactical task executive

The tactical task executive controls three subprocesses. For the Keypad interface, stimulus selection first selects a "blip" on the screen to be processed, then the hostility response process selects and executes the hostility-designating response, and then the identification response process selects and executes the response that identifies the track. For the Touchscreen interface, the identification process executes second, followed by the hostility response process. Once both responses have been made, the tactical executive can repeat the overall process starting with selecting a new stimulus. If there are no more blips to process, the tactical task terminates.

Stimulus selection process

The basic signal to process an event is that a blip changes color from black to red, blue, or amber. Stimulus selection chooses a blip to process, taking into account that if tracking is underway, the eye will be kept on the tracking target, so most of the tactical display is in peripheral vision. While the color change can be noticed in peripheral vision, the color itself is not available except parafoveally. But once the eye is on the tactical display, the colors for most of the blips will be available. Thus stimulus selection may have to examine more than one blip before finding one ready to process. A blip is chosen for examination or processing according to a priority scheme that favors colored blips first, color-changed blips second, and unknown blips third. The eye is then moved to the chosen blip; if the blip is black or white, it cannot be processed, so another blip is chosen. If it is red, blue, or amber, it is marked as the selected stimulus and handed on to the response processes.

In addition, a color-change event can be anticipated: if a blip is close enough to the bottom of the display, it can be chosen for watching until it or another blip changes color.

Either way, once stimulus selection places the eye on the blip, the visual processing necessary to recognize the behavior of the blip is underway while other processing is going on. The time required to recognize the behavior is on the order of a second, and was independently estimated for each type of blip.

Keypad response processes

The first response is to identify the hostility of the selected blip. If the blip is red or blue, the proper hostility response is simply the corresponding keystroke; this is selected and executed by a sequence of production rules like those in other EPIC models of choice reaction time [9, 10]. However, if the blip is amber, the process must wait, with the eye tracking the blip, until the behavior of the blip has been visually recognized. The identification response process then moves the eye to the blip number and waits until it has been recognized. The corresponding keypad response is then selected with a series of choice reaction rules and executed.

Touchscreen response processes

The first response is to identify the track; although the track number is present on the display, it is irrelevant. An aimed finger movement is made to "poke" the blip on the touchscreen; note that the eye is already on the blip, and must remain there to guide the aimed movement.

The second response is to designate the hostility. If the blip is red or blue, the corresponding response bar target can be quickly selected. If it is amber, it must wait for the behavior to be recognized. A poke movement is then made to the

selected response bar on the touchscreen; this movement also requires visual guidance. The eye is commanded at the same time to move to the response bar; it will arrive quickly enough to guide the finger movement.

Overlapping Tactical Task Subprocesses

These three tactical subprocesses can run in parallel, or *overlap*, to a considerable extent, as long as the task constraints are observed. The two adjacent responses must be made for a single track, and the responses must be made in the correct order. Thus the second response process must wait for the first response to be produced, but the duration of the wait need only be long enough to produce the correct ordering. Thus some of the second response processing could be concurrent with the processing and production of the first response. In addition, the stimulus selection process for the next stimulus could overlap with the second response process.

The models were built in such a way that the tactical task executive could control the amount of overlap in the tactical task subprocesses. The issues in such overlapping have a strong family resemblance to how computer Operating Systems are organized to allow prioritized concurrent task execution and manage the allocation of peripheral resources to different tasks so as to maximize system throughput [8].

These models have both constraints and opportunities for overlapping. The Keypad model assumes that the user gets enough practice so that it is not necessary to look at the keys, opening the way for considerable overlapping since the eye can be moved while responses are being selected and executed. But the Touchscreen interface involves aimed finger movements which require that the eye be on the target of the movement, effectively locking the eye to the hand until responding is complete. On the other hand, visual recognition of the blip behavior will be proceeding during identification responding, whereas the Keypad interface forces a separation of hostility recognition and identification responding.

The Bracketing Heuristic

The amount of overlapping enforced by the executive strategy has a major impact on task performance [11]; "daring" strategies that maximize overlapping can produce substantially faster performance than more "conservative" strategies. This aspect of task strategy is not strongly determined by either the task requirements or the architecture, but rather is a result of factors such as the amount of practice, level of motivation, or long-term fatigue avoidance, or even the user's possibly haphazard efforts to formulate a task strategy [7]. Such *optional* aspects of task strategy can be identified only by careful iterative construction of models that match the observed data in fine enough detail.

Here we present a simpler, more robust, analysis based on the *bracketing heuristic* [7]. The concept is to start with a basic strategy for the task, such as the one just outlined, and permute it into two versions: a *fastest-possible* strategy that drives the architecture at the highest speed possible that still meets the basic task requirements, and a *slowest-reasonable* model that conforms to the task instructions and requirements with no "bells and whistles" to increase speed. Observed performance should fall somewhere between these two models, which thus *bracket* the actual performance.

Bracketing is a way to construct truly predictive models in complex task domains where the optional strategy optimizations users would device cannot be forecast. In addition, bracketing could guide the construction of models that match the data. However, bracketing can also be used to explain phenomena independently of the optional aspects of task strategies. Thus, bracketing models for the Keypad and Touchscreen interfaces may be able to account for the important differences between the interfaces in a simple and straightforward way without the elaborate detail of models that match the data.

The Bracketing Models

The fastest-possible model for this task assumes that the tactical task has highest priority, so it anticipates all color-change events; in fact, it ignores the tracking task as long as there is any unprocessed blip on the display. It uses the maximum overlapping possible, and allows the eye to be used by the next task process as soon as the architecture permits. In the Touchscreen interface, the fast model also anticipates the first response by prepositioning the finger at a candidate blip during stimulus selection.

The slowest-reasonable model adheres to the nominal experimental instructions for the task and stolidly performs the tracking task until a color-change event is detected. It returns to tracking when there are no colored blips to process. It never anticipates events. Overall, each process in the slowest model must wait until an ongoing motor movement is complete; no advantage is taken of the ability of the EPIC architecture to overlap motor movements. Thus the only overlapping is that provided by perceptual processing, which always runs in parallel with other processing.

Bracketing the Keypad Interface Performance

Figure 5 shows the predicted RTs from the fast and slow models for the Keypad interface along with the observed times. Throughout this paper, observed times are shown as solid plotting points and lines, and predicted times with open points and dotted lines. Note first that the predicted times indeed bracket the observed times. Both fast and slow models show an automation deficit effect in which the first few events take longer than the matching events during the second high-workload period.

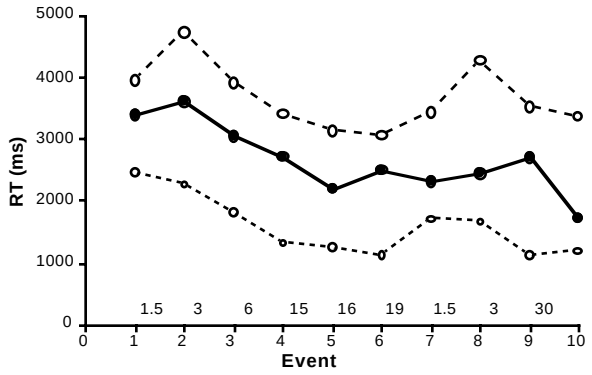

Fig. 5. Bracketing models for the Keypad Interface. Predicted RTs are shown as open points; observed as solid. Fastest-possible model predictions are the lower curve; slowest-reasonable is the upper curve.

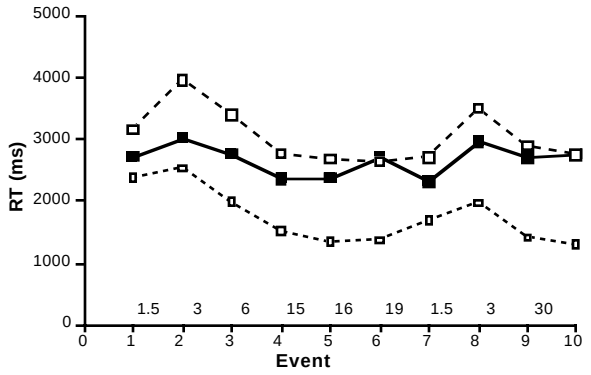

Fig. 6. Bracketing models for the Touchscreen Interface. Predicted RTs are shown as open points; observed as solid. Fastest-possible model predictions are the lower curve; slowest-reasonable is the upper curve.

The slow model has an exaggerated form of the effect in which the second of a pair of closely spaced events has an elevated RT. The mechanisms in the model that produce this result are similar to those that produce the psychological refractory period (PRP) effect, a laboratory phenomenon obtained when two simple choice reaction tasks are overlapped in time. Basically, because the two events are closely spaced in time, the second event processing must wait for some part of the processing of the first event to be complete. If enough time intervenes between the events, the second response is not delayed because the first response processing will be complete. Extensive previous modeling work with EPIC [9, 10] shows that the PRP effect is due either to conservative unnecessarily sequential task strategies, or to peripheral processing bottlenecks, such as the need to move the eye from the first to the second stimulus, or to responses having to queue up for control of the same motor processor. The fast model has less of this PRP-like effect because it overlaps processing very heavily. But milder forms of this effect is why the relatively closely spaced Events 1-4 and 7 and 8 have longer fast model RTs than events 5, 6, and 10.

Bracketing the Touchscreen Interface Performance

Figure 6 shows the predicted and observed RTs for the Touchscreen interface. A point to notice first is that the slow model fails to bracket the observed data on a couple of the events, even though it was constructed using the same guidelines defining slowest-reasonable as the slow Keypad model. Obviously a model can be made arbitrarily slow, but any attempt to make the slow model even slower just to completely bracket the data contradicts the *a-priori* stance of the bracketing logic. Rather, these isolated failures of bracketing appear on two events that come after long periods of tactical task inactivity, which suggests that subjects may have altered their strategy in some way during these inactive times.

Except for these two unusually slow event RTs, the fast and slow models basically parallel the observed times, but the slow model has more exaggerated peaks and valleys for the PRP-like effect. The observed values are quite close to the slow model during the low-workload periods, and somewhat closer to the fast model during the high-workload periods. In both models the automation deficit effect is small.

COMPARING THE INTERFACES VIA THE MODELS

In this section, we compare the two interfaces by comparing the performance of the fast models for the two interfaces, and the slow models for the two interfaces. This comparison will clarify what role the architecture plays versus the task strategy in determining the effects of the interface. The key idea is that the fastest-possible models mainly depend on the how the interface relates to the architecture, while the slowest-reasonable models magnify how the basic structure of the task depends on the interface.

Figure 7 shows the predicted RTs from the fast and slow models for the two interfaces. For clarity, the observed values are not shown.

Why the Two Interfaces are Almost Equally Fast

The fastest-possible models predict essentially identical times for the two interfaces, so from the viewpoint of architectural constraints, in fact, neither interface is superior overall to the other. The fastest model times depend mostly on just the perceptual-motor delays inherent in the architecture and the requirements of the interface. Generally, poking movements would be slower than keystrokes, but the basic reason for the identical fast model times can be seen by comparing the times for the first response and the time between the two responses (inter-response interval); the graphs are not shown because of space limitations. Compared to the Keypad first response, the Touchscreen first response is faster across the board, because it consists of an immediate poke at the looked-at blip rather than waiting for recognition of hostility and

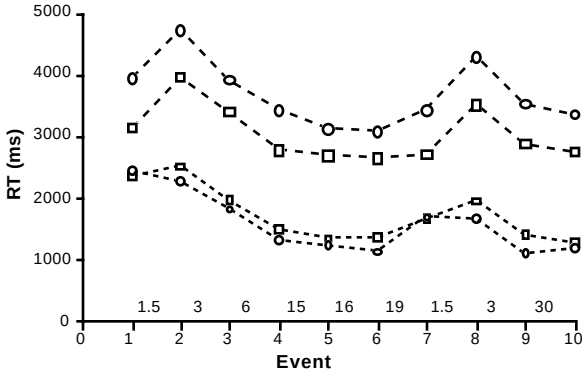

Fig. 7. Predicted RTs for fastest-possible (small points) and slowest-reasonable (large points) models for the Keypad (circles) and Touchscreen (squares) interfaces.

response selection. But the situation is reversed with the inter-response interval: the Touchscreen interface always takes about 500 ms longer to make the second response than the Keypad interface. The Keypad interface requires recognizing and responding to the blip number with a fast keystroke, rather than waiting on the lengthy hostility recognition and then making a relatively slow poking movement. As it happens, the sum of the first response times and oppositely-different inter-response intervals are almost identical for the two models, leading to very similar fast model total times. This pattern appears in both the models and the observed data.

In contrast, as shown in Figure 7, the slowest-reasonable models give a consistent advantage to the Touchscreen interface. If the user waits for each subtask to complete before going on the next subtask, then certain processes appear on the critical path in the Keypad interface that are not present in the Touchscreen interface. In the Keypad interface, the hostility perceptual processing is fully on the critical path (nothing else can be done until it is recognized), and the identification process cannot be overlapped with it. But as already outlined, in the Touchscreen interface the perceptual processing for the hostility response can done at the same time as the identification response processing. Thus, even if the two response processes are serialized, the first response can be overlapped with the visual processing required for the second. The result is a net advantage for the Touchscreen interface, even though touchscreen poking movements can be quite slow compared to keystrokes. Clearly if the response ordering in the Touchscreen interface was reversed, to be the same as the Keypad interface, the results would be quite different.

Thus the rough equivalence of the two interfaces is a result of subjects working somewhere between the fastest and slowest strategies, and because they might vary this setting during the task, which interface is superior in the data may appear inconsistent. The fact that the observed times tend to be closer to the fastest model during high workload, and to the slowest model during low workload, suggests that subjects do indeed modulate their task strategy as a function of workload during the task.

Why the Touchscreen has Less Automation Deficit

As was shown in Figure 2, the Touchscreen interface enjoys less of an automation deficit effect than the Keypad interface. In fact, the main situation in which the Touchscreen interface is faster than the Keypad is in the automation-deficit period of the first few events. These models explain the automation deficit in terms of the PRP-like delay effects summarized earlier. When the overall task goes from automated to manual mode, and the tactical task is started at a peak in workload, the first few events suffer additional delays that the later, steady-state events do not. The selection and processing of the first event is delayed, leading to a delay in processing the second event, and on through the others, until the increasing inter-event interval allows the models to catch up.

In contrast, in the steady state, as soon as a color-change happens, it is noted by the monitoring process and the tactical task is started immediately, resulting in a shorter response time. Even when the events are closely spaced as in the second workload peak, the delays in processing the first selected blip are not as large, and so the second blip does not suffer as much from PRP-like effects.

Why does the Touchscreen interface produce a smaller automation deficit than the Keypad interface? The answer is deceptively simple. If the user works somewhere between the fastest and slowest model strategies, the time to complete processing in the Keypad interface is longer than in the Touchscreen interface (for reasons already explained), so the start-up delays that both interfaces have in common will have larger PRP-like effects in the Keypad interface than in the Touchscreen interface. If the inter-event intervals were reduced somewhat, the Touchscreen interface would show a higher level of automation deficit, but of course, the Keypad interface would suffer even more. The exact extent of the difference in automation deficit depends on the extent to which the users follow strategies more like the fastest-possible rather than the slowest-reasonable, and of course they could change their strategies dynamically during task execution.

Thus the reduced automation deficit for the Touchscreen interface is simply a result of how this particular interface capability was applied to this task domain. A rearrangement of either interface task might make a substantial difference in the relative performance.

CONCLUSION

The Touchscreen interface does indeed have a narrower "gulf of execution" in that it is generally better than the Keypad interface in several important ways, but rather than a vague and mysterious effect, this advantage is a straightforward result of simpler response selection processing and more opportunities to overlap perceptual and motor processing. But the greater constraints on eye movements in the Touchscreen interface shows that the "gulf of evaluation" can depend on factors normally associated with the "gulf of execution."

The models show also that the Touchscreen had certain disadvantages that a different interface design might mitigate, such as the need to reserve the eye to guide the second response, and the fact that both responses were bottlenecked through relatively slow hand movements. For example, if the hostility response could be delivered via a speech-driven interface, the bottleneck produced by the need to make a second visually-guided hand motion would be eliminated, and both the eye and hand would be free sooner to seek the next blip to process.

The complex mix of effects appearing in this interface comparison is a result of the subtle interplay between interface design, task strategies, and the fundamentals of human perceptual and motor capabilities and limitations. The fact that the details of the interaction are critical is reminiscent of the point made some years ago by Whiteside et al. [12] in their seminal comparison of different interface styles: The style of the interface is not in itself critical; rather the key is getting all the details right with a well-crafted interface. Thus the specific ways in which the details of user processing play out in this interface comparison suggests that broad-brush generalizations of interface principles can not be relied upon to give designers reliable guidance that will help them to arrive at superior interfaces more quickly than brute-force user testing iterations. Rather, continued work with detailed computational model of human-computer interaction should help us develop a full scientific understanding of interfaces, which can then be delivered in modeling tools that are accessible to designers.

ACKNOWLEDGMENT

This work was supported by the Office of Naval Research.

REFERENCES

1. Ballas, J. A., Heitmeyer, C. L., & Perez, M. A. (1992). *Direct manipulation and intermittent automation in advanced cockpits*. Technical Report NRL/FR/5534--92-9375. Naval Research Laboratory, Washington, D. C.

2. Ballas, J. A., Heitmeyer, C. L., & Perez, M. A. (1992). *Evaluating two aspects of direct manipulation in advanced cockpits.* In Bauersfeld, P., Bennett, J., and Lynch, G., *CHI'92 Conference Proceedings: ACM Conference on Human Factors in Computing Systems*, Monterey, May 3-7, 1992.

3. Card, S. K., Moran, T. P., & Newell, A. (1983). *The psychology of human-computer interaction.* Hillsdale, NJ: Lawrence Erlbaum Associates.

4. Frohlich, D. M. Direct manipulation and other lessons. In M. Helander, T.K. Landauer, & P. Prabhu (Eds.). *Handbook of Human-Computer Interaction* (2nd ed.), Elsevier, 1997.

5. Hutchins, E. L., Hollan, J. D., & Norman, D. A. (1986). Direct manipulation interfaces. In D.A. Norman, & Draper, S. W. (Eds.), *User centered system design*. Hillsdale, NJ: Lawrence Erlbaum Associates.

6. Kieras, D. & Meyer, D.E. (1997). An overview of the EPIC architecture for cognition and performance with application to human-computer interaction. *Human-Computer Interaction*., **12**, 391-438.

7. Kieras, D. E., & Meyer, D. E. (2000). The role of cognitive task analysis in the application of predictive models of human performance. In J. M. C. Schraagen, S. E. Chipman, & V. L. Shalin (Eds.), *Cognitive task analysis.* Mahwah, NJ: Lawrence Erlbaum, 2000.

8. Kieras, D.E., Meyer, D.E., Ballas, J.A., Lauber, E.J. (in press) Modern computational perspectives on executive mental processes and cognitive control. Where to from here? In S. Monsell and J. Driver (Eds.), *Control of cognitive processes: Attention and Performance XVIII*. Cambridge, MA: MIT Press.

9. Meyer, D. E., & Kieras, D. E. (1997). A computational theory of executive cognitive processes and multiple-task performance: Part 1. Basic mechanisms. *Psychological Review*, **104**, 3-65.

10. Meyer, D. E., & Kieras, D. E. (1997). A computational theory of executive control processes and human multiple-task performance: Part 2. Accounts of Psychological Refractory-Period Phenomena. *Psychological Review*. **104**, 749-791.

11. Meyer, D. E., & Kieras, D. E. (1999). Precis to a practical unified theory of cognition and action: Some lessons from computational modeling of human multiple-task performance. In D. Gopher & A. Koriat (Eds.), *Attention and Performance XVII.*(pp. 15-88) Cambridge, MA: M.I.T. Press.

12. Whiteside, J., Jones, S., Levy, P. S., & Wixon, D. (1985). User performance with command, menu, and iconic interfaces. In *Proceedings of CHI '85.* New York: ACM.