Modern Computational Perspectives on Executive Mental Processes and Cognitive Control: Where to from Here?

David E. Kieras, David E. Meyer
University of Michigan

James A. Ballas
Naval Research Laboratory

Erick J. Lauber
University of Georgia

EPIC Report No. 12 (TR-99/ONR-EPIC-12)

August 1, 1999

This research was supported by the U.S. Office of Naval Research, Cognitive Science Program, under Grant Number N00014-92-J-1173, Grant Authority Identification Number NR 4422574. Reproduction in whole or part is permitted for any purpose of the United States Government.
Requests for reprints should be addressed to: David E. Kieras (kieras@eecs.umich.edu), Artificial Intelligence Laboratory, Electrical Engineering and Computer Science Department, Advanced Technology Laboratory Building, University of Michigan, 1101 Beal Avenue, Ann Arbor, MI 48109-2110, USA; David E. Meyer (demeyer@umich.edu), Cognition and Perception Program, Dept. of Psychology, University of Michigan, 525 East University, Ann Arbor, MI, 48109-1109, USA.

Modern Computational Perspectives on Executive Mental Processes and Cognitive Control: Where To From Here?*

David E. Kieras, David E. Meyer
University of Michigan

James A. Ballas
Naval Research Laboratory

Erick J. Lauber
University of Georgia

Abstract

Future research on cognitive control must precisely characterize the supervisory functions of executive mental processes. The achievement of this objective will be facilitated by formal concepts and algorithms from contemporary computer operating systems. In particular, operating-system fundamentals can help to advance work with the Executive-Process Interactive Control (EPIC) architecture, a theoretical framework for computational modeling of human multiple-task performance. EPIC models that incorporate general executive processes like those of operating systems provide insights about how people schedule tasks, allocate perceptual-motor resources, and coordinate task processes during multiple-task performance under both laboratory and real-world conditions. Such insights may lead to discoveries about the acquisition of procedural task knowledge and efficient multitasking skills.

---

* This document is a preprint of a chapter to be published in S. Monsell and J. Driver (Eds.), *Control of Cognitive Processes: Attention and Performance XVIII*, Cambridge, MA, M.I.T. Press. The subsequent text cites companion chapters by other authors as appearing in "this volume". Support for the research reported herein was provided by grant N00014-92-J-1173 from the Cognitive Sciences Program of the Office of Naval Research to the University of Michigan. We thank members of the Brain, Cognition, and Action Laboratory (David Fencsik, Darren Gergle, Jennifer Glass, Leon Gmeindl, Cerita Jones, Shane Mueller, Eric Schumacher, Mollie Schweppe, and Travis Seymour) for helpful assistance. Comments by Leon Gmeindl, Stephen Monsell, Travis Seymour, and two anonymous reviewers on drafts of this chapter are greatly appreciated. Correspondence about this chapter should be addressed to: David E. Kieras (kieras@eecs.umich.edu), Artificial Intelligence Laboratory, Electrical Engineering and Computer Science Department, Advanced Technology Laboratory Building, University of Michigan, 1101 Beal Avenue, Ann Arbor, MI 48109-2110, USA; David E. Meyer (demeyer@umich.edu), Department of Psychology, University of Michigan, 525 E. University, Ann Arbor, Michigan, MI 48109-1109, USA.

Introduction

Following the cognitive revolution in scientific psychology (circa 1950), many experimental psychologists and cognitive scientists have assumed that human cognition shares fundamental similarities with symbolic information processing by electronic digital computers (Lachman, Lachman, & Butterfield, 1979; Newell, 1990). Although their operations are serial in some respects, such computers can emulate parallel processing of multiple information streams and implement algorithms for modeling the performance of perceptual-motor and cognitive tasks. As a result, the computer metaphor has yielded significant discoveries about perception, attention, learning, memory, language, and problem solving. Furthermore, as computational hardware and software continue to evolve, the computer metaphor may become increasingly apt.

Encouraged by this prospect, our work has focused on characterizing executive mental processes with a particular theoretical framework, the *Executive-Process Interactive Control* (EPIC) architecture. Using EPIC, we have formulated precise computational models of human multiple-task performance under both laboratory and real-world conditions (e.g., Kieras & Meyer, 1997, 1999; Meyer & Kieras, 1997a, 1997b, 1999). EPIC models account well for quantitative data, predict new phenomena, and point toward promising directions for future research on cognitive control.

The functions of executive processes in EPIC correspond closely to ones provided by a computer operating system (OS) that supports parallel information processing for concurrent execution of multiple task programs (Stallings, 1998). This correspondence suggests that studying the fundamentals of contemporary OSs may facilitate the development of EPIC. Such study may also advance the conceptualization of executive mental processes in other theoretical frameworks (e.g., Baddeley, 1986; Braver & Cohen, this volume; Kimberg & Farah, this volume; Norman & Shallice, 1986), thereby helping to banish the "homunculus" of cognitive control about which previous pundits have complained vociferously (e.g., Newell, 1980; Neisser, 1967).

In our opinion, the modern computer metaphor is relevant to answering several related questions: Do people have general executive processes that are used across many contexts? Exactly what functions do these processes serve? How might they influence the representation and acquisition of procedural task knowledge? Are there task-specific aspects of cognitive control for which general executive processes must be supplemented through special training? Which experimental procedures are especially suited for eliciting and analyzing particular control operations? Does the human brain really implement the types of function that an OS provides?

Toward answering these questions, this chapter has five subsequent main sections. First, we introduce EPIC. Second, we describe results from applications of EPIC to modeling multiple-task performance and characterizing particular executive mental processes. Third, we present additional relevant concepts from contemporary computer technology and OSs. Fourth, we discuss how these concepts may promote research with EPIC and guide theorizing about cognitive control. Fifth, we summarize our conclusions and offer final thoughts about future research directions.

The EPIC Architecture

Extending proposals by previous theorists (e.g., Anderson, 1983; Card, Moran, & Newell, 1983; Newell, 1990), we have designed EPIC to integrate cognitive and perceptual-motor operations with procedural task analyses of skilled performance.

Architectural Components

EPIC has a central cognitive processor with a production-rule interpreter and a multi-partition working memory (WM) surrounded by peripheral sensors, perceptual processors, motor processors, and effectors that all operate in parallel (Figure 1). These permanent interconnected components constitute EPIC's "hardware". Each perceptual and motor processor functions as a distinct limited-capacity channel of input or output. Task performance is modeled by programming the cognitive processor with production rules that make decisions and generate responses based on the contents of

WM. The production rules, stimulus codes, and response codes may vary depending on specific task requirements.

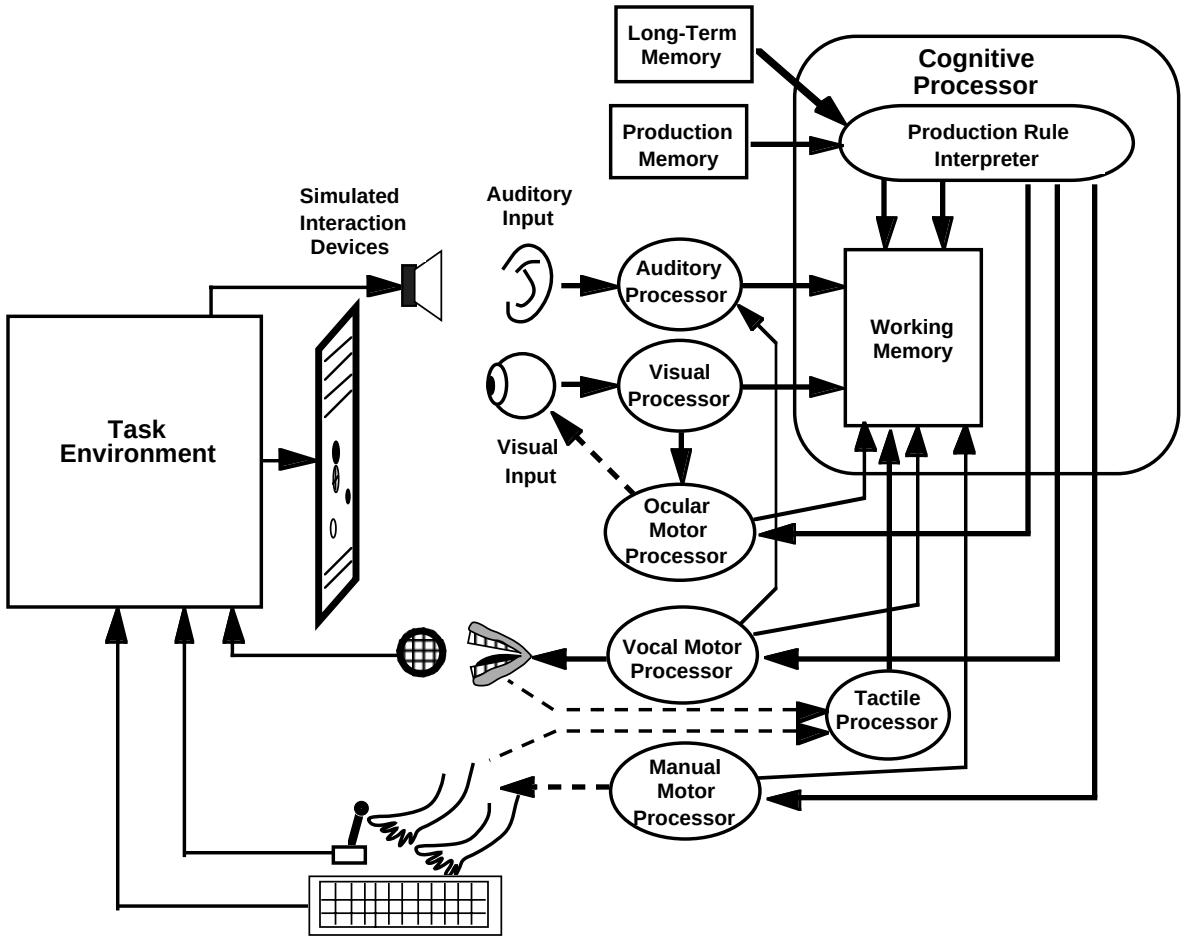

Figure 1. Overview of the EPIC architecture (adapted from Meyer & Kieras, 1999).

Consistent with basic periodicities of human information processing (Kristofferson, 1967), EPIC's cognitive processor operates in cycles that have stochastic durations whose mean is 50 ms. While doing so, the cognitive processor enables a high degree of parallelism in multiple-task performance. On each cycle, its production-rule interpreter tests the conditions of all rules in procedural memory, and executes the actions of all rules whose conditions match the current contents of WM. There is no set limit on how many rules can be applied simultaneously. Thus, cognitive processes involving distinct sequences of rules may progress simultaneously, sharing system resources as time passes.

Basics of Control

The flow of information processing in EPIC is controlled with production rules like the following one, which selects and initiates a manual "poke" response to a red target stimulus during a tactical decision task (Kieras & Meyer, 1997, 1999; Meyer & Kieras, 1999):

```
IF
((GOAL DESIGNATE-TARGET-FOR-TACTICAL-TASK)
 (STRATEGY MAKE-POKE-RESPONSE-IMMEDIATELY)
 (STEP MAKE-POKE-RESPONSE)
 (TAG ?OBJECT IS STIMULUS)
 (VISUAL ?OBJECT COLOR RED)
 (NOT (VISUAL ??? SIZE LARGE))
 (STATUS TACTICAL-TASK-PROCESS-HAS-EYE)
 (MOTOR MANUAL PROCESSOR FREE))
THEN
((SEND-TO-MOTOR-MANUAL-PROCESSOR PERFORM-POKE-(LEFT INDEX) ?OBJECT)
 (ADDWM (GOAL WATCH-FOR-DESIGNATION-EFFECT))
 (DELWM (STEP MAKE-POKE-RESPONSE))
 (ADDWM (STEP WAIT-FOR-WATCHING-DONE)))
```

Sequential rule execution. As illustrated here, EPIC production rules have conditions and actions that contain goal and step items. Adding and deleting step items in WM enables the rules to be executed in particular sequences. For example, the preceding rule would be enabled by putting STEP MAKE-POKE-RESPONSE in WM with an add-to-wm (ADDWM) action. Taking this item out of WM with a delete-from-wm (DELWM) action would disable the rule, and then putting STEP WAIT-FOR-WATCHING-DONE in WM would enable another subsequent rule. Because information in WM is subject to loss or corruption, errors of sequencing may occur under EPIC, as they do under real-world circumstances.

Subroutine calls. Using the same goal item in a set of EPIC production rules lets them function like a computer-program subroutine. The subroutine would be "called" by adding its shared goal item to WM. After the call, a start-up rule in the subroutine would "fire" and add its first step item to WM. When the subroutine finishes, its termination rule would delete the routine's goal and last step items from WM, and signal that the subroutine has finished. For example, the preceding rule calls a subroutine for watching the visual effects of the manual poke response. This entails adding two items to WM: GOAL WATCH-FOR-DESIGNATION-EFFECT, which is the goal item for the subroutine; and STEP WAIT-FOR-WATCHING-DONE, which is used by another rule that waits for the subroutine to be completed.

Interrupts. In this way, EPIC implements capabilities analogous to computer interrupts. A production rule can have conditions such that it waits for a certain future event to occur regardless of other intervening activities. When these conditions are satisfied, the rule may start the execution of other rule sequences to deal with the interrupting event.

Task Processes

Procedural knowledge for performing tasks is represented by EPIC production rules that fire in particular sequences. Our models embody programming-style principles like those applied in computer software design. For each task and subtask, there is a set of rules that have standard formatting of control items and I/O information. Standard protocols are used for task startup, completion, error detection, abort, and restart procedures.

Executive Processes

In modeling multiple-task performance, we formulate distinct sets of supervisory production rules that implement supraordinate executive processes. Their function is to add and delete working-memory items for controlling the execution of various task and subtask procedures. Under EPIC, an executive process may suspend a task process by deleting its goal item from WM, and then resume the task process by adding its goal item to WM again. Similarly, an executive process may use strategy items to instruct a task process about which of several alternative paths to take. These control operations can be accomplished through rules whose conditions match status items that the task process adds to WM along the way.

Applications of EPIC to Multiple-Task Performance

To illustrate more fully how we characterize executive mental processes, this section describes four cases in which EPIC models of multiple-task performance have been developed. These include (1) discrete successive tasks, (2) discrete concurrent tasks, (3) elementary continuous tasks, and (4) compound continuous tasks. From them, it will become clearer how EPIC enables task coordination and scheduling to be described under a variety of conditions. Also, the stage will be set for examining cognitive control from the perspective of computer operating systems.

Discrete Successive Tasks

In the discrete successive-tasks procedure, also known as *task switching*, participants either alternate between two different choice-reaction tasks or perform one task repeatedly during a series of discrete trials, with a *response-stimulus interval* (RSI) separating each response from onset of the next stimulus. Reaction time (RT) and accuracy are measured as a function of trial type, RSI, and other factors. Switching-time costs (STCs) are calculated from differences between mean RTs on alternating-task and repeating-task trials (for a review, see Pashler, this volume).

According to some theorists, executive mental processes contribute substantially to STCs (e.g., Meiran, 1996; Rogers & Monsell, 1995; Rubinstein, Meyer, & Evans, 1995; in this volume, see De Jong; Goschke; Keele & Rafal; Meiran). Following their lead, we have formulated an EPIC model to account for some results from the successive-tasks procedure. The details of this formulation concern both the representation of procedural task knowledge and the cognitive control of task switching.

Lauber's study. For now, our model deals with data from a study by Lauber (1995, Exps. 4 and 5). This study is especially interesting because it involved varying RSI, stimulus-response compatibility, and practice orthogonally. Additive and interactive effects of these factors strongly constrain the type of model that may account for them.

Twenty undergraduate students participated in Lauber's study. They were divided into two groups that performed basic choice-reaction tasks with different S-R mappings. Members of each group were tested individually during three 1-hr sessions. The stimuli for each task were visually presented digits. The responses were key presses made with fingers of the right hand. Stimuli and responses were paired to form four alternative S-R mappings, each of which was used in one of four different tasks. The tasks included *compatible Task A*, *compatible Task B*, *incompatible Task C*, and *incompatible Task D*. For Task A, the digits 1, 2, 3, and 4 were mapped respectively to the index, middle, ring, and little fingers; for Task B, this mapping was reversed. For Task C, the digits 1, 2, 3, and 4 were mapped respectively to the middle, little, index, and ring fingers; for Task D, this mapping was reversed.

During each test session, there were two types of trial block. One block type contained a series of alternating-task trials, and the other contained a series of repeating-task trials. On each alternating-task trial, participants in Group 1 performed Task A followed by Task B, or vice versa; during each repeating-task trial, they performed one of these tasks twice. A similar arrangement of Tasks C and D was used for Group 2. Before each trial block, participants were told what their tasks would be. Each block included two RSIs, 50 ms and 750 ms, which varied randomly across trials. The intertrial intervals equaled 1 s.

Empirical results. Figure 2 shows some results from the first session. Mean RTs of second-task (post-RSI) responses were reliably longer for alternating-task trials, incompatible S-R mappings, and short RSIs. Although some reliable two-way interactions occurred between these factor effects, S-R compatibility and RSI affected mean switching-time costs almost additively. Furthermore, despite these effects, large STCs persisted after the longer RSI, as other investigators have found (e.g., Allport & Wylie, this volume; Allport, Styles, & Hsieh, 1994; De Jong, this volume; Rogers & Monsell, 1995).[1] It is this overall pattern for which our EPIC model accounts.

[1] First-task responses yielded a similar pattern of results. Although mean RTs decreased across sessions, their pattern did not change qualitatively with practice. No significant asymmetries

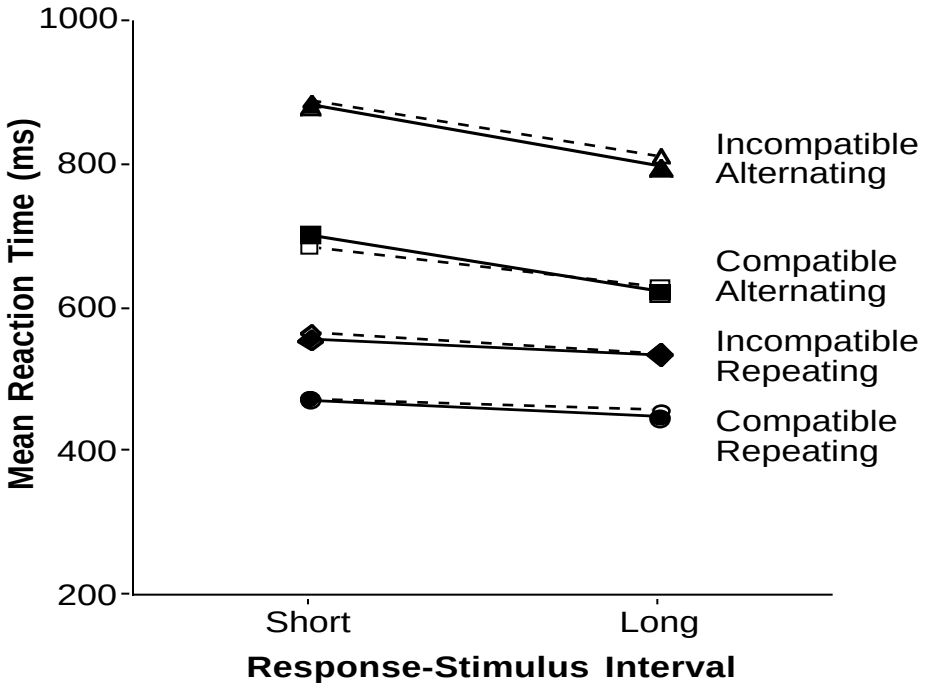

Figure 2. Second-task RTs from the first session in Lauber's (1995) study of task switching. The dark points connected by solid lines represent observed mean RTs as a function of RSI, task difficulty (compatible versus incompatible S-R mappings), and trial type (alternating-task versus repeating-task trials). The light points connected by dashed lines represent simulated mean RTs produced by the EPIC model in Figure 3.

EPIC models. Of course, there are various ways that we could model task switching with EPIC. For example, one conceivable model would have two sets of task-specific, goal-sensitive production rules available simultaneously in procedural memory. In this case, the rules used to select responses for Lauber's incompatible Tasks C and D might have the following forms:

```
IF
((GOAL PERFORM TASK C)
 (STEP MAKE PRESS-RESPONSE TO DIGIT 1)
 (VISUAL ?OBJECT DIGIT 1)
THEN
((SEND-TO-MOTOR MANUAL PERFORM PRESS (RIGHT MIDDLE))
 (DELWM (STEP MAKE PRESS-RESPONSE TO DIGIT 1))
 (ADDWM (STEP WAIT-FOR PRESS-DONE)))
```

occurred in switching-time costs. Error rates were moderately low (< 10%) on average and correlated positively with mean RTs, suggesting no systematic speed-accuracy tradeoffs.

```
IF
((GOAL PERFORM TASK D)
 (STEP MAKE PRESS-RESPONSE TO DIGIT 1)
 (VISUAL ?OBJECT DIGIT 1)
THEN
((SEND-TO-MOTOR MANUAL PERFORM PRESS (RIGHT RING))
 (DELWM (STEP MAKE PRESS-RESPONSE TO DIGIT 1))
 (ADDWM (STEP WAIT-FOR PRESS-DONE)))
```

Given the simultaneous availability of such rules, an executive process could switch tasks simply by changing the task goal items in WM, disabling one task's rules and enabling the other's.

Yet this type of model would fail to account for persistent large STCs such as Lauber observed. Under EPIC, changing goal items takes only one cognitive-processor cycle, which should be completed within about 50 ms regardless of other prevailing factors. However, Lauber's STCs ranged from 200 to 300 ms, they endured after a relatively long (750 ms) RSI, and S-R incompatibility affected them reliably. So additional delays associated with other control operations besides changing goal items presumably contributed to task switching here. Perhaps these contributions occurred because the tasks had different S-R mappings but involved the same stimuli and responses. Such mapping conflicts might substantially increase the amount of practice needed to learn adequate task-specific, goal-sensitive production rules (Anderson, 1983), requiring participants to rely initially on other types of procedural and declarative knowledge instead.

Thus, our modeling of Lauber's results has taken an alternative direction. Consistent with some other theorists (e.g., Rubinstein et al., 1995), we assume that to reduce conflicts in switching between similar tasks, five constraints are imposed: (1) at each moment, symbolic S-R mapping information for performing just one task is kept in WM; (2) switching tasks involves removing currently irrelevant information from WM; (3) the irrelevant information is replaced with relevant information for the next task; (4) these "cleanup" and "setup" operations entail relatively slow interactions with long-term memory; (5) setting up for the next task is triggered by its stimulus onset.

On the basis of these assumptions, we have formulated a model with a single set of generic production rules that perform both of Lauber's incompatible tasks. For each incompatible task, these rules select responses by using a particular list of S-R pairs in WM. This involves checking the stored S-R pairs serially to find one whose stimulus term matches the presented stimulus (cf. Theios, 1973). When the match is found, its associated response term is sent to the manual motor processor. Given this protocol, task switching requires not only changing task goal items but also retrieving the next relevant S-R pairs from long-term memory.

For performing both of Lauber's compatible tasks, the model has another set of generic production rules. They assume that EPIC's visual perceptual processor directly recodes each presented stimulus into two response symbols appropriate for the alternative compatible S-R mappings (e.g., "1" --> "index finger" and "1" --> "little finger"). A task rule then chooses and sends one or the other of these response symbols to the manual motor processor. This choice is made by referring to a WM strategy item that indicates which S-R mapping is currently relevant. Given this protocol, task switching requires not only changing task goal items but also retrieving the relevant strategy item from long-term memory.

These operations are controlled by an executive process that takes different paths for alternating-task and repeating-task trials (Figure 3). At the start of repeating-task trial blocks, the executive process calls a subroutine that sets up WM to perform a particular task, and then lets this task be performed twice during each trial. In contrast, at the start of each alternating-task trial, the executive process waits until the first-task stimulus has been recognized, next calls the subroutine that sets up WM for the first task, and then lets the first task be performed. After the first-task response has been made, the executive process calls another subroutine that cleans up WM, waits until the second-task stimulus has been recognized, calls the setup subroutine for the second task, lets the second task be performed, and finally cleans up WM again. Fitting our model to Lauber's data required adjusting the times taken by the WM setup and cleanup subroutines.

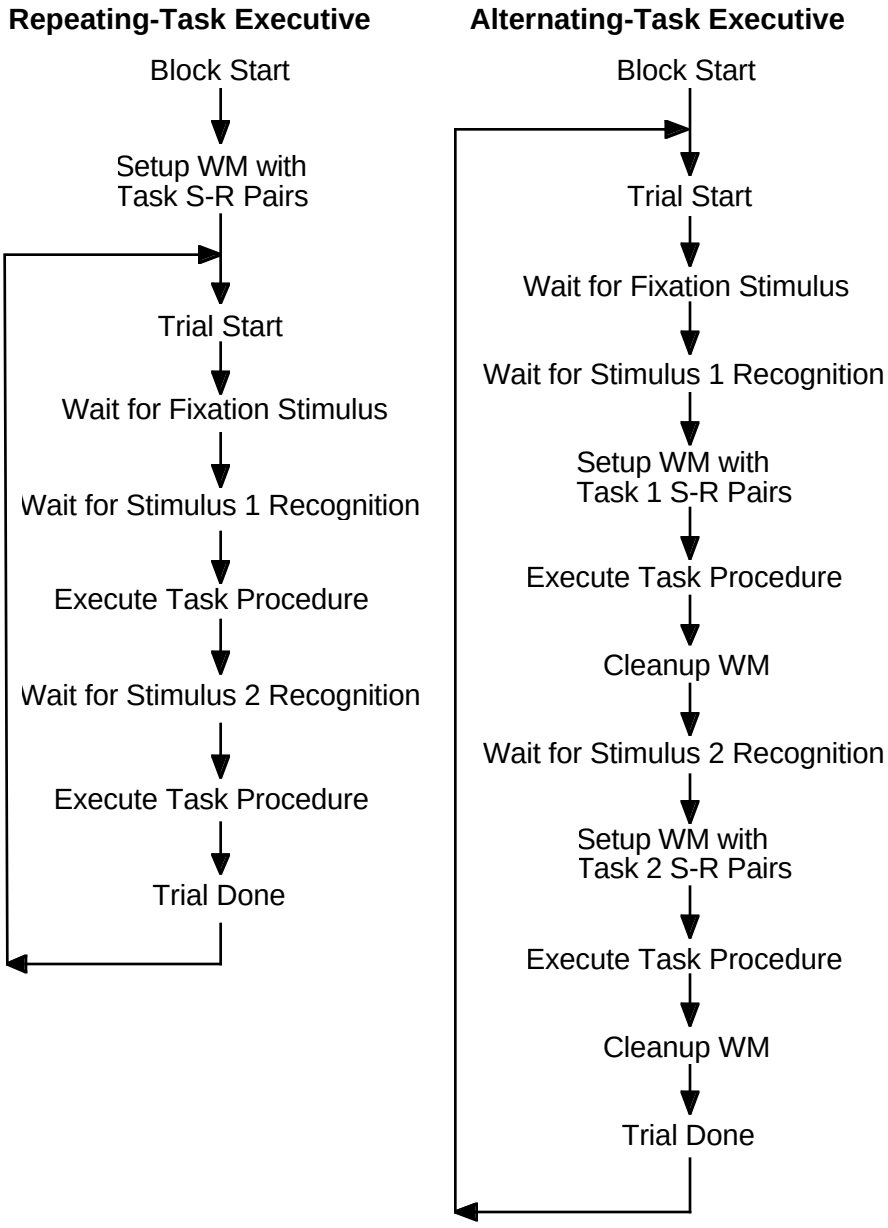

Figure 3. Flowchart of executive processes on repeating-task trials (left side) and alternating-task trials (right side) in the EPIC model for Lauber's (1995) study of task switching. Mean RTs produced by this model appear in Figure 2.

Simulated results. Figure 2 shows mean second-task RTs that our model produced. They account well for not only the main effects of trial type, RSI, and S-R mapping, but also their

additivities and interactions.[2] Our model succeeds much better than would one that switches tasks simply by changing goal items in WM.

Theoretical implications. The durations of WM setup and cleanup operations that we needed to fit Lauber's data are each about 150 ms. Why might they be so long? One possible answer is that in reality, these operations entail gradually activating relevant and inhibiting irrelevant symbolic long-term memory representations (cf. Allport et al., 1994; Anderson, 1983; Goschke, this volume). This would explain why STCs persist at long RSIs and WM setup is not started until the next task's stimulus has been recognized. Perhaps the executive process waits to start setting up WM because stimulus recognition helps amplify requisite memory activation. At present, EPIC does not implement such activation explicitly. Thus, supplementing EPIC with appropriate activation mechanisms could prove worthwhile.

Still, from our present perspective (Figure 3), the executive processes for task switching seem relatively simple. Other than calling WM setup and cleanup subroutines, they contribute very little to STCs. This is consistent with claims of Allport et al. (1994), who questioned whether task-switching studies reveal much about executive mental processes per se. Nevertheless, such studies could have further benefits in other respects. For example, they may yield new insights about the representation of procedural task knowledge, extending what we have discovered already through EPIC modeling.

Discrete Concurrent Tasks

A second context in which EPIC has enabled us to learn more about executive mental processes is the *psychological refractory period* (PRP) procedure (Pashler, 1994, this volume). In this procedure, participants perform two concurrent choice-reaction tasks during series of discrete trials. Typically the tasks involve different stimuli and responses. On each trial, a Task 1 stimulus is followed by a Task 2 stimulus. Because the stimulus-onset asynchrony (SOA) is relatively short, the Task 2 stimulus may precede the Task 1 response. However, participants are instructed to give Task 1 higher priority, and they may be encouraged to make the Task 1 response before the Task 2 response. RTs and response accuracy are measured as a function of the SOA and other task factors. The PRP procedure interests us because, despite its task prioritizing and stimulus sequencing, there is potentially ample opportunity for Tasks 1 and 2 to be performed at least somewhat in parallel. By formulating EPIC models under these conditions, we can better understand how such cognitive control is achieved.

EPIC model. For example, Figure 4 outlines the executive process of one model that has been tested extensively in our research concerning the PRP procedure (Meyer & Kieras, 1997a, 1997b). Here the executive process puts Tasks 1 and 2 respectively in "immediate" and "deferred" modes at the start of each trial. This is done by adding strategy items (e.g., STRATEGY TASK 1 IS IMMEDIATE) to WM. Putting Task 1 in immediate mode lets its responses be selected and sent to their motor processor as quickly as possible for movement production. While Task 2 is in deferred mode, its production rules can select symbolic identities of Task 2 responses and store them in WM, but the selected Task 2 response identities are not sent to a motor processor, and they are not produced as overt movements. However, when a prespecified "unlocking event" occurs subsequently (e.g., the overt Task 1 response is initiated), the executive process shifts Task 2 to immediate mode. Following this shift, previously selected Task 2 responses may be sent from WM to their motor processor for movement production. Also, if response selection has not yet finished for Task 2 before it is shifted to immediate mode, then subsequently the Task 2 production rules will both select and send the Task 2 responses directly to their motor processor.

---

[2] The model also accounts well for mean first-task RTs and the factor effects on them.

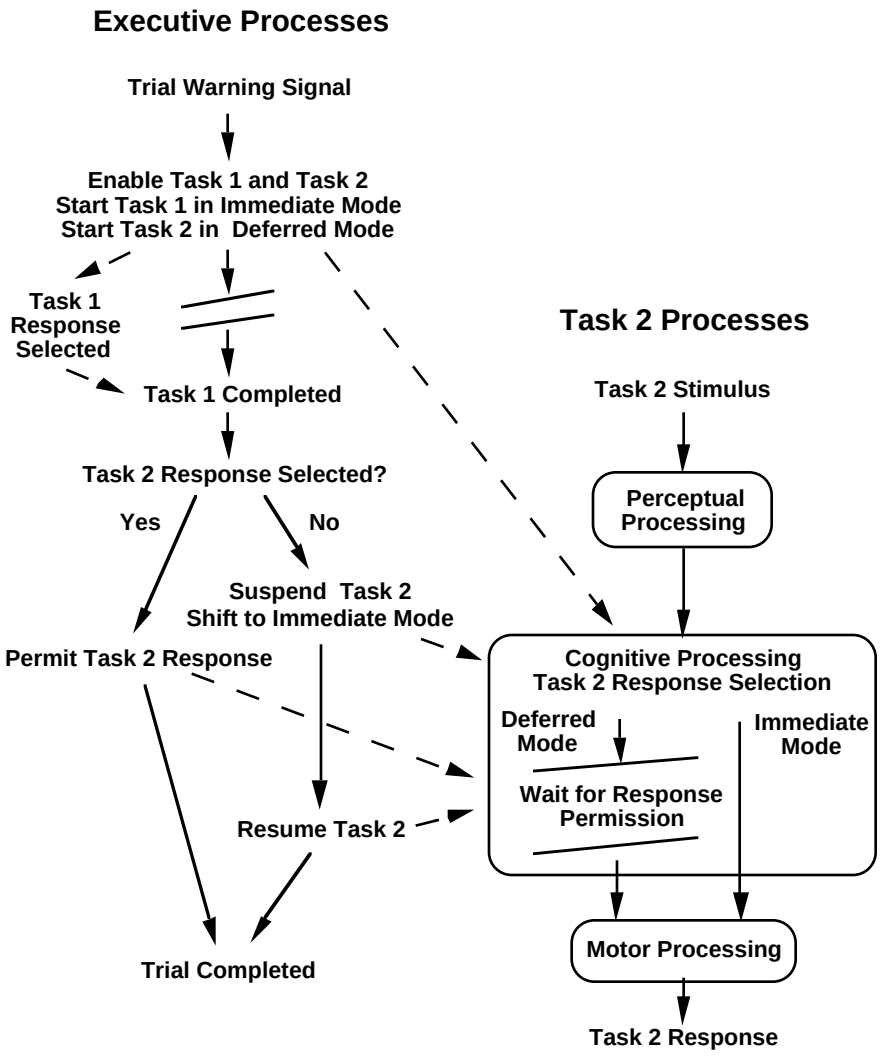

Figure 4. Flowchart of executive and secondary-task processes in the EPIC strategic response-deferment model for the PRP procedure.

Simulated versus empirical results. Comparisons between simulated and empirical results from various studies with the PRP procedure have been encouraging. Our EPIC strategic response-deferment model accounts accurately for differences between observed mean Task 1 and Task 2 RTs as well as additive and interactive factor effects on them. The model's goodness-of-fit is typically high ($R^2 > .95$) and involves only modest numbers of "free" parameters.

Theoretical implications. Our research has revealed that people schedule the PRP procedure's tasks through a combination of various mechanisms. Symbolic response codes for Tasks 1 and 2 may be selected concurrently under flexible strategic control whereby physical movements are produced in proper serial order. We have found no evidence that skilled dual-task performance is constrained by immutable "hardware" decision or response-selection bottlenecks, contrary to traditional response-selection bottleneck hypotheses (cf. in this volume, Jolicoeur, Dell'Acqua, & Crebolder; Ivry & Hazeltine; Pashler).

Elementary Continuous Tasks

The preceding conclusions based on EPIC have been strengthened by formulating computational models of executive mental processes for elementary continuous tasks (Kieras & Meyer, 1997). Here the focus is on visual-manual tracking and choice-reaction tasks that must be performed without predictable pauses along the way. By fitting quantitative results obtained under such conditions, we further demonstrate the existence and generality of strategic cognitive control that judiciously overlaps stages of processing in human multiple-task performance.

Martin-Emerson and Wickens' study. For this demonstration, our research has dealt especially with a study by Martin-Emerson and Wickens (1992). Their participants viewed upper and lower windows on a display screen. In the upper window were a circular target and crosshairs cursor. During 1-min test intervals, the cursor's location was perturbed haphazardly by an accelerative forcing function. The participants performed a compensatory tracking task, moving a right-hand joystick to keep the cursor on target. The tracking task was either hard or easy, requiring more or less frequent joystick movements. Meanwhile, in the lower window, horizontal arrows appeared intermittently. Depending on whether an arrow pointed right or left, the participants pressed a left-hand index or middle finger button. The centers of the task windows were separated by a visual angle that varied systematically across test intervals. As this angle increased, eye movements that traveled greater distances were required for the stimuli to be identified correctly. Both the tracking and arrow-discrimination tasks were supposed to receive high priority.

Empirical results. Figure 5 shows some results of this study. Mean RTs for the arrow discriminations increased reliably with the visual angle between display windows but were relatively unaffected by tracking difficulty. In contrast, root mean square (RMS) tracking errors were reliably greater for hard tracking, but the visual angle affected them relatively little. This occurred even though the tracking errors were measured during 2-s intervals that started at the onsets of the stimuli for the arrow-discrimination task.

EPIC models. To account for these results, we first formulated an EPIC model that uses "lockout" scheduling, which let us test predictions based on the traditional response-selection bottleneck hypothesis (cf. Pashler, this volume). Under this model, whenever an arrow occurs, tracking is suspended as soon as possible, performance of the arrow-discrimination task proceeds until completion, and then tracking is resumed. Given realistic delays in EPIC's motor processors, such lockout scheduling yielded excessively large RMS tracking errors. These discrepancies led us to reject this first model and to formulate a second model that instead uses more efficient overlapped task scheduling.

Figure 6 shows the task and executive processes of our second model. Here the executive process takes two initial steps: production rules for the tracking and arrow-discrimination tasks are enabled to select responses in immediate and deferred modes, respectively; also, commands are sent to the ocular motor processor for keeping the eyes on target. Subsequent perceived cursor movements trigger the tracking task's rules, which send commands to the manual motor processor for producing joystick movements, keeping the cursor positioned on target. Meanwhile, visual onsets of arrow stimuli may be detected. If an arrow is detected foveally or parafoveally, then it is identified without further ado, selection of a button press proceeds for it, and tracking continues until the identity of the button press becomes available in WM. Otherwise, if an arrow occurs in peripheral vision, the executive process takes several extra steps: it suspends tracking, moves the eyes to look at the arrow and start its identification, returns the eyes to look at the cursor, and resumes tracking until a deferred-mode button press has been selected by the rules for the arrow-discrimination task. Insofar as possible, this lets tracking continue simultaneously with arrow identification and button-press selection. Immediately after the selection of each button press, the executive process also suspends tracking and lets the identity of the selected button press be sent to the manual motor processor, after which tracking is resumed again. Thus, the present overlapped task-scheduling model is similar to our previous model for the PRP procedure (cf. Figure 4).

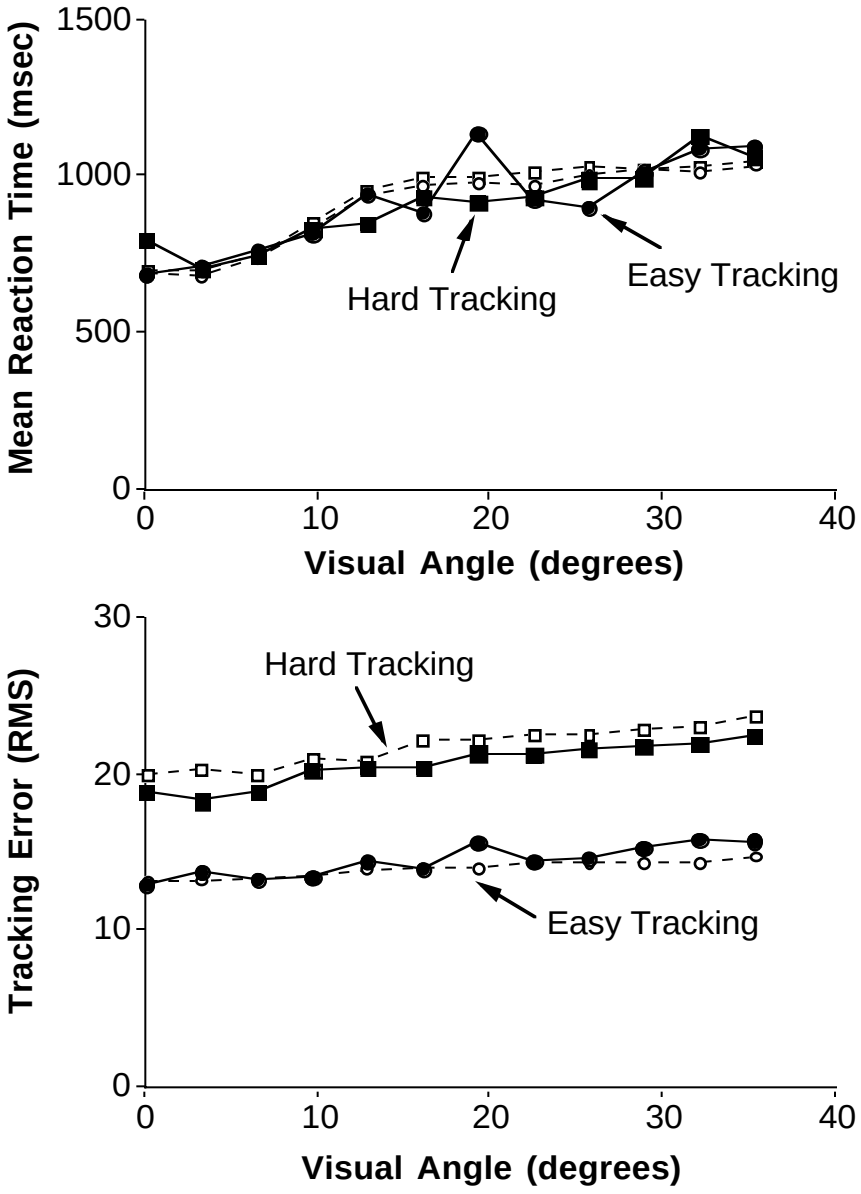

Figure 5. Results from Martin-Emerson and Wickens' (1992) study. Top panel: Observed mean RTs (dark points on solid lines) and simulated mean RTs (light points on dashed lines) produced by the EPIC model in Figure 6 for the arrow-discrimination task when it was performed concurrently with either an easy or hard visual-manual tracking task. Bottom panel: Observed and simulated mean RMS errors for the visual-manual tracking task when it was easy or hard.

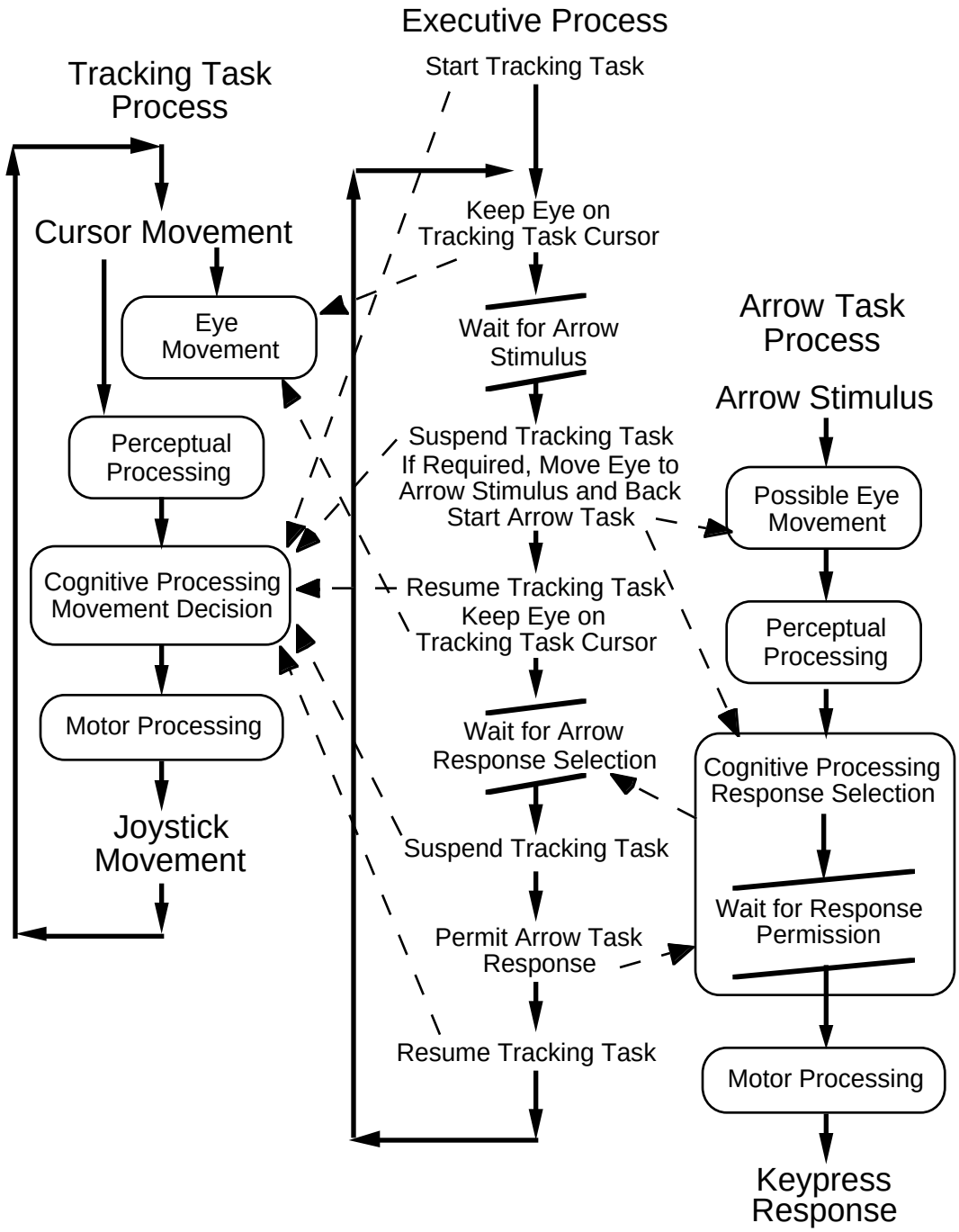

Figure 6. Flowchart of an EPIC model with a customized executive process that implements overlapped task scheduling for Martin-Emerson and Wickens' (1992) study. Dashed diagonal arrows from the executive process to the concurrent tracking and arrow-discrimination task processes represent context-dependent supervisory control imposed under these conditions. Mean RTs and RMS tracking errors produced by this model appear in Figure 5.

Simulated results. Figure 5 shows simulated results from the present model. Its mean RTs and RMS tracking errors closely approximate those produced by actual participants. Unlike lockout scheduling, overlapped scheduling does not yield excessively large tracking errors.

Theoretical implications. The present model's success supports our claims about how executive mental processes may temporally overlap visual, response selection, ocular motor, and manual motor operations in multiple-task performance. Apparently the types of control mechanism and scheduling strategies that we have proposed for discrete concurrent (e.g., PRP) tasks contribute as well to efficient performance of elementary continuous tasks. These mechanisms seem to be used regardless of whether the tasks involve the same (e.g., visual-manual) or different (e.g., auditory-vocal and visual-manual) perceptual-motor modalities.

Compound Continuous Tasks

Our characterization of executive mental processes applies not only to elementary but also to compound continuous tasks that entail several distinct subtasks. For example, one illustration of such generality has arisen from considering a study by Ballas, Heitmeyer, and Perez (1992). Their study, which simulated military aircraft operations, focused on concurrent visual-manual tracking and tactical decision making. In tracking, participants plied a joystick to superimpose a cursor over an evasive target plane. In tactical decision making, participants pressed finger keys to designate the hostility of numbered icons that depicted jet fighters, bombers, and missile sites. Because there were various types of icon and designation criteria, this decision making constituted a compound task.

To account for performance under these quasi-realistic conditions, we have found that an EPIC model with a three-level hierarchy of executive and task processes fits empirical data well (Kieras & Meyer, 1997, 1999; Meyer & Kieras, 1997b, 1999). As part of this model, a supraordinate dual-task executive process provides overall supervision for a tracking process, a display monitoring process, and a tactical executive process that coordinates three subprocesses -- stimulus icon selection, hostility response selection, and track-number response selection -- in tactical decision making (e.g., see Figure 7). Through this hierarchical control, the relative priority of tactical decision making and the temporal overlap of its subprocesses are varied dynamically, contingent on the numerosity of potentially hostile icons in the display. The model, with its adaptive scheduling mechanisms, accounts well for observed sequences of tactical-decision RTs and RMS tracking errors (e.g., see Figure 8).

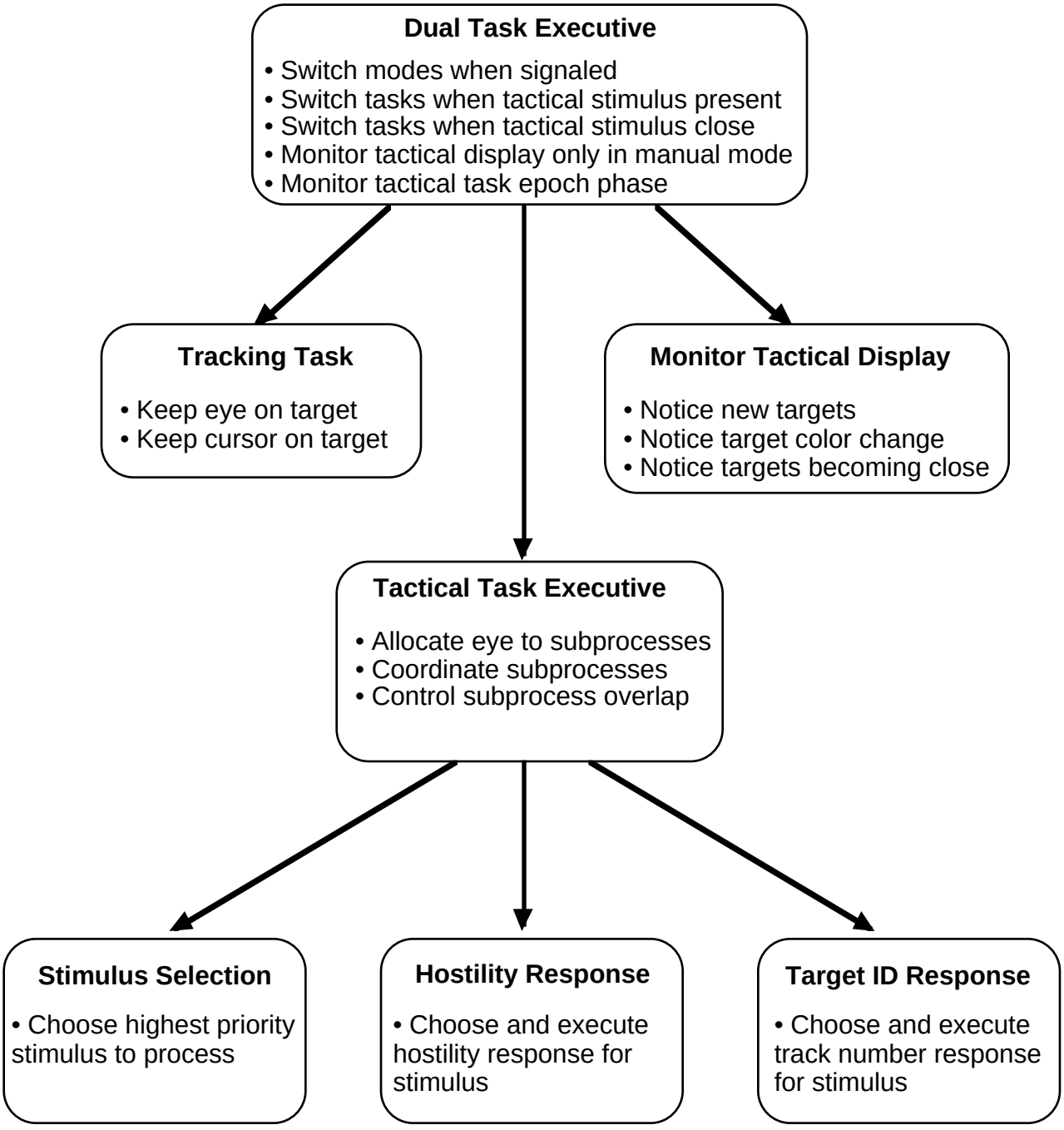

Figure 7. Flowchart of EPIC model for dual-task performance in Ballas et al.'s (1992) study. Mean RTs that this model produced for tactical decision making with dynamic overlapped task scheduling appear in Figure 8.

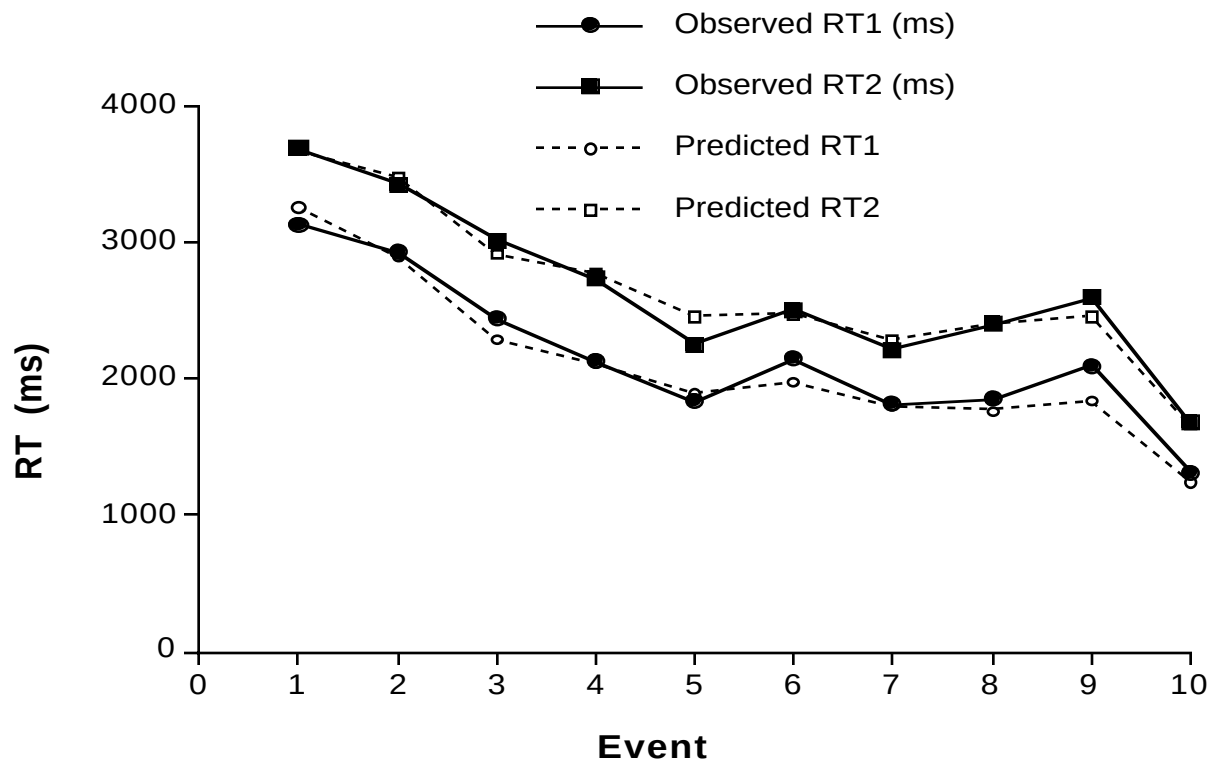

Figure 8. RTs for the tactical-decision task in Ballas et al.'s (1992) study. The dark points connected by solid lines represent observed mean RTs of hostility designation responses (circles) and track-number designation responses (squares), respectively, plotted versus the serial positions of successive tactical decisions. The light points connected by dashed lines represent simulated mean RTs produced by the EPIC model in Figure 7 when it dynamically varied the degree of overlapped task scheduling.

Interim Status Quo

From the preceding illustrations, it should be clear that EPIC yields significant theoretical insights about executive mental processes. Still, our progress thus far has been limited in some major respects.

Limitations of EPIC models. One limitation is that the executive processes of our models have been customized for particular task combinations. Although these processes may be somewhat similar across contexts, their formulation has incorporated considerable task-specific knowledge. For example, in modeling Martin-Emerson and Wickens' (1992) study, the executive directly controls eye movements from the stimulus arrows to the tracking cursor (Figure 6). This enhances tracking performance, consistent with available data, but makes the executive context dependent and non-modular. To be strengthened further, EPIC needs general executive processes that are context independent.

Previous theorists likewise have stressed the importance of general executive processes, as exemplified in proposals about the Central Executive (Baddeley, 1986) and Supervisory Attentional System (Norman & Shallice, 1986). However, they have not provided explicit computational algorithms that achieve the required generality. Thus, we must look elsewhere for ways to fulfill this need.

Accompanying EPIC's lack of general executive processes is a related deficiency. Competition among processes for access to limited hardware resources may cause miscommunication or deadlock, in which wrong information is transmitted or processes become perpetually stalled (Stallings, 1998). EPIC does not yet solve these concurrency problems adequately. Without adequate solutions, veridical modeling of complex adaptive multiple-task performance will be impossible.

Another limitation is that EPIC does not yet deal with procedural learning in multiple-task performance. How do people learn to schedule and coordinate concurrent tasks efficiently? How are their multitasking skills transfered across situations? Deeper answers are needed for modeling skill acquisition and developing effective instructional techniques in practical applications (Gopher, 1993).

Potential contributions of OS fundamentals. Fortunately, contemporary computer operating systems may stimulate further theorizing. Fundamental principles that underlie their operation provide basic ways for implementing context-independent control and solving problems of task concurrency (Stallings, 1998). By considering these fundamentals, we may augment EPIC with needed general executive processes, concurrency solutions, and multitasking skill acquisition.

Contemporary Operating Systems and Computer Technology

Contemporary OSs supervise information processing for task programs that are executed virtually or actually in parallel. Limited capacities of computer hardware impose constraints with which an OS must cope while trying to maximize process throughput. Accordingly, we next consider aspects of both hardware design and OS functions that bear on these matters.

Hardware Design

Starting with early computers like ENIAC, hardware design has become increasingly sophisticated (Tucker, 1997). As a result, modern computers typically have at least one central processing unit (CPU), at least one memory unit, and various input/output (I/O) peripherals. The CPU executes sequences of instructions for system and task programs. The memory unit stores programs and data, letting them be manipulated in similar ways. Thus, generic information-processing capabilities are implemented by the hardware, while overall system control and task procedures are provided by the software.

Uniprocessor architecture. Many OSs and task programs are used on computers with one CPU. Although this uniprocessor architecture executes instructions sequentially in some respects, its

components enable extensive parallelism. For example, separate streams of data may be transmitted simultaneously to or from different I/O peripherals, and the CPU may perform multiple suboperations in parallel. Exploiting such capabilities, an OS can sustain concurrent threads of processing at least somewhat as if each program had its own CPU.

Multiprocessor architectures. Moreover, some OSs and task programs have been implemented with multiple CPUs. These multiprocessor architectures enable true parallel processing and provide enormous, relatively inexpensive, computational power. Particularly relevant for us is the *shared-memory symmetric multiprocessor* (SMP). In it, multiple CPUs function as equivalent "peers" that share one memory unit and I/O peripherals. This corresponds at least approximately to EPIC's organization. Although EPIC has one cognitive processor, it tests conditions and executes actions of multiple production rules in parallel. Thus, when programmed with two or more rule sets, the cognitive processor emulates a collection of peer CPUs. Also as in a SMP, these rule sets share WM and I/O peripherals.

Consequently, contemporary OS fundamentals should be applicable to EPIC. Indeed, computer scientists have discovered that OS fundamentals are extremely general, applying across many uniprocessor and multiprocessor architectures. This suggests that what OSs and EPIC teach us will likely hold as well for the human mind and brain, which implement forms of multiprocessor parallelism too.

OS Background

To appreciate OS fundamentals, more background about them is in order (Stallings, 1998; Tucker, 1997).

History. OSs, like computer hardware, have become increasingly sophisticated. For early computers (circa 1950), people loaded and started programs manually. Subsequently (circa 1960), primitive OS *resident monitors* were developed to automate these processes. Following this development, OS capabilities were gradually extended. For example, one extension enabled overlapping CPU and I/O operations so that the CPU would not have to wait idly on slow mechanical devices. These advances led to *multitasking*, an overarching OS function (circa 1970).

In multitasking, an OS interleaves or overlaps execution of task programs that require certain limited hardware resources. When an execution process has taken a set time or must wait on pending I/O, it is suspended, and the CPU is allocated to another process. After completion of I/O or other prerequisites, the suspended process is resumed. Consequently, multiple processes may advance efficiently without individual users' intervention. Software for multitasking on uniprocessors has been adapted gracefully for multitasking on multiprocessors.

Objectives. Contemporary OSs achieve several objectives. Systems programmers developed OSs to keep CPU and memory hardware as busy as possible, increasing process throughput. OSs have also made it simpler and faster to formulate *non-cooperating task programs*, which are executed asynchronously and compete for hardware resources. Given OS services, such a program can be formulated as if it were the only one executed and no intricate control of I/O were required. Furthermore, OSs have facilitated the formulation of *cooperating task programs*, which are executed synchronously and share their products interactively.

However, OSs are neither logically necessary nor maximally efficient in every respect. Non-hierarchical "flat" programs can be formulated to perform multiple tasks concurrently on "bare" computer hardware without OS support. Through this formulation, the computational overhead of hierarchical software can be eliminated, and even faster performance can be achieved. Nevertheless, such improvement has serious costs. It requires dealing directly with many levels of control, so the time and effort to formulate flat programs can be exorbitant. Also, flat programs do not readily generalize beyond their original applications. In contrast, OSs provide a better compromise between speed of execution versus ease and generality of software development.

OS Functions

This compromise is enabled by OS functions that solve a basic problem: detailed sequences of execution for independent task programs cannot be predicted accurately. An OS must ensure that execution proceeds correctly and rapidly despite unpredictable interruptions and resumptions. The solution entails judicious task scheduling, resource allocation, process coordination, and conflict resolution.

Task scheduling. In task scheduling, decisions are made and implemented about when programs will be executed. Doing so requires prioritizing, preparing, initiating, suspending, preserving, resuming, and terminating each execution process at apt moments. OSs use various scheduling algorithms for this. Among them are *first-come first-serve, round robin, shortest remaining time, shortest-process next, highest response ratio,* and *least-time-consumed scheduling*, each of which may produce relatively high or low performance, depending on nuances of the prevailing context. Task scheduling by an OS must therefore be "tuned" adaptively to maximize overall throughput.

Resource allocation. An OS must also allocate hardware resources judiciously to individual processes, depending on resource availability and process needs. For example, during execution, a process may request resources; the OS may comply by allocating them immediately if they are available. Alternatively, if resources have been committed to other processes already, then the OS may deny the current request temporarily, and perhaps suspend the requesting process until its needs can be satisfied. Exactly when processes request and release their resources, combined with how the OS handles them, contributes significantly to attained performance.

Process coordination. Among the processes that are being executed, some may need to share intermediate products of their computations. For this sharing to succeed, these cooperating processes must be coordinated. Their coordination is required because interprocess communication involves writing to and reading from the same memory locations in proper serial order.

To facilitate interprocess communication, an OS performs several coordinative functions, including *mutual exclusion*, *process synchronization*, and *message passing*. Relying on these functions, a receiving process may request that the OS suspend it until an expected message arrives from another sending process. When the sending process is ready to transmit this message, it may request that the message be passed to the receiving process. The OS may then pass the message and resume its recipient.

Conflict resolution. Because concurrent processes impose high loads on hardware resources and may be non-cooperative, serious conflicts can arise. An OS has to avoid these conflicts as best possible, and resolve them gracefully when need be. This function is crucial for dealing with *deadlocks*, which entail closed chains of processes such that each process currently has exclusive ownership of some resource needed by the next process in the chain. Adaptive conflict resolution also helps deal with other undesirable situations such as *starvation*, in which some low-priority process is perpetually preempted by higher-priority processes.

Cognitive Control and OS Fundamentals

In light of the preceding discourse, it appears that assimilating OS fundamentals into theories of cognitive control might be beneficial. Contemporary OSs embody precise and comprehensive instantiations of executive processes. Such instantiations are scarce in current psychological theories. Thus, to promote further progress, we next discuss some stimulating theoretical concepts, multitasking models, and explanatory hypotheses inspired by these considerations.

Theoretical Concepts

The concepts that interest us here involve distinctions between various types of executive and task processes.[3]

Customized executives. One major distinction concerns customized versus general executives. By customized executive (CE) we mean a modular set of supraordinate mental processes that manage multiple-task performance based on unique context-dependent knowledge about the particular tasks and their temporal interrelations. A CE works for only one task combination and cannot be transferred readily across different situations.

Thus far, EPIC models have all used CEs. An instructive case of this is our model of performance in Martin-Emerson and Wickens' (1992) study. Its executive process preallocates resources (i.e., ocular and manual motor processors) to tracking and arrow discrimination without these task processes requesting them explicitly (Figure 6). The preallocation is possible here because the executive "knows" the task processes' needs already and satisfies them in proper sequence. Such use of context-dependent knowledge may be common after extensive practice under conditions in which high performance is desired.

However, our theorizing need not be confined to models with CEs. New EPIC models may be formulated on the basis of general executives that function at least partly like contemporary OSs. From testing them empirically, we learn more about the extent to which OS fundamentals characterize how human multiple-task performance is controlled.

General executives. A general executive (GE) is a modular set of supraordinate mental processes that manage multiple-task performance without using unique context-dependent knowledge about the tasks and their temporal interrelations. Given such generality, cognitive control can be achieved for different task combinations through standard functions like those of contemporary OSs. Implementing these functions in EPIC is straightforward because it resembles a shared-memory symmetric multiprocessor.

Nevertheless, determining whether a GE should be added to EPIC requires answering a fundamental question about cognitive control: do people have GEs and use them for multiple-task performance? We might expect an affirmative answer given the potential ease of preparing and efficiency of executing task programs based on GE functions. Yet the only way to be sure about this is to formulate and test EPIC models that rely on a GE. We take this course after introducing more distinctions that will facilitate our pursuits.

Managerial styles. Another relevant distinction concerns managerial styles of GEs. At one extreme, a *conservative GE* can have a strict regimented style of scheduling task processes and allocating limited resources to them. Under such regimentation, task processes may have to request resources before using them; processes may be suspended when their requested resources are unavailable; and processes that are not prone to make deferent resource requests may be kept from starting (i.e., locked out) until others have finished. Alternatively, a *liberal GE* can have a tolerant laissez-faire managerial style. Given this tolerance, task processes may be allowed to proceed at least partially unabated while their requested resources are unavailable, and processes that are prone to use resources without requesting them also may be accommodated insofar as possible. In principle, a GE's managerial style is adaptable to particular situations. Such adaptability, contingent on the "manners" of task processes, will determine the attained level of multiple-task performance.

Process manners and etiquette. Task processes can have various manners of interaction with a GE. Proper etiquette for a task process entails requesting resources (e.g., motor mechanisms) immediately before they will be used, waiting for the GE's permission to use them, and then releasing the resources immediately after their use is complete. A *polite process* conforms to all of

---

[3] Insofar as we know, the following distinctions have not been made explicitly in OS textbooks. They are introduced here to address issues about human cognitive control. This is essential because these issues extend beyond ones associated with computer applications where experienced task programmers adhere consistently to common a priori conventions.

these rules. This establishes favorable circumstances for a laissez-faire managerial style through which relatively high multiple-task performance is attainable.

Theoretically, however, some task processes may be impolite. For example, a *presumptuous process* might use crucial resources without requesting them. An *impatient process* might request resources but not wait for permission to use them. A *greedy process* might request resources too early and release them too late. Such inconsiderate conduct will force a GE to be more conservative, curtailing the processes' temporal overlap and impeding their progress.

Cost-benefit assessment. To assess the costs and benefits of alternative GE managerial styles, various factors are relevant. One is *interaction overhead*, which includes scheduling, allocation, and abdication costs for supervising task processes. *Scheduling costs* are times consumed by adding and deleting goals in WM to start, suspend, resume, and terminate processes selectively. *Allocation costs* are times consumed by making and fulfilling resource requests. *Abdication costs* are times consumed by releasing resources. Ideally, these costs should be paid in ways that decrease *resource-possession times* (i.e., amounts of time during which a task process possesses crucial resources). Also, as best possible, the payments should increase *process-overlap intervals* (i.e., intervals during which multiple processes are advancing simultaneously).

Taking these factors into account, impolite task processes may escape some interaction overhead, but they increase resource-possession times and force the GE to eliminate process-overlap intervals. In contrast, a liberal GE and polite task processes make an attractive compromise. Their process-overlap intervals and resource-possession times may be relatively long and short, respectively, thereby more than compensating for the GE's moderate interaction overhead.

Nevertheless, there are other ways to perform better on all scores. CEs (customized executives) tuned for particular task combinations can achieve even lower interaction overhead, shorter resource-possession times, and longer process-overlap intervals. As discussed later, this leads to interesting hypotheses about multitasking skill acquisition.

New Multitasking Models

To illustrate how these theoretical concepts help clarify the nature of cognitive control, we have implemented them in two new EPIC models for Martin-Emerson and Wickens' (1992) study. Model 1 has a conservative GE that supervises two impolite task processes. Model 2 has a more liberal GE that supervises two polite task processes. By comparing these models to our previous one that has a customized executive (Figure 6), we examine the effects of managerial style and process manners on multiple-task performance.

Model 1: Conservative GE with impolite processes. In Model 1, tracking and arrow discrimination are assumed to be impolite processes. They do not request or release resources for producing eye and hand movements. Instead, each process tries to move the eyes and hands without regard for what is happening elsewhere in the system, creating prospects for "jams" in EPIC's motor processors.

To cope with this impoliteness, Model 1 has a GE that uses a first-come first-serve (FCFS) algorithm for scheduling the tracking and arrow-discrimination task processes in strict lockout mode. Under it, these processes may be started optionally when their stimuli (arrows and suprathreshold tracking errors) are detected. However, the GE lets only one process proceed at a time. If stimuli for both processes occur simultaneously, then the lower priority one (viz. tracking) is postponed until the higher priority one (viz. arrow discrimination) has responded to its current stimulus.

This protocol resembles the one of Norman and Shallice's (1986) supervisory attentional system (SAS). There action schemata are activated by "trigger" stimuli and contend for limited response mechanisms. Precluding conflicts from this *contention scheduling*, the SAS transmits top-down activation to the highest-priority schema, favoring it over lower-priority schemata. In our Model 1, the lockout scheduling is like the selective prioritization imposed by the SAS. Thus, we may test both Model 1 and the SAS by comparing Model 1's performance to real data.

Table 1 shows results of this comparison. When a small ($< 5^{\circ}$) visual angle separates the displays of the tracking and arrow-discrimination tasks, simulated RTs from Model 1 are considerably less than observed ones (mean difference = 103 ms), but at larger ($> 10^{\circ}$) angles,

Table 1. Reaction times and tracking errors produced by four alternative EPIC models to account for observed data from Martin-Emerson and Wickens' (1992) study.

Tracking Task	Visual Angle	Mean Arrow-Discrimination RT (ms)					Mean RMS Tracking Error (pixels)				
		Model 1	Model 2	Model 3	Model 4	Data	Model 1	Model 2	Model 3	Model 4	Data
easy	small	608	690	686	612	697	19.4	17.2	13.4	13.5	13.3
	medium	781	836	782	696	787	22.5	18.0	13.6	13.6	13.3
	large	1070	1106	978	886	997	28.6	20.1	14.3	14.5	15.0
hard	small	632	688	686	608	749	28.7	25.2	19.9	19.7	18.6
	medium	802	842	787	700	786	34.8	27.6	20.6	20.9	19.6
	large	1097	1112	996	887	976	43.2	30.0	22.7	22.9	21.4

Note: RTs and tracking errors were measured as a function of the tracking-task difficulty (easy vs. hard) and the magnitude of the visual angle ("small" < 5˚; "medium" 6˚ to 10˚; "large" > 10˚) between the display windows for the tracking and arrow discrimination tasks (cf. Figure 5).

simulated RTs are considerably greater than observed ones (mean difference = 97 ms). Furthermore, Model 1's simulated RMS tracking errors are much larger than observed ones; when tracking is difficult, they differ by more than a factor of two at large visual angles. Model 1 performed very poorly even though under it, tracking and arrow discrimination progress as fast as reasonably possible while they are underway, and there are no resource allocation or abdication costs of supervising them. Instead, the poor performance of Model 1 stems from an absence of process overlap caused by its GE having to cope conservatively with the impoliteness of the task processes in their use of motor resources.

These results disconfirm both Model 1 and the SAS with respect to Martin-Emerson and Wickens' study. Contrary to these models, under at least some conditions, cognitive control for multiple-task performance is more efficient than a conservative GE and impolite task processes allow. We investigate the sources of this efficiency more fully by considering a second new model.

Model 2: Liberal GE with polite task processes. In Model 2, tracking and arrow discrimination are assumed to be polite processes. Each task process requests motor resources immediately before it would use them, does not use them until the GE grants permission, and releases them immediately after they have been used. Given this politeness, the GE lets these processes advance simultaneously insofar as possible, even after one of them has requested resources that the other is currently using. Such liberalism is feasible because the task processes make eye and hand movements in a considerate manner that avoids motor-processor "jams", thereby enabling more process overlap than Model 1 allows.

Another virtue of Model 2 is its straightforward flow of control. Compared to our original model for Martin-Emerson and Wickens' study (Figure 6), Model 2 has a relatively simple flowchart (Figure 9). Consequently, during multitasking practice, the skill embodied in Model 2 should be fairly easy to acquire.

Consistent with these points, Table 1 shows that Model 2 produces somewhat better performance than Model 1 does. Especially when tracking is difficult, simulated RMS errors from Model 2 are markedly smaller compared to those from Model 1. Nevertheless, there remain significant discrepancies between the performance of Model 2 and the observed data. Both the simulated tracking errors and simulated RTs are still excessively large, suggesting that actual participants achieved even more process overlap than Model 2 allows.

Why and how might this be? An answer may come from reconsidering our original model for Martin-Emerson and Wickens' study, to which we now refer as "Model 3".

Model 3: CE with resource preallocation and enhanced task processes. As depicted before (Figure 6), Model 3 uses a customized executive that exploits context-dependent knowledge about the tasks and their temporal relationships. Based on this knowledge, the CE preallocates resources (i.e., ocular and manual motor processors) to tracking and arrow discrimination without being requested to do so. This enables the task processes to advance even more quickly than under Model 2. Under Model 3, the task processes also prepare eye movements beforehand. Together, these enhancements further facilitate performance so that Model 3's simulated RTs and tracking errors are considerably less than those of Model 2, closely approximating observed data (Table 1).

The good fit of Model 3 suggests that participants in Martin-Emerson and Wickens' study achieved excellent multiple-task performance through especially efficient cognitive control. Without this efficiency, limitations of perceptual-motor mechanisms would have precluded such performance. The CE of Model 3 overcomes these limitations more so than a GE can. Nevertheless, during the course of practice, participants may have relied on a GE to acquire their high level of multitasking skill. How this could happen is considered next.

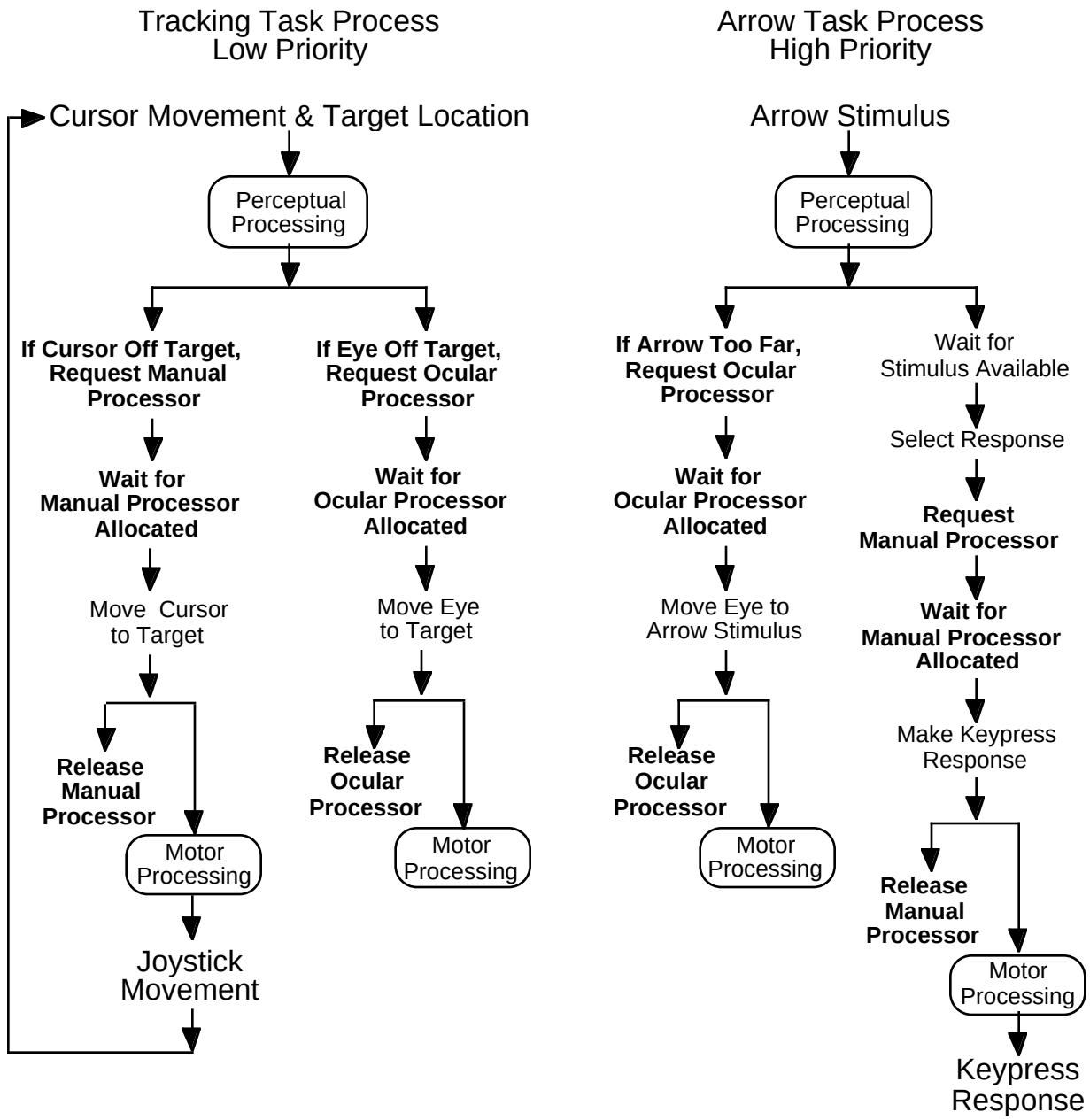

Figure 9. Flowchart of an EPIC model that performs the tracking and arrow-discrimination tasks of Martin-Emerson and Wickens (1992) with polite task processes and a general executive whose managerial style is liberal in task scheduling and resource allocation (cf. Figure 6).

Explanatory Hypotheses

Taken together, our results from Models 1, 2, and 3 lead to hypotheses that explain various major aspects of multiple-task performance and skill acquisition.

Multitasking skill-acquisition stages. We hypothesize that multitasking skill acquisition progresses through several stages. These include *preprocedural interpretative multitasking* (Stage 0), *general hierarchical competitive multitasking* (Stage 1), *general hierarchical cooperative multitasking* (Stage 2), *customized hierarchical multitasking* (Stage 3), and *customized heterarchical multitasking* (Stage 4). Each of these stages can be characterized with respect to its degree of efficiency, types of interaction between executive and task processes, and exploitation of context-dependent procedural knowledge.

Preprocedural interpretive multitasking is necessitated by a fundamental dependence between procedural and declarative task knowledge. We call this "Stage 0" because it occurs at the start of practice before sets of production rules for the particular tasks have been created. During it, people must use a generic interpretive process to execute propositional instructions about how the tasks should be performed. Here performance is presumably slow and error prone, placing heavy loads on WM as people "think" their way verbally through each task. Nevertheless, it is from this explicit directed intentional activity that more efficient procedural knowledge for subsequent task performance emerges (Anderson, 1983; Bovair & Kieras, 1991; Kieras & Bovair, 1986).

Once such knowledge becomes available, general hierarchical competitive multitasking may ensue. We call this "Stage 1" because it is the first one during which a GE would supervise task processes that are executed through individualized sets of production rules. It is also a stage during which task scheduling and coordination are managed as in our Model 1 for Martin-Emerson and Wickens' study. Here performance presumably entails a conservative GE with strict lockout scheduling of impolite task processes whose manners in using perceptual-motor resources are impulsive, presumptuous, and greedy. This impoliteness may be attributed to a need for more practice in order to acquire rules that conform with proper task etiquette.

As practice continues, general hierarchical cooperative multitasking may come next. During this Stage 2, task scheduling and coordination would be managed as in our previous Model 2. Here performance presumably entails a liberal GE with temporal overlapping of task processes that request, use, and release system resources politely. This politeness enables the GE to be more permissive in letting these processes advance rapidly toward completion.

Customized hierarchical multitasking would involve an even higher skill level. During this Stage 3, task scheduling and coordination may be managed as in our previous Model 3. Here unique context-dependent knowledge about the particular tasks and their temporal interrelations presumably is exploited to preallocate system resources without time-consuming requests for them, thereby further increasing temporal overlap among task processes. Also, as in Model 3, these processes may be enhanced to prepare their motor responses anticipatorily.

Culminating this evolution is customized heterarchical multitasking. During this Stage 4, performance presumably is controlled without supraordinate executive processes. Instead, the task processes interact directly with each other, self-governing their resource usage as efficiently as possible. This interaction optimizes overall system throughput, completely eliminating scheduling, allocation, and abdication time costs that contribute to the transaction overhead of hierarchical cognitive control.

Table 1 shows some benefits of such optimization. Here we have included results from a fourth model that uses the customized heterarchical multitasking of Stage 4 to simulate performance in Martin-Emerson and Wickens' (1992) study. The RMS tracking errors of Model 4 closely approximate the data, and its mean RTs are even shorter than observed ones. Although Martin-Emerson and Wickens' participants were highly skilled, they had not yet reached their ultimate asymptotic performance level.

Executive learning mechanisms. Operations within and transitions between the preceding stages of multitasking skill acquisition may be mediated by various executive learning mechanisms (cf. Anderson, 1983; Bovair & Kieras, 1991; Chong & Laird, 1997; Kieras & Bovair, 1986). These

mechanisms could entail several components: a *task interpreter* that executes propositional instructions for performing single and multiple tasks during Stage 0; a *task compiler* that creates rudimentary sets of production rules for the initially impolite task processes of Stage 1; a *task socializer* that refines these processes so they become more polite in Stage 2; an *executive modulator* that tailors the GE's managerial style to be either conservative or liberal, depending how polite the task processes are; an *executive customizer* that creates customized executives to enable even more efficient control in Stage 3; and an *executive integrator* that "flattens" the CEs, converting their flow of control from a hierarchical to heterarchical organization in Stage 4.

We hypothesize that such mechanisms are sensitive to the evolving characteristics of performance. For example, during Stage 1, simultaneous attempts by multiple impolite task processes to produce movements in the same response modality could generate motor-processor "jams". These jams might be detected by the executive modulator, leading it to have the GE be conservative for a period of time during which the task socializer works toward making the task processes more polite. The task socializer and executive modulator also could operate partly on the basis of noticing that the task processes do not request and release resources properly. Later, after the task socializer achieves its objectives, the executive modulator perhaps would adjust the GE to be more liberal because motor-processor jamming has ceased. Accompanying the latter adjustment, the executive customizer might start creating a CE that later triggers the process of hierarchical-to-heterarchical flattening by the executive integrator. Of course, future research will be needed to understand and model the details of such hypothetical learning mechanisms.

Multitasking skill-acquisition phenomena. By doing so, we may eventually explain and predict many empirical phenomena of multitasking skill acquisition. For example, Gopher (1993) has found that multiple-task performance is better after variable-priority rather than fixed-priority training. In his fixed-priority training condition, one group of participants gave equal priorities to visual-manual tracking and choice-reaction tasks throughout a series of practice sessions. In his variable-priority training condition, a second group of participants also gave the two tasks equal priorities on some occasions, but devoted higher priority to either tracking or choice reactions on other occasions. After variable-priority training, the second group performed better than the first group even when the two tasks received equal priorities. Similar results have been reported by Meyer et al. (1995). The benefits of variable-priority training could stem from the task socializer and executive modulator receiving a wider range of feedback that guides them more quickly through successive stages of skill acquisition.

Our hypotheses likewise account for results obtained with some other laboratory paradigms. For example, RTs from the PRP procedure sometimes manifest a response-selection bottleneck (Pashler, 1994, this volume). This seems to occur especially when participants receive relatively little practice at coordinating their primary and secondary tasks (Schumacher et al., 1999). A possible reason is that participants lack sufficient opportunity to socialize initially impolite task processes, so their GE has to deal with this impoliteness through strict lockout scheduling (cf. Meyer & Kieras, 1997a, 1997b).

Conclusions

Assimilating the fundamentals of contemporary computer OSs into theories of cognitive control opens many promising paths for future research. With this assimilation, it will be possible to characterize a wider range of control functions more precisely, and to test more definitively for the existence of general as well as customized executive processes. These advances also will lead to more detailed and veridical analyses of multitasking skill acquisition. Computational modeling based on the EPIC architecture provides one vehicle whereby this progress can occur.

For the present prospects to be fully realized, future research must use a wide variety of empirical procedures to investigate multiple-task performance. This investigation should extend beyond basic laboratory paradigms like the task-switching and PRP procedures. They are helpful for isolating particular elementary control functions. However, by themselves, these procedures come nowhere near to engaging the whole host of executive mental processes with which people are

presumably endowed. Rather, to explore these processes more completely, overlapping-task procedures with complex realistic tasks and unpredictable stimulus-response event sequences will be needed (e.g., Ballas et al., 1992).

Another major path for future research will involve identifying systematic relationships between underlying brain mechanisms and the executive mental processes revealed by taking OS fundamentals into account. Because OS fundamentals apply quite generally to shared-memory symmetric multiprocessors, of which the brain is perhaps one type, it seems reasonable to speculate that the brain implements these fundamentals as well. If so, then insights from EPIC computational modeling, when used to interpret results from studies of brain imaging and focal lesion analysis, could eventually yield fundamental solutions to the mind-body problem of cognitive control.

References

Allport, A., Styles, E., & Hsieh, S. (1994). Shifting intentional set: Exploring the dynamic control of tasks. In C. Umilta & M. Moscovitch (Eds.), Attention and performance XV (pp. 421-452). Cambridge, MA: M.I.T. Press.

Anderson, J. R. (1983). The architecture of cognition. Cambridge, MA: Harvard University Press.

Baddeley, A. D. (1986). Working memory. Oxford, UK: Oxford University Press.

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.

Bovair, S., & Kieras, D. E. (1991). Toward a model of acquiring procedures from text. In R. Barr, M. L. Kamil, P. B. Mosenthal, & P. D. Pearson (Eds.), Handbook of reading research (Vol. II, pp. 206-229). New York: Longman.

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

Chong, R. S., & Laird, J. E. (1997). Towards learning dual-task executive process knowledge using EPIC-Soar. Proceedings of the 19th annual conference of the Cognitive Science Society (pp. 107-112). Hillsdale, NJ: Lawrence Erlbaum Associates.

Gopher, D. (1993). Attentional control: Acquisition and execution of attentional strategies. In D. E. Meyer & S. Kornblum (Eds.), Attention and performance XIV. Synergies in experimental psychology, artificial intelligence, and cognitive neuroscience (299-322). Cambridge, MA: M.I.T. Press.

Kieras, D. E., & Bovair, S. (1986). The acquisition of procedures from text: A production-system analysis of transfer of training. Journal of Memory and Language, 25, 507-524.

Kieras, D. E., & 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.

Kieras, D. E., & Meyer, D. E. (1999). 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, in press.

Kristofferson, A. B. (1967). Attention and psychophysical time. In A. F. Sanders (Ed.), Attention and performance (pp. 93-100). Amsterdam: North-Holland Publishing Co.

Lachman, R., Lachman, J. L., & Butterfield, E. C. (1979). Cognitive psychology and information processing: An introduction. Hillsdale, NJ: Lawrence Erlbaum.

Lauber, E. J. (1995). Executive control of task switching operations. Unpublished doctoral dissertation, University of Michigan, Ann Arbor, MI.

Martin-Emerson, R., & Wickens, C. D. (1992). The vertical visual field and implications for the head-up display. Proceedings of the 36th Annual Symposium of the Human Factors Society. Santa Monica, CA: Human Factors Society.

Meiran, N. (1996). Reconfiguration of processing mode prior to task performance. Journal of Experimental Psychology: Learning, Memory, and Cognition, 22, 1423-1442.

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

Meyer, D. E., & Kieras, D. E. (1997b). EPIC -- A computational theory of executive cognitive processes and multiple-task performance: Part 2. Accounts of psychological refractory-period phenomena. Psychological Review, 104, 749-791.

Meyer, D. E., & Kieras, D. E. (1999). Précis 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.

Meyer, D. E., Kieras, D. E., Lauber, E., Schumacher, E., Glass, J., Zurbriggen, E., Gmeindl, L., & Apfelblat, D. (1995). Adaptive executive control: Flexible multiple-task performance without pervasive immutable response-selection bottlenecks. Acta Psychologica, 90, 163-190.

Neisser, U. (1967). Cognitive psychology. New York: Appleton Century Crofts.

Newell, A. (1980). Harpy, production systems, and human cognition. In R. A. Cole (Ed.), Perception and production of fluent speech (pp. 289-380). Hillsdale, NJ: Lawrence Erlbaum.

Newell, A. (1990). Unified theories of cognition. Cambridge, MA: Harvard University Press.

Norman, D. A. & Shallice, T. (1986). Attention to action: Willed and automatic control of behavior. In R. J. Davidson, G. E. Schwartz, & D. Shapiro (Eds.), Consciousness and self-regulation (Vol. 4, pp. 1-18). New York: Plenum Press.

Pashler, H. (1994). Dual-task interference in simple tasks: Data and theory. Psychological Bulletin, 116, 220-244.

Rogers, R., & Monsell, S. (1995). Costs of a predictable switch between simple cognitive tasks. Journal of Experimental Psychology: General, 124, 207-231.

Rubinstein, J. S., Meyer, D. E., & Evans, J. (1995). Executive control of cognitive processes in task switching. Manuscript submitted for publication.

Schumacher, E. H., Lauber, E. J., Glass, J. M. B., Zurbriggen, E. L., Gmeindl, L., Kieras, D. E., & Meyer, D. E. (1999). Concurrent response-selection processes in dual-task performance: Evidence for adaptive executive control of task scheduling. Journal of Experimental Psychology: Human Perception and Performance, 25, 791-814.

Stallings, W. (1998). Operating systems: Internals and design principles (3rd Edition). Upper Saddle River, NJ: Prentice Hall.

Theios, J. (1973). Reaction time measurements in the study of memory processes: Theory and data. In G. H. Bower (Ed.), The psychology of learning and motivation (Vol. 7, pp. 43-85). New York: Academic Press.

Tucker, A. (Ed.) (1997). The computer science and engineering handbook. Boca Raton, FL: CRC Inc.