HUMAN-COMPUTER INTERACTION, 1997, Volume 12, pp. 391-438

An Overview of the EPIC Architecture for Cognition and Performance With Application to Human–Computer Interaction

David E. Kieras and **David E. Meyer**
University of Michigan

ABSTRACT

EPIC (Executive Process–Interactive Control) is a cognitive architecture especially suited for modeling human multimodal and multiple-task performance. The EPIC architecture includes peripheral sensory-motor processors surrounding a production-rule cognitive processor and is being used to construct precise computational models for a variety of human–computer interaction situations. We briefly describe some of these models to demonstrate how EPIC clarifies basic properties of human performance and provides usefully precise accounts of performance speed.

David E. Kieras is a cognitive psychologist and computer scientist whose speciality is computational modeling of human cognition and performance with an emphasis on application to human–computer interaction and other system design problems; he is an Associate Professor in the Department of Electrical Engineering and Computer Science at the University of Michigan. **David E. Meyer** is a mathematical and experimental cognitive psychology scientist with interests in human cognition and action; he is a Professor in the Cognition and Perception Program of the Department of Psychology at the University of Michigan.

CONTENTS

1. INTRODUCTION

Cognitive Architectures. A cognitive architecture is a theoretical structure and set of mechanisms for human cognition, within which models for specific tasks and phenomena can be constructed. Since the proposals of Anderson (1976) and Laird, Rosenbloom, and Newell (1986), cognitive architectures have become recognized as the fundamental theoretical ap-

proach in cognitive psychology. An architecture proposal is a synthesis of theoretical concepts that attempts to subsume a variety of specific models and mechanisms into a single coherent whole. When the architecture is represented computationally, its implications and applicability can be easily and rigorously explored and tested. Progress in rigorous cognitive theory requires the development of more comprehensive and accurate computational cognitive architectures.

Cognitive Architecture and Human–Computer Interaction (HCI). The significance of cognitive architectures for the more practical concerns of HCI and user-interface design lies in two areas. First, to the extent that the key phenomena of relevance to HCI can be captured in an architecture, the architecture acts as a codified theoretical summary of the phenomena. Such a codification can then be learned and applied by HCI researchers and practitioners much more easily than the traditional approach of studying and attempting to apply a vast collection of isolated phenomena, individual experimental results, and small-scale models. Second, if the architecture supports constructing models for tasks easily enough and makes accurate enough predictions of task performance, the cognitive architecture then provides a foundation for engineering models for evaluating user-interface designs early in the development process, which can provide valuable usability information in addition to traditional user testing methods. For example, as discussed by John and Kieras (1996), the various extant members of the GOMS family of engineering models are based on some simple cognitive architectures but have been useful in interface design and evaluation. By developing more sophisticated architectures that have predictive power in more complex situations, we should be able to develop more accurate and more comprehensive engineering models to aid in HCI design.

The EPIC (Executive Process–Interactive Control) Architecture. This article provides an overview of the EPIC architecture being developed by Kieras and Meyer for modeling human cognition and performance (Kieras, Wood, & Meyer, 1997; Meyer & Kieras, 1997a, 1997b). EPIC is similar in spirit to the Model Human Processor (MHP; Card, Moran, & Newell, 1983), but EPIC incorporates many recent theoretical and empirical results about human performance in the form of a software framework for computer simulation modeling. Using EPIC, a model can be constructed that represents the general procedures required to perform a complex multimodal task as a set of production rules. When the model is supplied with the external stimuli for a specific task, it will then execute the procedures in whatever way the task requires, thus simulating a human's performing the task and generating the predicted actions in simulated real time. EPIC is an architecture for constructing models of

performance. It is not yet a learning system and so has no mechanisms for learning how to perform a task. Rather, the purpose of EPIC is to represent in detail the perceptual, motor, and cognitive constraints on the human ability to perform tasks.

Like most cognitive architectures, EPIC was not developed primarily for addressing HCI problems but is a larger scientific endeavor to represent important theoretical concepts of human intelligence or abilities. However, because HCI is a subset of human performance, a good proposal for a cognitive architecture should allow one to analyze and compare interface designs and then recommend and evaluate improvements. At this time, EPIC is mainly useful as a research system for exploring human performance limitations that determine the effects of a particular interface design, both at low levels of specific interaction techniques and at high levels of systems that support complex task performance in multimodal time-stressed domains. In the future, it should be possible to develop EPIC-based design analysis techniques that can be routinely applied in system design.

Organization of This Article. In this article, we describe the rationale for the development of EPIC, summarize the architecture, and discuss some important general modeling issues. Then, we illustrate EPIC's contributions to HCI with a series of application vignettes—brief examples showing how EPIC can be used in both predictive and explanatory modes to address both elementary aspects of interface design and complex phenomena related to human interaction with semi-automated systems. We use these vignettes because our goal in this article is not to exhaustively explore one application of the EPIC architecture but rather to present and justify the architecture by illustrating its wide applicability.

2. THE EPIC (EXECUTIVE PROCESS–INTERACTIVE CONTROL) ARCHITECTURE

2.1. Fundamental Motivations for Developing EPIC

Embodied Cognition

Historically, the proposals for computational models of human cognition both in cognitive psychology and artificial intelligence have tended to emphasize the purely cognitive aspects of the human system, finessing the details of how the human perceives the environment or acts upon it. Such an approach can be seriously misleading. For example, human visual capacity to detect and recognize objects is not uniform but varies with the distance on the retina from the fovea. The retina can be oriented through a motor system, the oculomotor mechanism, to control what part of the

visual environment can be accessed in detail. Thus, in many tasks, the availability of the stimulus depends on the details of when, how, or whether the eye is oriented toward the stimulus. Likewise, conventional cognitive theory has tended to assume that, after the human decides to act, there are no fundamental problems in carrying out the intended action. However, the human motor system is quite complex in its own right and interacts strongly with the cognitive and perceptual systems (Rosenbaum, 1991). For example, different movements can take a substantial amount of time to execute, and this time can depend heavily on the details of the required movements and the history of previous, possibly interfering, movements. Furthermore, response execution can make demands on the perceptual system as well, the most extreme being the need for vision in aimed movements; thus, making a response may interfere with collecting the visual information needed for the next part of the task.

More recently, some of the computational cognitive architectures have begun to be *embodied,* to include some of the constraints imposed by the perceptual-motor system. For example, Newell (1990) explicitly included such constraints in his outline of a cognitive architecture, and some effort has been made to include perceptual-motor mechanisms in more recent work with the Soar architecture (Laird et al., 1986) since its introduction. Likewise, the ACT-R architecture proposed by Anderson (1993) has begun to include some perceptual-motor mechanisms. Our first goal in developing EPIC is not only to incorporate key facts about human perceptual and motor constraints into a theory of human cognition but also to give the perceptual and motor mechanisms equal status with cognition in accounting for human performance. Thus, EPIC's production-rule cognitive processor is surrounded by perceptual and motor processors, whose time course of processing is represented in some detail based on the current human performance research literature. How long it takes an EPIC model to do a task depends intimately on how EPIC's eyes, perceptual mechanisms, and effectors are used in the task. At this level of detail, the interactions between processors during task execution can be remarkably subtle, so the representation of task timing in a computational simulation model is critical to understanding human performance.

Computational Models of Both Performance and Cognition

Our second motivation for developing EPIC is a corollary of the first—to more fully extend the current cognitive architecture computational modeling paradigm to the field of human attention and performance. Although some computationally realized models of human performance have been available for years (e.g., HOS, SAINT; see Elkind, Card, Hochberg, & Huey, 1989; McMillan et al., 1989), these models generally do not have the form of cognitive architectures so much as being analytic tools

for practical system design—a form of the engineering models discussed by John and Kieras (1996). The field of human performance itself has suffered from a lack of computational modeling, although historically, it is one of the most extensively researched and most practically useful of psychological fields. Most research on human performance has been conducted at the level of qualitative interpretation of empirical results and verbally expressed and evaluated theory. One symptom of this lack of the detailed rigor available with computational models is that key issues concerning the fundamentals of human information processing have remained unresolved for many years, such as the status of the single-channel-versus-multiple-resource debate discussed by Meyer and Kieras (1997a). Thus, another goal of developing EPIC has been to advance the state of psychological theory in a theoretically underdeveloped area.

An important benefit of working in the human performance domain is that this empirical literature abounds in quantitative data that typically have a precision and detail not found in more purely cognitive task domains. Unlike many traditional cognitive modeling efforts, ours is committed to obtaining detailed and quantitatively accurate fits to empirical data in a variety of performance task domains. One reason for making this effort is that for models of performance to be practically useful in system design problems, they must be reasonably accurate. A more fundamental reason is that trying to match detailed data with quantitative accuracy serves as a powerful constraint in constructing models for phenomena. That is, a key function of a cognitive architecture is to provide some theoretical constraint on the possible models (see Newell, 1990). Trying to match detailed quantitative data acts as a further constraint, further reducing the number of arbitrary decisions to be made in constructing an EPIC model for a task. The constraints imposed by the combination of the detailed quantitative effects in the empirical data, the task structure, and the fixed architecture mean that there are relatively few "degrees of freedom" in constructing a model that fits well (see Meyer & Kieras, 1997b, for further discussion).

The Executive Process and Multiple-Task Performance

A third motivation for developing EPIC has been to explore the mechanism and the role of executive processes, which control and supervise other cognitive processes, analogous to the "supervisor" in a computer operating system (see Meyer & Kieras, 1997a). Theorists of human performance have presented various proposals about the nature of the executive process. Unfortunately, these proposals have usually lacked either a coherent theoretical basis or a computational representation. These theorists have also often proposed that the executive process is implemented via some type of special mechanism that sits outside or above the regular cognitive system and presumably has its own principles of operation.

However, since proper supervision of behavior is simply a form of skill, we have sought to represent the executive process in the same way as other forms of skill—just as the supervisory component of a computer operating system is just another computer program.

A good approach to understanding both the nature of the executive process and the details of the human cognitive architecture is to understand multiple-task performance. In multiple-task situations, the human has to perform two or more tasks simultaneously; the overall task situation can be subdivided into two or more tasks, each of which can be meaningfully performed in isolation (one is not a logical subtask of the other), and the tasks are performed over the same period of time. A good example of a multiple-task situation occurs in an airplane cockpit; a pilot may need to simultaneously pilot the aircraft and track an enemy target on a radar display. The main problem confronting the human is to execute the independent tasks in a coordinated fashion that meets some constraints on overall performance, such as giving one task priority over the other.

The literature on multiple-task performance is extensive and is not summarized here; for a review, see Gopher and Donchin (1986) and Meyer and Kieras (1997a). Of course, human information processing is limited in capacity, and a single-channel bottleneck has traditionally been assumed. Nevertheless, humans can do multiple tasks, sometimes impressively well, and their ability to do so depends strongly on the specific combinations of tasks involved. The *multiple-resource theory* is an attempt to summarize these dependencies, which pose a fundamental theoretical dilemma about how to reconcile the complex patterns of people's multitasking abilities with some notion that the overall capacity of the human system is limited. With EPIC, however, we do not make the assumption that central capacity for cognitive processing is limited. Such an assumption is traditional but lacks both empirical and metatheoretical justification. In contrast, we assume that limitations on human ability are all structural; that is, performance of tasks may be limited by constraints on peripheral perceptual and motor mechanisms or by limited verbal working memory capacity, rather than by a pervasive limit on cognitive-processing capacity. The executive strategy has the responsibility of meeting the performance requirements of the tasks in spite of these structural limitations.

To meet performance goals, the executive process must coordinate the use of the perceptual, cognitive, and motor resources of the system so that the tasks can be conducted with the proper relative priority and speed. Multiple-task situations stress human capabilities very seriously, and so the observed patterns of behavior provide clear insights into the abilities and limitations of the human information-processing system architecture. Yet, despite the practical importance of multiple-task performance, the empirical and theoretical understanding of multiple-task performance has been quite limited. Nevertheless, EPIC models have been successful in accounting with

unprecedented accuracy for performance in laboratory versions of multiple-task situations (Kieras & Meyer, 1995; Meyer & Kieras, 1997a, 1997b). In addition, understanding the allocation of resources in multiple-task situations contributes to understanding single-task situations; to maximize performance, the executive process must allocate processing resources to different parts of the single task in a properly coordinated fashion.

2.2. Description of the EPIC Architecture

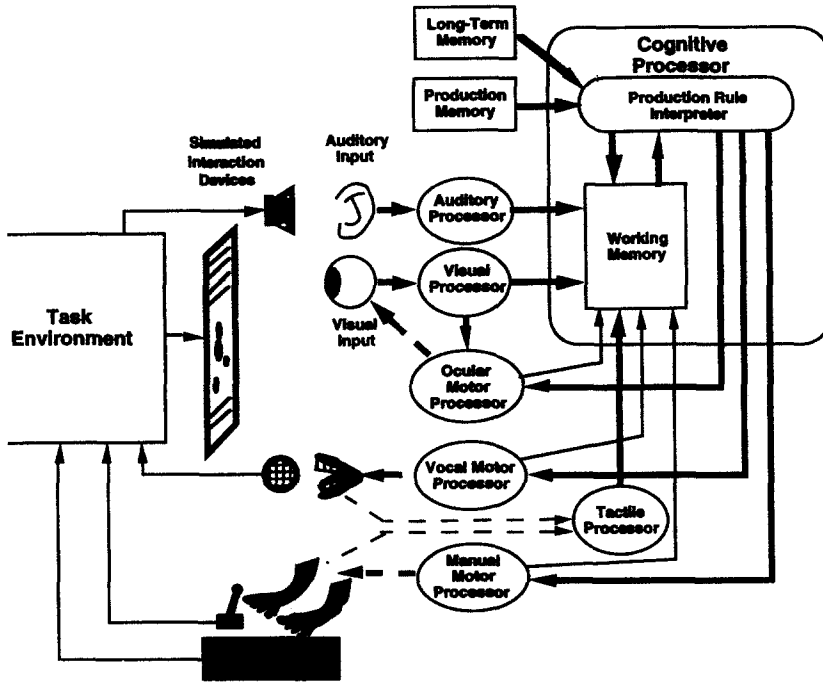
Overview

Figure 1 shows the overall structure of processors and memories in the EPIC architecture. At this level, although EPIC bears a superficial resemblance to earlier frameworks for human information processing, it incorporates a new synthesis of theoretical concepts and empirical results and so is more comprehensive and more detailed than earlier proposals for human performance modeling (e.g., MHP, HOS, SAINT; see McMillan et al., 1989). EPIC is designed to explicitly couple detailed mechanisms for basic information processing and perceptual-motor activity with a cognitive analysis of procedural skill—namely, that represented by production-system models such as CCT (Bovair, Kieras, & Polson, 1990), ACT-R (Anderson, 1993), and Soar (Laird et al., 1986). Thus, EPIC has a production-rule cognitive processor surrounded by perceptual-motor peripherals; applying EPIC to a task situation requires specifying both the production-rule programming for the cognitive processor and the relevant perceptual and motor-processing parameters. EPIC computational task models are *generative,* in that the production rules supply general procedural knowledge of the task, and, when EPIC interacts with a simulated task environment, the EPIC model generates the specific sequence of serial and parallel human actions required to perform the specific tasks. Rather than reflecting specific task scenarios, the task analysis reflected in the model is general to a class of tasks.

The software for constructing EPIC models is currently implemented in Common LISP, with models typically developed in Macintosh Common Lisp, and then simulation production runs are executed under Franz Allegro Common LISP on a fast Unix workstation. All of the models described or cited in this article have actually been implemented and run to produce the claimed predicted results. Although the simulation software is available to interested researchers, as is a detailed technical description of the architecture,[1] we have focused exclusively on developing the architecture and modeling important tasks that stress the scientific accuracy of the architecture. Thus, at this time, EPIC is not packaged in a

1. Available via anonymous ftp at ftp://ftp.eecs.umich.edu/people/kieras/EPICarch.ps.

Figure 1. **Overall structure of the EPIC architecture simulation system. Task performance is simulated by having the EPIC model for a simulated human (on the right) interact with a simulated task environment (on the left) via a simulated interface between sensory and motor organs and interaction devices. Information flow paths are solid lines; mechanical control or connections are dashed lines. The processors run independently and in parallel both with each other and with the task environment module.**



"user-friendly" manner; full-fledged LISP programming expertise is required to use the simulation package, and there is no introductory tutorial or user's manual.

The EPIC software framework includes not only the modules for simulating a human but also facilities for simulating the interaction of the human with an external system such as a computer. Figure 1 shows a simulated task environment (on the left) and a simulated human as described by the EPIC architecture (on the right), with objects such as simulated screen items and simulated keys making up the physical interface between them. The task environment module assigns physical locations to the interface objects and generates simulated visual events and sounds that the computer or other entities in the environment produce in response to the simulated human's behavior. Having a separate environment simulation module greatly simplifies the programming of a complete simulation and helps enforce the generality of the procedural knowledge

represented in the EPIC model. That is, the task environment module is driven by a task instance description that consists only of the sequence and timing of events external to the human user, and the simulated user must deal with whatever happens in the simulated task environment.

With regard to the EPIC architecture itself (as shown in Figure 1), there is a conventional flow of information from sense organs, through perceptual processors, to a cognitive processor (consisting of a production rule interpreter and a working memory), and finally to motor processors that control effector organs. EPIC goes beyond MHP by specifying separate perceptual processors with distinct processing-time characteristics for each sensory modality and separate motor processors for vocal, manual, and oculomotor (eye) movements. There are feedback pathways from the motor processors, as well as tactile feedback from the effectors, which are important in coordinating multiple tasks. The declarative–procedural knowledge distinction of the "ACT-class" cognitive architectures (e.g., Anderson, 1976) is represented in the form of separate permanent memories for production rules and declarative information. Working memory (WM) contains all of the temporary information needed for and manipulated by the production rules, including control items such as task goals and sequencing indices, and also conventional WM items, such as representations of sensory inputs. WM has separate partitions for different types of information, such as auditory WM, visual WM, the *control store,* and so forth. The structure of WM and the properties of the processors are described in more detail later. When numeric values are given for various time parameters, they are labeled as either *standard values* that we assume are fixed in all applications of the architecture or *typical values* that may vary depending on the properties of a specific task situation. Standard values are based on our reading of the human performance literature; although they may be revised and refined, they are supposed to hold across all applications of the architecture. The nonstandard parameter values need to be estimated to model a specific task, but we hope that, with additional modeling experience and focused empirical studies, collections of typical parameter values will become available for use in constructing new models.

Perceptual Processors

A single stimulus input to a perceptual processor can produce multiple outputs to be deposited in WM at different times. The perceptual processors in EPIC are simple "pipelines," in that an input produces an output at a certain later time, with no "moving window" time-integration effect as assumed by MHP. The tactile perceptual processor handles movement feedback from effector organs; this feedback can be important in coordinating multiple tasks (Meyer & Kieras, 1997a, 1997b) but is not elaborated further here.

Visual Processor. EPIC's model of the eye includes a retina that determines what kind of sensory information is available about visual objects in the environment based on the distance (in visual angle) on the retina between the object and center of the fovea. EPIC's current highly simplified model of the retina contains three zones, each with a standard radius: the fovea (1°), the parafovea (10°), and the periphery (60°). Certain information (e.g., contents of character strings) might be available only in the fovea, whereas cruder information (e.g., whether an area of the screen is filled with characters) is available in the parafovea. Only severely limited information (e.g., location of objects; whether an object has just appeared) is available in peripheral vision. Of course, the exact availability of visual information in different areas of the retina depends on the specific physical properties of the stimulus. For example, a large isolated character might be discriminable many degrees away from the fovea, while reading words embedded in text displayed in small type would require that the words be in or very close to the fovea. Unfortunately, the human performance literature does not appear to contain a body of well-parameterized results on the properties of nonfoveal vision, meaning that the exact time and availability parameters of visual stimuli must be estimated for new task-specific models.

In EPIC's visual working memory, the visual perceptual processor maintains a representation of which objects are visible and what their properties are. Visual working memory is "slaved" to the visual situation; it is kept up-to-date as objects appear, disappear, change color, and so forth or as eye movements or object movements change what visual properties are available from the retina. In response to visual events, the visual processor can produce multiple outputs with different timings. When an object appears, the first output is a representation that a perceptual event has been detected (standard delay = 50 msec), followed later by a representation of sensory properties (e.g., shape; standard delay = 100 msec) and still later by the results of pattern recognition, which might be task-specific (e.g., a particular shape represents a left-pointing arrow; typical delay = 250 msec).

Auditory Processor. The auditory perceptual processor accepts auditory input and then outputs to working memory representations of auditory events (e.g., speech) that disappear after a time. For example, a short tone signal produces, first, an item corresponding to the onset of the tone (standard delay = 50 msec); then, later, an item corresponding to a discriminated frequency of the tone (typical delay = 250 msec); and, last, an offset item (standard delay = 50 msec). For simplicity, such items simply disappear from memory after a fixed time (typical delay = 4 sec).

Speech input is represented as items for single words in auditory working memory. The auditory perceptual processor requires a certain time to

recognize input words (typical delay = 150 msec after acoustic stimuli are present) and produces representations of them in auditory working memory. These items then disappear after a time, the same as other auditory input. To represent the sequential order of the speech input, the items contain arbitrary symbolic tags for previous item and next item that link the items in sequence. Thus, a speech input word carries a certain next-tag value, and the next word in the sequence is the item that contains the same tag value as its previous tag. Using these tags, a set of production rules can step through the auditory working memory items for a series of spoken words, processing them one at a time. For example, one of the models described in this article processes a spoken telephone billing number by retrieving the recognized code for each digit in the tag-chained sequence and using it to specify a key press action. Available empirical literature on auditory perception lacks comprehensive results, so many auditory perceptual parameters must be estimated during task-specific model construction.

Cognitive Processor

Production Rules and Cycle Time. The cognitive processor is programmed in terms of production rules, and so an EPIC model for a task must include a set of production rules that specify what actions in what situations are performed to do the task. Example production rules for the models described in this article are presented later. EPIC uses the parsimonious production system (PPS) interpreter, which is especially suited to task modeling work (Bovair et al., 1990). PPS rules have the format (<rule-name> IF <condition> THEN <actions>). The rule condition can test only the contents of the production-system working memory. The rule actions can add and remove items from working memory or send a command to a motor processor.

The cognitive processor operates cyclically. At the beginning of each cycle, the contents of working memory are updated with the output from perceptual processors and the previous cycle's modifications; at the end of each cycle, the contents of the production-system working memory are updated, and commands are sent to the motor processors. The mean duration of a cycle is a standard 50 msec. The cognitive-processor cycles are not synchronized with external stimulus-and-response events. Inputs from the perceptual processors are accessed only intermittently, when the production-system working memory is updated at the start of each cycle. Any input that arrives during the course of a cycle must therefore wait temporarily for service until the next cycle begins. This is consistent with a variety of phenomena, such as the apparent temporal granularity of perceived stimulus successiveness (Kristofferson, 1967). EPIC also can run in a mode in which the cycle duration is stochastic, with a standard mean

value of 50 msec and all other time parameters scaled to this stochastic value; the variance of the stochastic distribution of cycle time is chosen to produce a coefficient of variation of about 20% for a simple reaction time, corresponding to the typical observed value.

Cognitive Parallelism. Most traditional production-system architectures require that only one production rule can be fired at a time and that only its actions will be executed. Should more than one rule have matching conditions, some kind of conflict-resolution mechanism is required to choose which rule to fire. Soar (Laird et al., 1986) is perhaps the most complex, in that the production rules only propose operators to apply, and so many rules can be fired at once, and then a separate process decides which single candidate operator to apply. However, PPS has a radical and very simple policy: On each cognitive-processor cycle, PPS will fire all rules whose conditions match the contents of working memory and will execute all of their actions. Thus, EPIC models may have true parallel cognitive processing at the production-rule level; multiple "threads" or processes can be represented simply as sets of rules that happen to run simultaneously.

The multiprocessing ability of the cognitive processor, together with the parallel operation of all the perceptual-motor processors, means that EPIC models for multiple-task performance do not incorporate a gratuitous assumption of limited central-processing capacity or of a central-processing "bottleneck." It is critical to be clear on exactly what is or is not being claimed. Although EPIC has no built-in limit on how many strategies or processes the cognitive processor can be executing simultaneously, the rate of execution is limited, the perceptual-motor mechanisms are of course limited, and any memory mechanisms involved in the task are limited. For example, the reason why a person cannot perform two long divisions in his or her head simultaneously is that a limited structural resource is involved—namely, verbal short-term memory. For example, the eyes can fixate on only one place at a time, and the two hands are bottlenecked through a single processor. Thus, a task that demands manual responses to visual stimuli distributed over a wide space will result in severely limited human performance, even if the purely cognitive demands are trivial. In contrast, people can, and often do, perform multiple cognitive tasks simultaneously, such as collecting one's slides while answering questions at the end of a talk, as long as the strategies are otherwise compatible.

Omitting a central-capacity limit or bottleneck might seem to be a radical recasting of conventional cognitive theory, but this claim is actually consistent with a long-standing line of empirical and theoretical discussion in the human performance field that challenges the traditional assumption of limited central capacity (see Meyer & Kieras, 1997a). Our own detailed quantitative modeling of a variety of multiple-task data (Meyer & Kieras, 1997b) shows that EPIC's assumptions about the nature

of cognitive and perceptual-motor limitations are quite consistent with a large variety of empirical data.

This decision about the nature of human limitations is also a matter of scientific tactics: Our theoretical strategy has been to make some radical simplifying assumptions and then explore their consequences through modeling, complicating the architecture only as required. Thus, we start with the known limitations of human memory and perceptual-motor mechanisms and adopt less apparent limitations only if the data compel us. Thus far, our simple and radical set of assumptions about the nature of multiple-task processing limitations has held up well.

Working Memory. EPIC's production-system working memory is in effect partitioned into several working memories.[2] Visual, auditory, and tactile memory contain the current information produced by the corresponding perceptual processors. The timing and duration of these forms of working memory are described earlier. Motor working memory contains information about the current state of the motor processors, such as whether a hand movement is in progress. This information is updated on every cycle.

Two other forms of working memory deserve special note. These forms are amodal, in that they contain information not directly derived from sensory or motor mechanisms. One amodal working memory is the *control store,* which contains items that represent the current goals and the current steps within the procedures for accomplishing the goals. An important feature of PPS is that control information is simply another type of working memory item and so can be manipulated by rule actions; this is critical for modeling multiple-task performance, in that production rules for an executive process can control subprocesses by manipulating the control store.

The second amodal working memory, simply termed "general WM," can be used to store miscellaneous task information. At this time, EPIC does not include assumptions about the decay, capacity, and representational properties of general working memory. Our research strategy in developing EPIC has been to see what constraints on the nature of this general WM are required to model task performance in detail rather than to follow the customary strategy in cognitive modeling of assuming these

2. EPIC's working memory structure is not "hard-wired" into PPS. PPS actually has only a single working memory, which could more clearly be termed the *database* for the production rules. PPS can be used as a multiple-memory system simply by following a convention such as the first term in a database item indicating the "type" of memory item, as in the examples later in this article. Likewise, the format of items in working memory or the required contents of rule conditions are not fixed in the architecture; we have preferred to develop such restrictions through modeling experience rather than prematurely prescribe them in the architecture.

constraints in advance. Such capacity and loss assumptions for these memory systems do not seem to be required to account for the time course of performance in tasks modeled in EPIC thus far; other limitations determined by the perceptual and motor systems appear to dominate performance. These latter substantial but underappreciated limitations would have been obscured by gratuitous assumptions about central-processing capacity or working memory (see Meyer & Kieras, 1997a, 1997b, for more discussion). For similar reasons, at this time EPIC assumes that information is not lost from the control store, and there is no limit on the capacity of the control store. Research is underway to explore how loss or corruption of information in EPIC's working memories might account for the occurrence and properties of human errors during task performance.

Motor Processors

The EPIC motor processors are much more elaborate than those in the MHP, producing a variety of simulated movements of different effector organs and taking varying amounts of time to do so. As shown in Figure 1, there are separate processors for the hands, eyes, and vocal organs, and all can be active simultaneously. The cognitive processor sends a command to a motor processor that consists of a symbolic name for the type of desired movement and any relevant parameters, and the motor processor then produces a simulated movement with the proper time characteristics. The various processors have similar structures but different timing properties and capabilities based on the current human performance literature in motor control (Rosenbaum, 1991). The manual motor processor has many movement forms, or styles, but the two hands are bottlenecked through a single manual processor, and so normally can be operated either one at a time or synchronized with each other. The oculomotor processor generates eye movements either upon cognitive command or in response to certain visual events. The vocal motor processor produces a sequence of simulated speech sounds given a symbol for the desired utterance.

Movement Preparation and Execution. The various motor processors represent movements and movement generation in the same basic way. Current research on movement control (Rosenbaum, 1980, 1991) suggests that movements are specified in terms of movement features, and the time to produce a movement depends on its feature structure as well as its mechanical properties.

The overall time to complete a movement can be divided into a *preparation* phase and an *execution* phase. The preparation phase begins when the motor processor receives the command from the cognitive processor. The motor processor recodes the name of the commanded movement into a set of movement features, whose values depend on the

style and characteristics of the movement, and then generates the features, taking a standard 50 msec for each one. The time to generate the features depends on how many features can be reused from the previous movements (repeated movements can be initiated sooner) and how many features have been generated in advance. After the features are prepared, the execution phase begins with an additional standard delay of 50 msec to initiate the movement followed by the actual physical movement. The time to physically execute the movement depends on its mechanical properties both in terms of which effector organ is involved (e.g., eye vs. hand) and type of movement to be made (e.g., one-finger flexion to press a button under the finger vs. a pointing motion with a mouse).

The movement features remain in the motor processor's memory, so future movements that share the same features can be performed more rapidly. However, there are limits on whether features can be reused; for example, if a new movement is different in style from the previous movement, all of its features must be generated anew. Also, if the task permits the movement to be anticipated, the cognitive processor can command the motor processor to prepare the movement in advance by generating all of the required features and saving them in motor memory. Then, when it is time to make the movement, only the initiation time is required to commence the mechanical execution of the movement.

Finally, a motor processor can prepare the features for only one movement at a time and will reject any subsequent commands received during the preparation phase, but the preparation for a new movement can be done in parallel with the physical execution of a previously commanded movement. Once prepared, the movement features are saved in motor memory until the previous execution is complete, and the new movement is then initiated. The cognitive-processor production rules can exploit this capability by sending a motor processor a new movement command as soon as it is ready to begin preparing the features for the new movement. The result can be a series of very rapid movements whose total time is little more than the sum of their initiation and mechanical execution times.

An Example of Motor-Processor Operation. To strike a key using a one-finger peck movement style (like that used in "hunt-and-peck" typing), the cognitive processor commands the manual motor processor to perform a peck movement with a finger (e.g., the right index) to a specified object in the physical environment (the key). This movement style involves five features: peck style, hand, finger, direction, and extent of motion, which is the distance between the current location of the designated finger and the location of the target object. If a previous movement was also a peck movement with the same hand and finger, only the direction and extent might have to be generated anew. If the movement is also similar in direction and extent to the previous movement, then all of

the features could be reused; none would have to be generated anew. After the features are generated, the movement is initiated. The time required to physically execute the movement to the target is given by Welford's form of Fitts' law (see Card et al., 1983, chap. 2), with a standard minimum execution time of 100 msec, reflecting that, for small movements to large targets, there is a physiologically determined lower bound on the actual duration of a muscular movement. After the simulated finger hits the key, it is left in the location above the key to await the next movement.

Manual Motor Processor. EPIC's manual motor processor represents several movement styles, including punching individual keys or buttons already known to be below the finger, pecking keys that may require some horizontal motion, posing the entire hand at a specified location, pattering two-finger movements one after the other, poking at an object (e.g., on a touch screen), pointing at an object with a mouse, and plying a control (e.g., a joystick) to position a cursor onto an object. Each style of movement has a particular feature structure and an execution-time function that specifies how long the mechanical movement takes to actuate the device in the task environment.

Vocal Motor Processor. EPIC's vocal motor processor is not very elaborated at this time; it is based on the minimal facilities needed to model certain dual-task situations (see Meyer & Kieras, 1997a, 1997b). A more complete version of the vocal motor processor would be able to produce extended utterances of variable content, taking into account that the sequential nature of speech means that movements can be prepared on the fly during the ongoing speech. The current version of EPIC assumes that simple fixed utterances can be designated with a single symbol and require only the preparation of two features before execution begins. The actual production of the sound is assumed to be delayed by about 100 msec after initiation and continues for a time determined by the number of syllables in the words. Further development of the vocal motor processor is planned for the future.

Oculomotor Processor. EPIC's eye movements are produced in two modes, voluntary and involuntary (reflexive). The cognitive processor commands voluntary eye movements, which are saccades to a designated object. A saccade requires generation of up to two features, the direction and extent of the movement from the current eye position to the target object. Execution of the saccade currently is estimated to require a standard 4 msec per degree of visual angle. The oculomotor processor also makes involuntary eye movements, either saccades or small smooth adjustments in response to the visual situation (hence the arrow between the visual perceptual processor and the oculomotor processor in Figure 1). A

sudden onset (appearance) of an object can trigger an involuntary saccade. The fovea being somewhat off-center on an object will produce a "centering" movement, which will automatically help zero in on an object after a saccade. Also, the eye will automatically follow a slowly moving object using smooth movements and occasional small saccades (cf. Hallett, 1986). In some of the tasks presented later in this article, EPIC can follow moving objects with a mixture of voluntary and involuntary eye movements. The partial autonomy of the oculomotor processor permits the cognitive processor to choose an object to examine, command that the eye be moved to it, and then leave the details of keeping it centered on the fovea to the oculomotor processor.

2.3. Constructing Models in EPIC

Constraints on Model Construction

Fixed Architecture, Variable Strategies. The presentation of any modeling approach should document what aspects or parameters of the modeling framework are fixed and are thus supposed to generalize across applications and what aspects or parameters have to be adjusted to fit the data or estimated from data specific to the situation being modeled. In EPIC, the most important fixed aspect is the connections and mechanisms of the EPIC processors, which are supposed to apply without modification across task domains. Our models are thus built by "programming" a fixed and comprehensive architecture with a task strategy expressed in production rules executed by the cognitive processor. We always attempt to explain phenomena in terms of task-specific cognitive strategies before changing the architecture itself. Thus, the key aspect of EPIC that is free to vary in a task-specific way is the task-specific production-rule programming, which is constrained to some extent because it must be written to execute the task correctly and reasonably efficiently.

Fixed and Free Parameters. The fixed parameters are most time parameters in the processors and the feature structure of the motor processors for individual styles of movement. The model properties and parameters that are then free to vary from model to model or task to task are, first, the task-specific sensory availabilities and perceptual encoding types and times involved in the task (constrained to be similar and constant over similar perceptual events) and second, the styles of movement used to control the device (e.g., touch-typing vs. visually guided pecking), if they are not constrained by the task.

Model Inputs and Outputs. Similarly, any modeling approach should document what information the model builder has to supply in order to

construct the model and what information the constructed model, will then produce in return for the supplied information. To construct an EPIC model, the model builder has to supply the information corresponding to the three free parameters just described—namely:

1. A production-rule representation of the task procedures.
2. Task-specific sensory availabilities and perceptual-processor encodings and timings.
3. Any movement styles not determined by the task requirements.

In addition, the model builder must supply:

4. The simulated task environment, which includes the physical locations and characteristics of relevant objects external to the human.
5. A set of task instances whose execution time is of interest; these instances must specify only environmental events and their timing and are used to control only the environment module of the simulation.

In return for these inputs, the EPIC model will interact with the simulated task environment, generating the predicted sequences of simulated human actions required by each task instance and the predicted time of occurrence of each action. If the production rules were written to describe general procedural knowledge of how to perform the task, these predictions can be generated for any task instance subsumed by these general procedures.

Modeling Multiple Tasks and Executive Processes

The Executive Process. Some theorists of multiple-task performance postulate an executive-control process that coordinates the separate multiple tasks (e.g., Norman & Shallice, 1986). We do likewise, but a key feature of our approach is that the executive-control process is just another set of production rules. These rules can control other task processes by manipulating information in the control-store partition of the production-system working memory. For example, we assume that each task is represented by a set of production rules that have the task goal appearing in their conditions, and so an executive-process rule can suspend a task by removing its governing goal from the control store and then cause it to resume operation by reinserting the goal. Also, the executive process can cause a task to follow a different strategy by placing in general WM an item for which task rules test, thus enabling one set of rules and disabling another. In addition, the executive process may control sensory and motor peripherals directly (e.g., moving the eye fixation from one point to another) in order to

allocate these resources between two tasks. Thus, rather than postulating an executive control mechanism that is somehow different in kind than other cognitive mechanisms, EPIC has a uniform mechanism for the control of behavior both at the executive level and at the detailed level of individual task actions. As a corollary, learning how to coordinate multiple tasks is simply learning another (possibly difficult) skill, as has been proposed by some recent investigators (e.g., Gopher, 1993).

Modeling Elementary Multiple Tasks. Our first work with EPIC focused on the simplest and most heavily studied dual-task situation in the research literature, the so-called psychological refractory period (PRP) procedure. The PRP procedure consists of two temporally overlapping choice reaction-time tasks; the subject is instructed to make the response for the first task before making the response for the second task. The primary measure of interest is the reaction time (RT) for the second task, which may be affected by the temporal spacing between the two task stimuli. The basic empirical result is that the second response is substantially delayed as the spacing between the two stimuli decreases. The conventional interpretation of this effect (the PRP effect) is that the human has a central response-selection bottleneck, and so the second response cannot be selected or initiated until the first response has been made. However, the details of the effect, and how it depends on other factors such as the stimulus and response modalities of the two tasks, form a complex pattern that has never been satisfactorily explained in any detail.

Meyer and Kieras (1997a, 1997b) provided an exhaustive treatment of the PRP effect using EPIC simulations, and mathematical analyses based on them, to account quantitatively for the results in many published and new experiments. This account interprets the PRP effect as a product of task strategy rather than as a "hard-wired" central bottleneck. In order to conform to the task instructions, subjects must adopt a strategy that postpones initiating the second response until they can ensure that it does not occur before the first response; the magnitude of the delay in the second response depends on how much of the second-task processing can be overlapped with the first task, which in turn depends on the details of the task structure (e.g., whether eye movements are required), the task difficulty, and the task modalities. The EPIC architecture captures the relevant constraints very well; Meyer and Kieras were able to construct models that accounted for the specific patterns of effects in quantitative detail and that revealed the underlying structure of the phenomena. Details and a discussion of recent experiments claiming to refute the EPIC account of PRP are available in Meyer and Kieras (1997a, 1997b).

Lessons From Multiple-Task Modeling. An immediate insight from the application of the EPIC architecture to multiple-task domains is that

there are many possibilities for performing task activities in parallel. That is, in dual-task models using EPIC, the role of the cognitive strategies in coordinating activity between the two tasks is critical to accounting for the observed effects, and, in many situations, these strategies are surprisingly subtle and efficient. In dual-task experiments, the subject is supposed to complete each of two tasks as rapidly as possible, but the higher priority task must be completed before the lower priority task, regardless of the relative speed of the perceptual or motor processing involved in the two tasks. If two tasks require the same motor processor, both perceptual and cognitive processing on the lower priority task can go on while the higher priority task is allowed to control the motor processor. After the motor processor has commenced execution of a higher priority response, the lower priority task can be given control of the motor processor, thereby honoring the task coordination requirements while maximizing speed. If the two tasks involve different motor modalities, portions of the lower priority response can be prepared in advance, so that this response can be made more quickly when its turn comes. If the two tasks compete for the use of both the eyes and the hands, the executive rules can dynamically switch control of the two processors between the two tasks so that their processing is interleaved, minimizing idle time. Thus, in a dual-task situation, the cognitive-processor strategies are responsible for allocating the eyes and the motor processors to the two tasks as needed to maximize overall performance. Similarly, in a single multimodal task with a requirement for speed, the task strategy is responsible for ensuring that the individual processors do their work as soon as possible so as to minimize the total time required for the task.

The Need for Modeling Policies

Model Fitting Versus Performance Prediction. There are multiple possibilities for how activities in a multimodal task can be overlapped under the EPIC architecture. One way to identify the specific strategy that governs overlapping in a task is to propose a strategy, generate predicted performance under that strategy, compare the predicted performance to empirical data, and repeat until the predicted data match the empirical results. In typical scientific cognitive modeling work devoted to verifying a cognitive architecture and understanding how a task might be done, it is acceptable to arrive at task strategies in this post hoc model-fitting mode. However, after scientific work on cognitive architectures has progressed beyond simple demonstrations of feasibility, success at a priori prediction is required to fully establish the architecture on scientific grounds and to use the architecture in practical settings to analyze the merits of alternative designs. Predicting performance on an a priori basis requires not only a usefully accurate cognitive architecture but also a set of modeling policies

for how to choose and represent task strategies on an a priori basis. Developing such policies can only be done by systematically characterizing the space of possible models and testing their accuracy; an example was reported by Kieras et al. (1997), who modeled a task previously studied by Gray, John, and Atwood (1993).

An Example of Performance Prediction via Modeling Policies. In Kieras et al. (1997), EPIC was used to predict performance in a well-practiced multimodal task: Telephone operators collect billing numbers spoken by customers, enter the numbers into a computer workstation, and verify the numbers before allowing the call to proceed. The volume of this work is such that saving a few seconds of work time per call is worth millions of dollars annually in labor costs. Human operators normally overlap speaking and listening to the customer with striking keys and watching for information to appear on the screen. The time taken to handle the call is not simply the sum of the individual activity times but is a complex function of which activities can be overlapped and to what extent. Our EPIC models were constructed on an a priori basis following several modeling policies that start with a simple procedurai task analysis which is then translated in a standardized format into a set of EPIC production rules. The predicted task execution times were accurate within 10% to 14%, which is useful in an engineering context. The EPIC architecture accurately represents the perceptual and motor constraints in the task, making it possible to easily construct a model on an a priori basis that predicts the task time accurately enough to aid in choosing between alternative designs. The effort required to construct the EPIC models is fairly modest. In return, the resulting EPIC models can generate predicted execution times for all possible task instances within the scope of the model. Thus, EPIC models appear to be efficient engineering models for multimodal high-performance tasks.

3. EXAMPLES OF APPLYING EPIC TO HUMAN–COMPUTER INTERACTION

Our goal in this article is to present the EPIC architecture and show how it can be applied to a variety of problems in HCI. Thus, in this section, we emphasize the variety by presenting a series of vignettes illustrating current applications of EPIC to various HCI situations, ranging from low-level interaction phenomena to complex interactions with visual displays in dual-task settings. Two recurring themes in this work are the critical role of visual layout and eye movements and the importance of parallel processing or multitask execution, even in simple situations.

3.1. Selecting Items From Menus

Choosing items from a pull-down menu with a mouse is a standard feature of many current interfaces. However, the research and practical design literature does not contain a comprehensive or even very explicit model of how such menu access is done. At first glance, it would seem that the user must first visually scan the list of menu items, looking at each one in turn, and when the desired item is found, make a movement with the mouse to position the cursor there. Other authors (e.g., Sears & Shneiderman, 1994) have made other proposals for menu search that are rather more elaborate but without detailed empirical support, and there is even some support for the notion that menu search can be completely random (Card, 1984), although other results appear to refute it (Lee & MacGregor, 1985). EPIC can be used to represent different hypotheses about how users select menu items, and the predicted results can then be compared to data with high precision. Working with us, Anthony Hornof has begun to model menu access as part of a larger program of research on visual search and visual layout. Hornof's first results suggest that EPIC can be used to address a variety of visual issues in interface design (see also Hornof & Kieras, 1997).

Hornof has modeled performance in a task that was one condition of an experiment by Nilsen (1991) in which subjects selected items from a pull-down menu. In Nilsen's experiment, the subject was shown a digit, and then he or she clicked on a target. This would cause a vertical menu of the digits 1 to 9 to appear in a random order below the cursor position. The subject then pointed to and clicked on the previously designated digit in the menu. The time to select a digit as a function of its location in the randomly ordered menu was fairly linear, with a slope of about 100 msec per item; similar results have been obtained elsewhere.

Serial-Search Model

Hornof constructed a serial-search model for menu-item selection that corresponds to the one-at-a-time hypothesis of visual search. The eye is moved to the next object down the menu, and if that object matches the sought-for item, a pointing movement is initiated to the item; otherwise, the eye is moved to the next object. Figure 2 shows the key production rules in this model. First, the rule IF-NOT-TARGET-THEN-SACCADE-ONE-ITEM waits for the current visual object (bound to the variable ?OBJECT) to have a text LABEL property in visual working memory and fires if this label does not match the sought-for label bound to the variable ?NT. If this rule fires, the DELDB action (delete from the production-system database) and the ADDDB action (add to the database) update the production-system database to make the object below the current item be the new current item, and the SEND-TO-MOTOR action instructs the ocular motor processor to move the eye to this object.

Figure 2. **Production rules from the serial search model of menu selection. The first rule repeatedly moves the eye down the menu as long as the text label for the menu item fails to match the sought-for item in working memory. If a matching item is found, the second rule points the mouse cursor to it.**

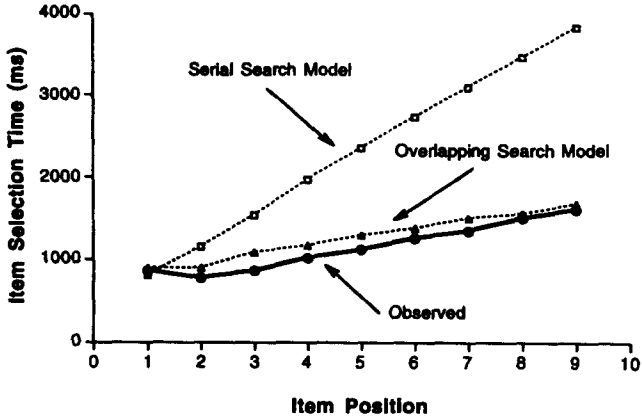
```
(IF-NOT-TARGET-THEN-SACCADE-ONE-ITEM
IF
((GOAL DO MENU TASK)
  (STEP VISUAL-SEARCH)
  (WM CURRENT-ITEM IS ?OBJECT)
  (VISUAL ?OBJECT IS-ABOVE ?NEXT-OBJECT)
  (NOT (VISUAL ?OBJECT IS-ABOVE NOTHING))
  (MOTOR OCULAR PROCESSOR FREE)
  (VISUAL ?OBJECT LABEL ?NT)
  (NOT (WM TARGET-TEXT IS ?NT)))
THEN
((DELDB (WM CURRENT-ITEM IS ?OBJECT))
  (ADDDB (WM CURRENT-ITEM IS ?NEXT-OBJECT))
  (SEND-TO-MOTOR OCULAR MOVE ?NEXT-OBJECT)))

(TARGET-IS-LOCATED-BEGIN-MOVING-MOUSE
IF
((GOAL DO MENU TASK)
  (STEP VISUAL-SEARCH)
  (WM TARGET-TEXT IS ?T)
  (VISUAL ?TARGET-OBJECT LABEL ?T)
  (WM CURSOR IS ?CURSOR-OBJECT)
  (MOTOR MANUAL PROCESSOR FREE))
THEN
((DELDB (STEP VISUAL-SEARCH))
  (ADDDB (STEP MAKE RESPONSE))
  (SEND-TO-MOTOR MANUAL PERFORM
    POINT RIGHT ?CURSOR-OBJECT ?TARGET-OBJECT)))
```

When the text label for this object becomes available, the rule might fire again. However, if the text label for the object matches the sought-for label, the rule **TARGET-IS-LOCATED-BEGIN-MOVING-MOUSE** will fire next instead. This rule disables both itself and the first rule from firing again by removing the **(STEP VISUAL-SEARCH)** item from the database, enables the next step in the procedure by adding a **STEP** item, and sends the manual motor processor an instruction to perform a right-hand movement using the **POINT** style that positions the mouse cursor onto the target; the execution time follows Fitts' law (see earlier; Card et al., 1983).

Applying this model to Nilsen's (1991) task requires estimating a parameter for how long it takes to recognize the text label for the digits and where on the retina this recognition could be done. Hornof assumed that the digit recognition could be done only in the fovea and that it required 200 msec per digit—a value that had been used for similar stimuli in models for other

Figure 3. **Observed and predicted menu selection times. Observed times (solid points and lines) are from Nilsen (1991); predicted times (open points and dotted lines) are explained in the text.**



domains. EPIC was run in a simulated version of Nilsen's experiment, and predicted menu selection times were obtained. As shown in Figure 3, the serial-search model seriously misfits the data by predicting a slope of about 380 msec per item, far larger than the observed value of about 107 msec. The reason for the discrepancy is that, according to the EPIC architecture, the serial-search model will require an estimated time of at least 200 msec for the perceptual process to identify the menu item, about 50 msec for a cognitive-processor production rule to fire to initiate the eye movement to the next item, and due to the savings from repeated similar movements, only slightly more than about 50 msec for each subsequent eye movement. Even if the perceptual processing takes much less time (e.g., only 100 msec), the oculomotor and cognitive times are still too long to produce a slope as shallow as 100 msec. In summary, data such as Nilsen's are extremely difficult to reconcile with a model that assumes a strategy of sequentially fixating and deciding about each menu item separately.

Overlapping-Search Model

Hornof next developed a more sophisticated strategy that more fully exploits the parallelism possible in EPIC. A scanning process moves the eye from one item to the next, relying on the parallel perceptual processing "pipeline" to complete the recognition processing of each item. Meanwhile, a separate matching rule waits for the sought-for item to be recognized and appear in working memory; then, it stops the scan and initiates the mouse movement. The relevant rules are shown in Figure 4. The production rule SACCADE-ONE-ITEM moves the eye from item to item as rapidly as possible. Because the movements are repeated, the oculomotor

Figure 4. **Production rules from the overlapping-search model. The first rule moves the eye rapidly down the menu, regardless of whether a matching item is present. The second rule halts the scan if a matching item appears in visual working memory and enables the third rule, which moves the eye back to the matching item and launches the mouse-pointing movement to it.**

```
(SACCADE-ONE-ITEM
IF
((GOAL DO MENU TASK)
   (STEP VISUAL-SWEEP)
   (WM CURRENT-ITEM IS ?OBJECT)
   (VISUAL ?OBJECT IS-ABOVE ?NEXT-OBJECT)
   (NOT (VISUAL ?OBJECT IS-ABOVE NOTHING))
   (MOTOR OCULAR PROCESSOR FREE)
   )
THEN
((DELDB (WM CURRENT-ITEM IS ?OBJECT))
   (ADDDB (WM CURRENT-ITEM IS ?NEXT-OBJECT))
   (SEND-TO-MOTOR OCULAR MOVE ?NEXT-OBJECT)))

(STOP-SCANNING
IF
((GOAL DO MENU TASK)
   (STEP VISUAL-SWEEP)
   (WM TARGET-TEXT IS ?T)
   (VISUAL ?TARGET-OBJECT LABEL ?T))
THEN
((DELDB (STEP VISUAL-SWEEP))
   (ADDDB (STEP MOVE-GAZE-AND-CURSOR-TO-TARGET))
   (ADDDB (WM TARGET-OBJECT IS ?TARGET-OBJECT))))

(MOVE-GAZE-AND-CURSOR-TO-TARGET
IF
((GOAL DO MENU TASK)
   (STEP MOVE-GAZE-AND-CURSOR-TO-TARGET)
   (WM TARGET-OBJECT IS ?TARGET-OBJECT)
   (WM CURSOR IS ?CURSOR-OBJECT)
   (MOTOR OCULAR PROCESSOR FREE)
   (MOTOR MANUAL MODALITY FREE))
THEN
((DELDB (STEP MOVE-GAZE-AND-CURSOR-TO-TARGET))
   (ADDDB (STEP MAKE RESPONSE))
   (SEND-TO-MOTOR OCULAR MOVE ?TARGET-OBJECT)
   (SEND-TO-MOTOR MANUAL PERFORM
      POINT RIGHT ?CURSOR-OBJECT ?TARGET-OBJECT)))
```

processor produces them very rapidly, as already described. Before the eye moves on, the digit in the fovea gets started in the perceptual recognition "pipeline" already mentioned, and eventually the recognized LABEL property of the object gets deposited in visual working memory. Meanwhile, the rule STOP-SCANNING functions as an independent "demon" waiting for an item to appear in visual working memory that matches the target. As soon as it does, this rule shuts down the scanning process by removing the item (STEP VISUAL-SWEEP) and then enables the rule MOVE-GAZE-AND-CURSOR-TO-TARGET. This rule then launches an eye movement and mouse cursor movement to the matching object. Thus, the search for the matching item is conducted partially in parallel, because the process of recognizing the labels for the desired object is overlapped in time with the eye movements required to move the fovea between the objects. Using the same parameters for the digit-recognition availability and time, this overlapping-search model predicts the values shown in Figure 3; the model predicts a slope of about 103 msec, which is an excellent match for the empirical value.

Implications for Eye and Hand Movements

This model makes claims about eye movements that might seem extreme and that certainly are subject to empirical test. Also, this is not the only possible EPIC model that could fit the data well, and there are additional effects in Nilsen's (1991) data to account for (see Hornof & Kieras, 1997). However, it is important to see how the claims of this model follow from the EPIC architecture. Because the menu items are uniformly spaced along a single line, the eye movements between the items are repeated movements, meaning that the feature structure of each movement is identical to that of the previous movement. Once started, the eye movements are very fast because the oculomotor-processor movement-preparation time is zero, leaving only the execution time. The cognitive processor simply commands each movement as soon as the oculomotor processor is ready. After a visual stimulus has been foveated, it can continue to be processed in the perceptual recognition pipeline while the eye moves on to the next item. The fully parallel capability of EPIC's cognitive processor makes it a simple matter to completely overlap the execution of the scanning process with the detection of the sought-for item. Thus, one possibility that EPIC presents is that the relatively fast menu selection time can be a simple result of scanning being done in parallel with matching.

Another possible model would be based on the idea that more of the menu items can be recognized outside the fovea (or the "effective" fovea is larger). For example, if about three digits can be discriminated simultaneously, then the scanning strategy could simply move the eye in jumps large enough to include each item in the "effective" fovea only once. This

alternative model requires fewer, and less frequently executed, eye movements but still produces the overall fast menu selection times. In addition, there is another important effect in Nilsen's (1991) unordered menu data: Shorter menus are processed more rapidly across the board, which suggests a random search strategy like that argued for by Card (1984). By assuming that subjects perform a mixture of these different strategies, Hornof and Kieras (1997) were able to account for these effects in Nilsen's data. The goal of this line of work is a unified account of menu selection in terms of visual search and mouse pointing mechanisms, with the ultimate goal being an evaluation tool for visual layout: Two designs for the visual layout of the interface can be specified, and the corresponding EPIC models for the task could be run to determine which interface demands more visual work in the form of eye movements or perceptual delays.

An important detail in these results concerns the contribution of the mouse-movement time, which is well known to follow Fitts' law, a nonlinear function. However, both the predicted and observed times in these results are rather linear—what happened to the mouse-movement time? One possibility is that Nilsen's (1991) subjects did not make single mouse movements to the item but instead "scanned" the mouse cursor down the menu, halting at the desired item, as suggested by Sears and Shneiderman (1994). However, proposing that the eye could scan this rapidly is difficult enough to accept; that the hand could be moved this quickly seems quite implausible. A better and simpler explanation is the quantitative explanation provided by EPIC. It happens that over the range of target sizes and distances involved in Nilsen's menus, the Fitts'-law times for mouse movement are both relatively short compared to the total item-selection times and are also very close to being linear. Consequently, the mouse-movement time component of the total time does not result in noticeable nonlinearity. Thus, in contrast to verbal reasoning about qualitative properties of effects, detailed quantitative models such as EPIC can greatly clarify how different component mechanisms actually contribute to task-execution time.

Conclusions

This work illustrates how even an elementary aspect of using an interface can involve important and subtle aspects of the human performance architecture. Also, it shows how the quantitative parameter values built into EPIC impose powerful constraints on what strategies can serve as accurate models of cognitive processing. For example, it is possible to rule out the serial-search strategy because EPIC determines that the absolute minimum times for moving the eye and making the decision for each item leave no time for perceptual processing of the items. This minimum time, in turn, is determined by EPIC's feature representation of movements, in which the similarity of the repeated eye movements resulting from the

Figure 5. **Production rule that enters each digit in the model for the telephone operator task. The output from the auditory recognition identifies the key to be struck by the manual motor processor.**

```
(*Enter-number*Get-next-digit
IF
((GOAL Enter number)
  (STEP Get next digit)
  (WM Next speech is ?prev)
  (AUDITORY SPEECH PREVIOUS ?prev NEXT ?next TYPE DIGIT CONTENT ?digit)
  (VISUAL ??? SHAPE FIVE-KEY)
  (MOTOR MANUAL PROCESSOR FREE))
THEN
((SEND-TO-MOTOR MANUAL PERFORM Peck ?digit)
  (DELDB (WM Next speech is ?prev))
  (ADDDB (WM Next speech is ?next))))
```

specific visual layout of the task means that they will take little time to prepare. A final quantitative result provided by EPIC's mechanisms concerns the linearity of the selection-time function: The nonlinear component contributed by the mouse-movement time is not sufficiently large, given the specific layout of the menu, to produce a detectably nonlinear selection-time function.

3.2. Auditorily Driven Keyboard Data Entry

Another illustration that even elementary HCI tasks can involve considerable parallelism appears in modeling telephone operator tasks. A skilled telephone operator can listen to a string of digits spoken by a customer and enter them on a keypad while they are still being spoken. During such processing, the auditory perceptual processor, the cognitive processor, and the manual motor processor are operating simultaneously; the cognitive processor plays the role of mediator between the auditory and motor processors, feeding instructions to the motor processor as rapidly as the recognized digits become available from the perceptual processor and as soon as the motor processor is ready to accept instructions for another keystroke movement preparation. As part of Kieras et al.'s (1997) work, this performance was examined and modeled in detail.

Overlapping Auditory and Manual Processing

The auditory processor produces a series of auditory working memory items that contain the recognized digits recoded as identities of the keypad keys, chained together to preserve the order in which they were heard. Figure 5 shows a production rule, *Enter-number*Get-next-digit, from a model

used in Kieras et al. (1997). The rule uses these recoded digits to make the corresponding keystrokes in a "pipeline" fashion similar in spirit to John's (1996) model of transcription typing. Each recognized spoken digit is represented in auditory working memory as a sequentially tagged item of the form (AUDITORY SPEECH PREVIOUS ?prev NEXT ?next TYPE DIGIT CONTENT ?digit), where the variable ?digit represents the recoding supplied by the auditory perceptual processor that designates the physical target of the corresponding key.

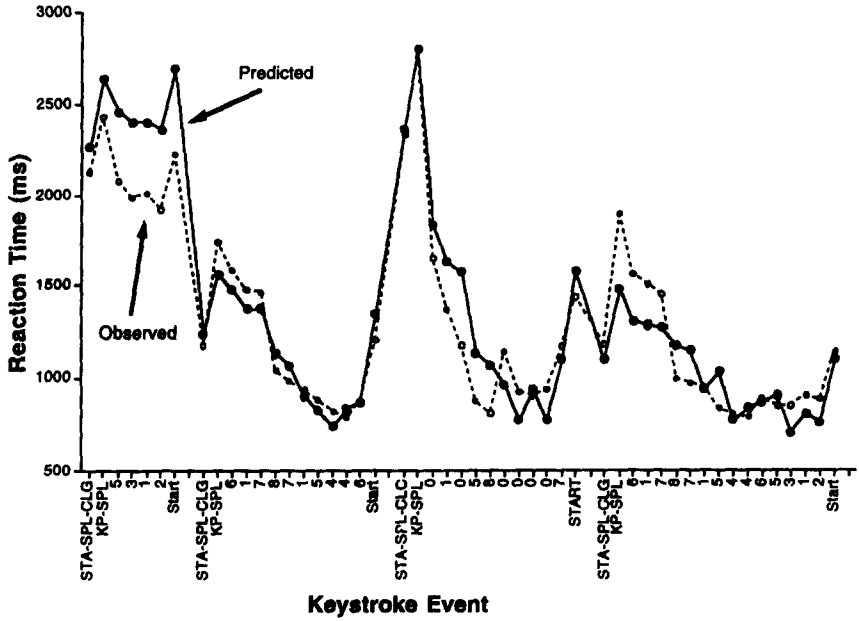
As each digit arrives in working memory, the rule fires when the manual motor processor has begun executing the previous keystroke, then sends the keystroke command corresponding to the digit to the manual motor processor, and also updates a "pointer" in WM to the next speech item in auditory working memory to be processed. The rule also requires that before the digit can be typed, the shape of the center key on the keypad, the FIVE-KEY, must be in visual working memory to ensure that the target key is in view. The manual motor processor uses the code for the digit key to prepare the features for a peck-style movement to strike the key and then initiates the prepared movement as soon as the hand is physically free to do so. The preparation process takes 0 to 100 msec, depending on the similarity of the present movement to the previous movement, while movement itself takes 50 msec to initiate, followed by a minimum of 100 msec to physically execute. Thus, there is enough time during each keystroke execution to prepare the features for the next keystroke, assuming that the cognitive processor has provided the next keystroke instruction soon enough. If so, then a sequence of keystrokes can be made quite rapidly, on the order of 150 to 200 msec apart. However, if the auditory input is not supplied rapidly enough, the keystrokes will be slower, but exactly how much slower depends on the subtle details of the event timing.

Buffering Effects

In the telephone operator task, the subtask of entering spoken digits is part of a larger task in which the operator must determine from the customer's speech what keys to strike to indicate the billing category of the call and then enter the billing numbers. The operator first indicates the billing category by striking the STA-SPL-CLG key, then strikes the KP-SPL key to signal that the billing number is about to be entered on the numeric keypad, and, then, enters the billing number digits. Performance in the whole telephone operator task was modeled by Kieras et al. (1997), but a specialized EPIC model was used to explore some of the details of the auditory digit-entry subtask. In this model, the operator waits for the customer to speak the first digit before striking the STA-SPL-CLG key, followed by the KP-SPL key, and then strikes the digit keys according to the rule shown in Figure 5.

Individual stimulus and response timings were obtained from some of the videotaped performances used in Kieras et al. (1997). Figure 6 shows

Figure 6. **Observed and predicted RTs for keystrokes made in response to speech input. Observed RTs are solid points and lines; predicted RTs are open points and dotted lines. The keystroke events are for four different task instances performed by a single subject. Each task instance involves a different sequence of keystrokes made in response to a billing request followed by digits spoken in a unique timing pattern. The keystrokes are shown in temporal order on the horizontal axis.**



the observed and predicted RTs for each keystroke, measured from the auditory stimulus event defined as the stimulus for that keystroke. The observed RTs are from a set of four task instances performed by a single operator, where each task instance was a unique interaction between a customer and an operator, with a different sequence of digits, spoken in a different timing pattern, from the other instances. Thus, the RTs shown in Figure 6 are unaggregated, individual subject observations. Although there is a tendency for the RT to decrease during the sequence of keystrokes within a task instance, the RTs during this seemingly trivial task vary tremendously, both within and between task instances.

The values predicted from the EPIC model capture the general trend that the first several keystrokes are substantially delayed by the need for the previous keystrokes to be completed before the first digit keystroke. Gradually, the digit keystrokes catch up because the customer is speaking the digits at a lower average rate than they can be typed. The shortest RTs occur when the processing has caught up to the point that there is no idle waiting time. The auditory recognition parameter can be estimated as the

difference between these observed shortest RTs and the predicted time for the cognitive and motor processing to produce these keystrokes. This parameter estimate, 400 msec, together with the earlier described strategy, suffices to produce the observed complex profile of RTs in response to the speech input. Given the single-observation quality of the data, the fit between predicted and observed RTs is quite good.

Conclusions

The model shows that many of the keystrokes in the task are delayed more than the architecture requires, suggesting that performance could be speeded up by changing the workstation design in two ways. First, if the first two keystrokes could be eliminated or placed elsewhere in the task, performance would be speeded up, and fewer digits would have to be buffered in working memory. Second, if the customer spoke the digits at a higher rate, or speech compression was used to produce the same effect, the task could in fact be done faster on the average. Given that, in this task domain, saving a second of average task time is credited with a considerable financial saving (Gray et al., 1993), the ability of EPIC to reveal these detailed aspects of performance is an important result.

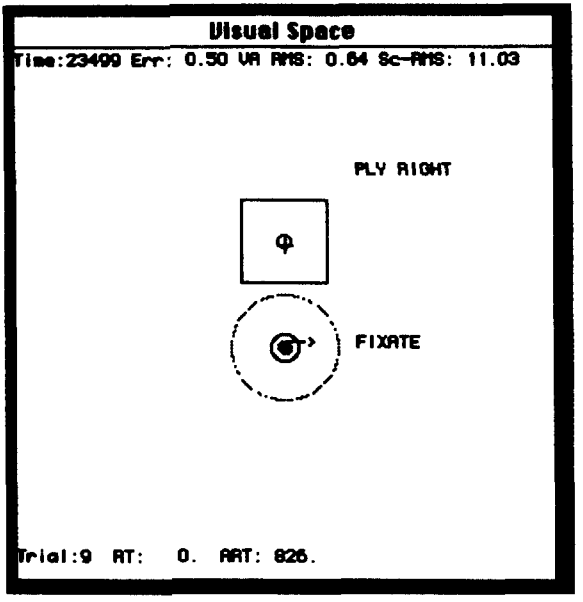
3.3. Eye Movements and Executive Control in a Simple Dual Task

Computers are used not just in desktop or office settings but also in real-time contexts such as airplane cockpits, where many of the displays and controls are actually computer interfaces. As already mentioned, EPIC has been developed to deal with multiple-task situations, such as those in which a pilot has to track a target on one display and make decisions using the information in another display. The spatial distribution of visual information and the timing of eye movements play critical roles in determining performance in such situations. In the remainder of this article, we describe two studies, one involving a simple form of this paradigm and the other involving a complex form.

We have modeled some results obtained by Martin-Emerson and Wickens (1992) that are especially instructive about the role of eye movements and the use of visual information during dual tasks. Figure 7 shows the EPIC model display of the experimental task. The display represents the visual environment of EPIC with the objects in their correct sizes and positions; the small gray circle marks the location and size of EPIC's fovea, currently on the choice stimulus, and the larger gray circle marks the boundary of the parafovea, a region of intermediate discriminative ability.

The lower priority of the two tasks is a compensatory tracking task carried out in the upper box of the display. A quasirandom perturbing

Figure 7. **EPIC model display for the Martin-Emerson and Wickens (1992) task. The tracking task is in the upper square; the eye is fixated on the choice-stimulus arrow appearing (not to scale) in the lower circle.**



force drives the cursor (the cross) away from the target (the small circle), and the subject must manipulate a joystick with the right hand to keep the cursor centered on the target. The higher priority of the two tasks is a choice-reaction task: Occasionally a stimulus appears in the choice-stimulus area (the solid circle below the tracking box), which is either a left- or right-pointing arrow (not shown to scale in the display). The subject must respond by pressing one of two buttons with the left hand as soon as possible, all the while attempting to maintain the cursor on the target.

The major independent variable is the distance (in visual angle) between the tracking target and the choice stimulus, and a second independent variable is the difficulty of the tracking task. The two dependent variables are the RT for the choice task and a measure of tracking performance (viz., the average root mean square error [*RMSE*] in the tracking task), collected for a 2-sec period following the onset of the choice stimulus. The observed effects are that the choice RT increases with the angular distance between the target and the choice stimulus but is unaffected by tracking difficulty. The *RMSE* increases somewhat with the angular distance for both levels of tracking difficulty.

Our models for this dual-task situation assume that successful tracking requires the eye to be kept on the tracking cursor, and likewise, the eye must be moved to the choice stimulus in order to discriminate it. However,

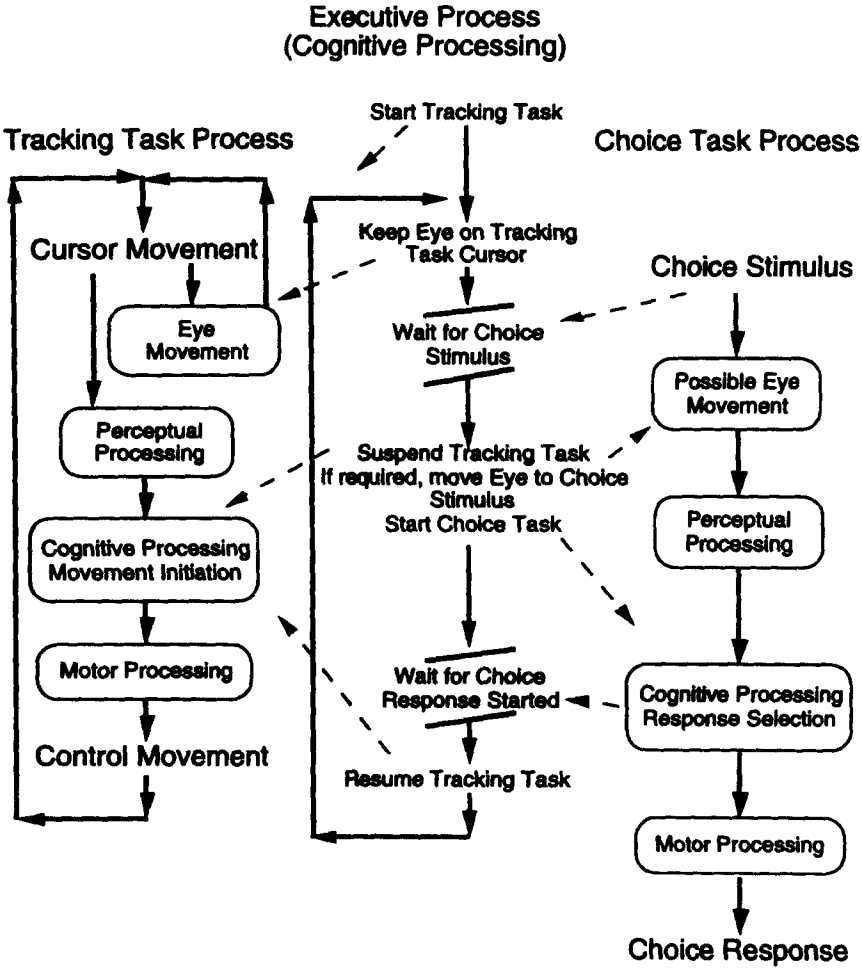
if the choice stimulus is close enough to the tracking cursor, parafoveal vision will be adequate to discriminate the stimulus without moving the eye. Hence, the two tasks often, but not always, compete for use of the eye. Finally, because both tasks involve manual responses, they compete for access to the manual motor processor. We illustrate how EPIC can be applied to this task with two models.

A Simple Lockout Model of Executive Control

The lockout model uses a simple strategy that is consistent with traditional thinking about dual-task situations; namely, the lower priority task is locked out (suspended) while the higher priority task is executed. The strategy is shown in Figure 8, where our attempt to portray parallel interactive processes in flowchart form is explained as follows. The flowchart on the left-hand side of Figure 8 diagrams the tracking task. The tracking-task process waits for a cursor movement, and then it simply makes a motor movement if the cursor is too far off the target and waits for another cursor movement. In addition, another very simple iterative subprocess ensures that the eye stays on the cursor in case the autonomous oculomotor mechanism fails to keep up. On the right-hand side of Figure 8 is the flowchart for the choice task. When the stimulus appears, there may be a delay while an eye movement is made to it, followed by recognition of the stimulus and selection and production of a response. The executive process is the flowchart in the center; it monitors and controls the other processes via items in the production-system working memory; these relations are shown by the dashed arrows. The executive first starts the tracking task, allocates control of the eye to it, and then waits for the choice stimulus to appear. When the stimulus appears, the executive process suspends the tracking task, enables the choice-task rules, and then moves the eye to the stimulus if it is too far away to be discriminated. The executive waits for the choice response to be initiated, and then resumes the tracking task, and returns control of the eye to it. In this way, using what we term *lockout scheduling*, the executive process allows only one task to be done at a time, ensuring that the choice task has priority over the tracking task and that the eye and the manual motor processor are used for only one task at a time.

Unfortunately, this simple strategy does not fit all aspects of the data. Figure 9 shows the observed values for the choice RT and the tracking error together with the values predicted by the lockout model. By estimating the perceptual recognition time parameter from the data, we can fit the choice RT function fairly well. As in the data, there is no effect of tracking task difficulty because the choice task is given priority over tracking. The first few points are fairly flat, because here the choice stimulus can be recognized in the parafovea, and the eye does not need to be moved. The upward slope of the curves at larger separations reflects the time required

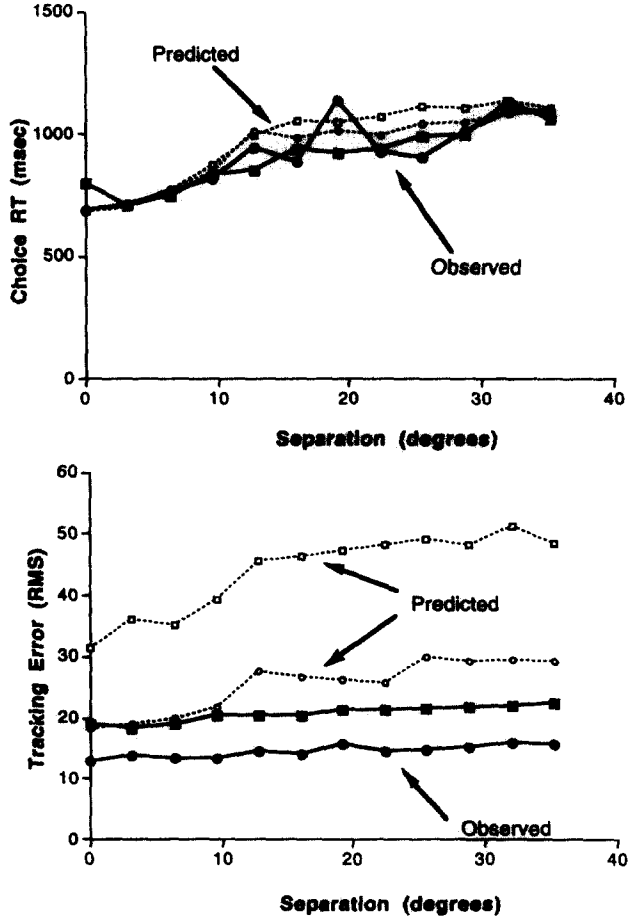
Figure 8. **Flowchart of lockout-model strategy for the Martin-Emerson and Wickens (1992) task. The tracking task is suspended whenever the choice task is executing. The solid lines represent the flow of execution in each process; the dashed lines represent signal and control relations.**



to move the eye. The fit of the simulated tracking data is extremely poor, however. The magnitude of the tracking error is seriously overpredicted, as are the effects of tracking difficulty and visual separation.

The lockout model cannot be made to fit the tracking data better by adjusting the relevant parameter values; it is already using the minimum plausible time estimates for all of the perceptual and motor parameters involved. Because the RMSE is measured for a brief (2-sec) period of time starting with the onset of the choice stimulus, if tracking is suspended for too long during this period, the effect will be substantial. The lockout

Figure 9. **Observed and predicted effects of stimulus separation and tracking difficulty for the lockout model. Observed values are solid points and lines; predicted values are open points and dotted lines. Square points are for the difficult-tracking condition; circular points are for the easy-tracking condition. The fit of the choice RT is satisfactory, but predicted tracking performance is far too poor.**

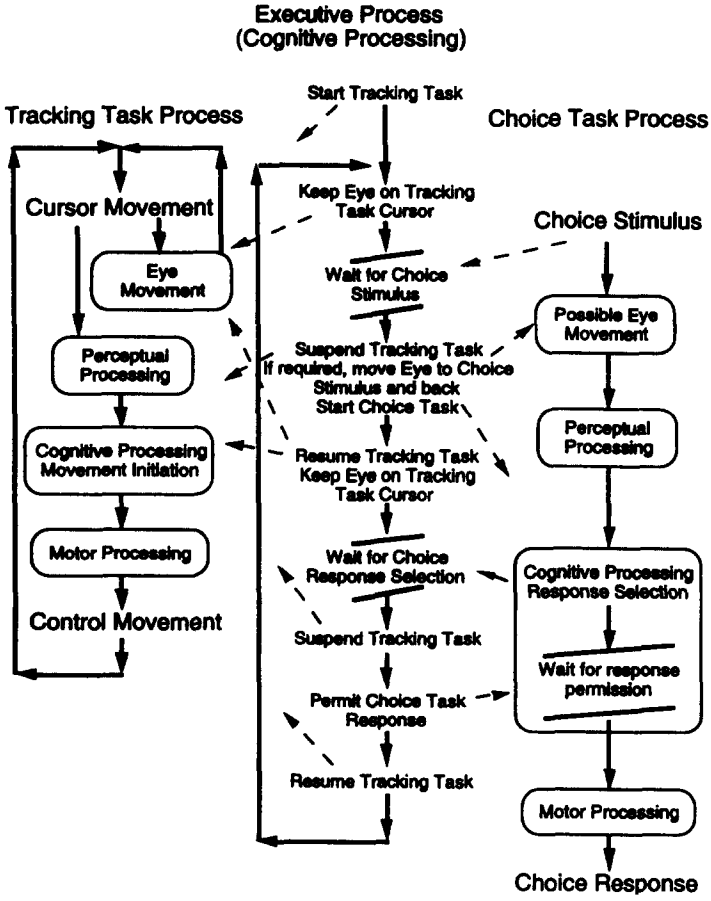


model suspends the tracking task for such a long time that considerable tracking error accumulates; it is simply too inefficient.

An Interleaved Model of Executive Control

In order to provide a more efficient strategy, we constructed a second model, the interleaved model, in which the executive process overlaps the two tasks as much as possible; this strategy is shown in Figure 10. The executive process starts out the same as in the lockout model, but when the

Figure 10. **Flowchart of the interleaved model for the Martin-Emerson and Wickens (1992) task. The tracking task is executed while choice stimulus recognition and response selection go on and is interrupted for the minimum necessary time.**



choice stimulus appears, the executive moves the eye to it and then immediately begins to move the eye back to the tracking task, relying on the "pipeline" property of the visual system to acquire the stimulus and continue to process it even after the eye has returned to the tracking cursor. The tracking task is suspended only while the eye is away looking at the stimulus for the choice task. The executive then uses the same approach as in our PRP models for allocating control of the manual motor processor. When the choice task has chosen the response, it signals the executive, which again suspends the tracking task, gives the choice task permission to command the manual motor processor, and then resumes the tracking task right away. Thus, the same task priorities are honored,

but the tracking task is interrupted as little as possible. The predictions from this model are shown in Figure 11. The choice RTs are again well fit, but now the tracking-task predictions are extremely close as well. Because the executive allocates the eye and the manual motor processor to the tracking task for the maximum amount of time, the tracking task rules can squeeze in a few movements while the choice task is underway, resulting in substantially better tracking error than in the lockout model.

Conclusions

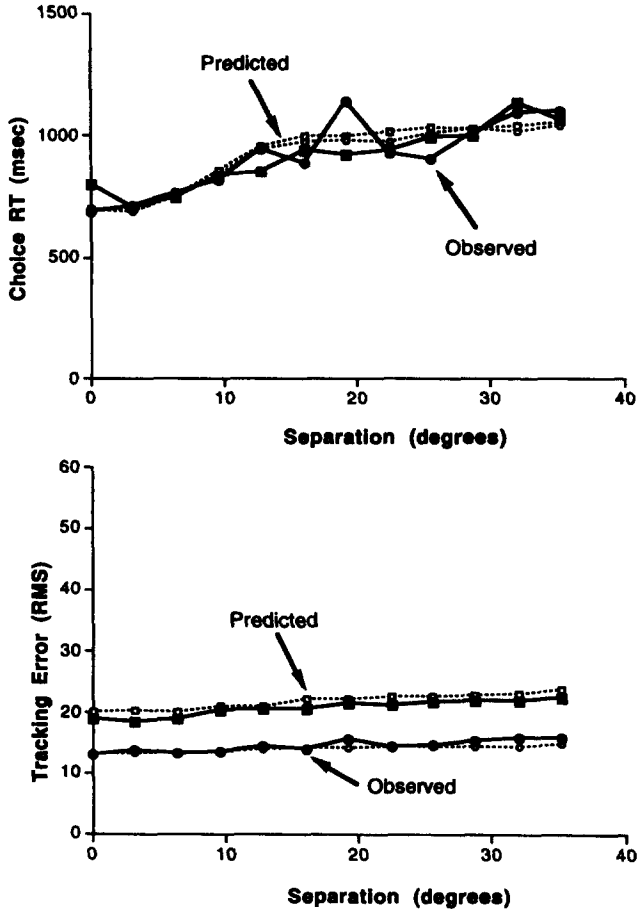
As mentioned earlier, control of the eye has often been unappreciated, but it can clearly be critical in dual-task paradigms. A more subtle result is that subjects can and apparently do use highly refined strategies that can be surprisingly efficient for coordinating dual tasks. As an aside, in this model the executive handles the use of the eye directly, by moving it to the stimulus for the appropriate task. An alternative is to let each task move the eye itself, with the executive granting permission to move the eye to the appropriate task. This latter approach is the one followed in the complex dual-task model to be discussed.

An important general conclusion well illustrated by this particular modeling work concerns a common misunderstanding about computational models. They do not in fact have so many "degrees of freedom" that they can be made to fit any data at any time. Working within the fixed EPIC architecture sets powerful constraints. Given the basic lockout strategy, there are no parameter values or specific strategy details that would allow us to fit the data as a whole. The only way an EPIC model could fit the data is by assuming a fundamentally different strategy. Thus, a general conclusion (see also Meyer & Kieras, 1997b) is that the exercise of seeking quantitatively accurate accounts of data within a fixed architecture is extremely informative both about the accuracy of the architecture itself and about the structure and requirements of the task.

3.4. A Complex Dual Task With Automation

Our modeling work on the Martin-Emerson and Wickens (1992) task laid the foundations for our work on a more complex dual tracking/choice task. This task was developed by Ballas, Heitmeyer, and Perez (1992a, 1992b) to resemble a class of tasks performed in combat aircraft in which analyzing the tactical situation is partially automated by an on-board computer. To help with our explanation, we show the EPIC model display for this task in Figure 12. The right-hand box contains a pursuit-tracking task in which the cursor (cross) must be kept on the target (small box). In the experiment reported by Ballas et al., average tracking-error data were collected during various phases of the experiment. The left-hand box

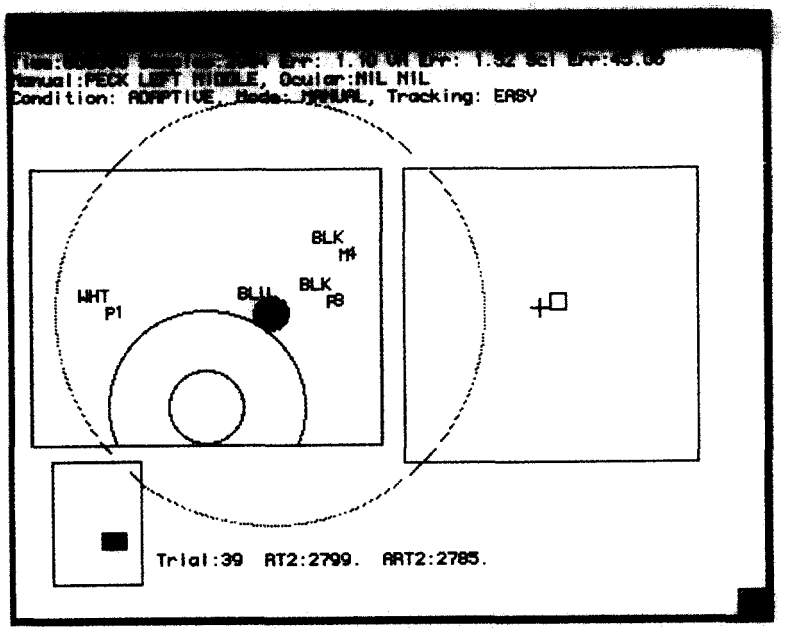
Figure 11. **Observed and predicted effects of stimulus separation and tracking difficulty for the interleaved model. Observed values are solid points and lines; predicted values are open points and dotted lines. Square points are for the difficult-tracking condition; circular points are for the easy-tracking condition. The fit is good for both choice RT and tracking error.**



contains the choice task, a tactical decision task in which targets (or "tracks") must be classified as hostile or neutral based on their behavior. EPIC's eye is shown currently on one of the targets. These targets represent fighter aircraft, cargo airplanes, and missile sites that move down the display as the subject's aircraft travels. Ballas et al. collected choice RT data during performance of the tactical-decision task.

In the actual Ballas et al. (1992a) display, each type of target was coded by an icon; for simplicity, in the EPIC display, they are represented

Figure 12. **EPIC model display for the Ballas et al. (1992a, 1992b) task. The tracking-task target and cursor are on the right; four targets are moving down the tactical-decision task display on the left. The small open rectangle on the bottom left represents the response keypad, which has been displaced from its actual position for convenience; the small solid rectangle represents the current position of the finger used to "peck" keys. The eye has been positioned on the blue target, and one of the keystrokes is in progress.**



instead by a code letter. A track number identifies each object. Targets appear near the top of the display and then move down the display. After some time, the on-board computer attempts to designate the targets, indicating the outcome by changing the target color from black to red, blue, or amber, which the EPIC display shows with a three-character abbreviation. The subject must respond to the color change in a target in one of two ways. If the target becomes red (hostile) or blue (neutral) the subject must simply confirm the computer's classification; the response is striking a key for the hostile/neutral designation followed by a key for the track number. If the target becomes amber, the subject must classify the target based on a set of rules concerning the target's behavior and then type the hostility designation and track number. After the response, the target changes color to white and then disappears from the display some time later. The basic dependent variable is the two RTs for the targets, measured from when the target changes color to when the first and second of the two response keystrokes are made.

Ballas et al. (1992a, 1992b) investigated different interfaces for the tactical task. Our earlier description pertains to one of the four interfaces; the other three interfaces involved using a tabular display instead of the graphical radar-like display and a touchscreen instead of a keypad. The model presented here accounts for performance for the graphical-keypad interface already described. Additional work is underway on the other display and response formats.

A Performance Deficit Produced by Automation

Ballas et al. (1992a, 1992b) examined the effects of *adaptive automation.* These effects arise when, from time to time, the tracking task becomes more difficult, and the on-board computer takes over the tactical task, signaling as it does so. The computer then generates the correct responses to each target at the appropriate time, with the color changes showing on the display as in the manual version of the task. Later, the tracking becomes easy again, so the computer signals and then returns the tactical task to the subject; the experiment is arranged so that the subject must resume the tactical task when several black targets are on the display and one target has simultaneously changed color. Under such conditions, Ballas et al. observed an *automation deficit effect* in which, for a time after resuming the tactical task, subjects produced longer response times in the tactical task compared to their usual steady-state manual performance. This effect raises serious concerns about possible negative consequences of automation in combat situations; if the automation fails, the operator can lack *situation awareness,* and it might take a dangerously long time to catch up.

A Model for the Ballas Task

From single-task performance, we estimated the parameters for the basic perceptual encoding operations required in the tactical task, namely, recoding the blue and red colors to the appropriate key and recognizing the hostility of different kinds of targets, which takes considerably longer (more than a second). We have assumed that assessing the hostility of a target requires the target to be visually fixated, but that a target's color is available parafoveally, and that color-change events (which result in luminance changes) are visible in peripheral vision along with object onsets and offsets. Thus, when doing the tracking task, a color-change event in the tactical task can be detected, and this event can be used to tell that the tactical task requires action. However, like other transient events, this information will disappear from visual working memory quickly unless the task strategy involves recoding it into a more durable working memory form. After the eye has been moved to the tactical-task display, the colors

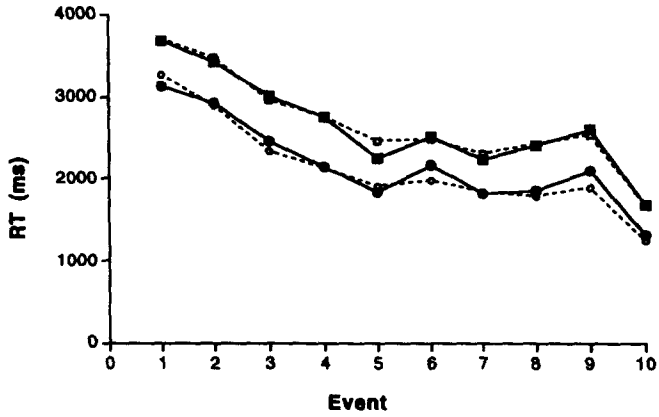
of all of the individual targets will usually become available, because most often they will fall in the parafovea.

Our model for this task and interface uses a lockout strategy at the top level of coordinating the tracking task and the tactical task but involves considerable parallelism within the tactical task. When the tactical task is the responsibility of the human (as opposed to that of the on-board computer), the executive process allows the tracking task to run until it is time to work on the tactical-decision task. In the meantime, the executive process simply notes that a target has appeared and continues tracking. The executive waits for a target to change color or to get too close to the centermost *ownship* circle; these events can be detected in peripheral vision, but the responsible target or color cannot be. The executive then suspends the tracking task and allocates the eye to the tactical task. The tactical task follows a priority scheme in choosing which target to view and process: Targets with a designation color (amber, red, or blue) are first priority, followed by targets whose color has changed, followed by targets whose color is unknown, and finally followed by an undesignated (black) target that is close to the ownship circle. If no targets qualify, the tactical task terminates, and the executive resumes the tracking task. If there is a qualifying target, the eye is moved to the chosen target; if there is more than one qualifying target at the same level of priority, one is chosen at random. The appropriate response is then made about the hostility of the target when the perceptual information (color coding or hostility behavior) becomes available; then the tactical-task process moves the eye to the target track number, and when the label is available, the corresponding response is chosen and made. After the second response is on its way to the manual motor processor, the process of choosing a new target begins in parallel with completion of the response. Thus, the tactical task has three major phases: choosing the stimulus to be processed and selecting and producing each of the responses for the chosen stimulus. Much of these three processes can be overlapped.

An Explanation for Automation Deficit

Our hypothesis about the source of the automation-deficit effect is that when resuming the tactical task, the tactical-task strategy must sort out a large number of targets, whereas during steady-state tactical-task operation, the targets are handled as they appear. That is, we assume that when the tactical task is automated, the subject does not bother to store any information about the state of the tactical display in working memory. Thus, when it is time to resume the tactical task, multiple targets are present on the tactical display, and there is no record in working memory of which have changed colors or when the color changes occurred. Thus, the information required to choose a target according to the priority

Figure 13. **Automation deficit effect shown in the observed and predicted RTs for events following resumption of the tactical task. Observed RTs are solid points and lines; predicted RTs are open points and dotted lines. The lower curve is for the hostility-designation RT; the upper curve is for the track-number RT. Starting with Event 1, the events are closely spaced, and responses are delayed and then speed up as the tactical task catches up with the situation. Event 7 begins a similar matched sequence of closely spaced events, but the tactical task is performed as needed and so is able to keep up.**



scheme is missing, so the tactical-task strategy simply picks the first target to inspect at random. After moving the eye to it and waiting for the color to become available, the strategy processes the target as usual if it is red, blue, or amber. However, if the target is white or black, it cannot be processed, and so another target is picked at random according to the priority scheme. When all candidate targets have been dealt with, tracking is resumed, and future target changes are processed as they appear.

The automation deficit results because when the tactical task is being performed normally, targets are usually processed in the order in which they change color, keeping the average RT to a minimum. In contrast, when the tactical task is resumed after automation, multiple targets must be inspected, and no information has been kept on the order in which they have appeared or changed color (otherwise, the automation is of little value!). Thus, the targets are inspected in random order, so targets that changed first will have to wait longer on average to be inspected than if they were processed as soon as they changed colors.

Figure 13 shows some automation deficit results obtained with the model in comparison to detailed data (supplied by James Ballas) that includes both RTs for correct responses only. The graph shows the predicted and observed RTs for each of the two responses for each target, measured from the time that they become designated (change color), in the order in which the targets become designated after the tactical task is

resumed. Thus, Event 1 corresponds to the first color change of a target after task resumption, Event 2 to the second, and so forth. The overall quality of the fit is very good.

An important feature of these data is that the temporal spacing of events was systematically varied. Starting with Event 1, which is simultaneous with the signal to resume the tactical task, the events happen close together in time, and then thin out until by Event 6, usually only one target is present on the display; a matched pattern of event spacing starts with Event 7. Thus, the tactical processing is not evenly paced; there are two matched periods of high workload followed by easy stretches.

The Event 1 RT is long because the tactical task is started only after the auditory resumption signal has been recognized, and then it will frequently look at some other target first, further delaying the processing. Event 2 occurs very soon after Event 1, and so responding to it is also seriously delayed because it must wait for the delayed processing of Event 1 to be completed. The increasing event spacing allows the tactical task to gradually catch up, resulting in decreasing RTs. But starting with Event 7, the events are closely spaced again, and subsequent events suffer from the processing delays that again dissipate with the increasing event spacing. However, because the state of the display is being monitored at the time of Event 7, the tactical task is able to find its target much more quickly than is the case at Event 1. Thus, the effects of event spacing are more serious if the tactical-display monitoring is just being resumed than if it was ongoing. Ballas et al. (1992a, 1992b) defined the automation deficit effect as the difference between the Event 1 RT and the RT for a matched subsequent event, Event 7. The model accounts for this measure quite accurately.

Relation to Elementary Dual-Task Phenomena

Modeling this task has revealed a remarkable continuity with our earlier modeling work with EPIC on the PRP task mentioned earlier (Meyer & Kieras, 1997a, 1997b). The PRP task consists of two overlapping choice RT tasks; the major effect is that responding to the second task is delayed while the first response is being made. As the time between the two stimuli is increased, the RT to the second task declines toward its single-task value. We chose the PRP effect as the first phenomenon to address with EPIC because it was the simplest laboratory version of a dual-task paradigm and thus a good starting point. However, even Ballas et al.'s (1992a, 1992b) complex simulated-cockpit task produces PRP effects; that is, the declining RT pattern for the first six events in Figure 13 is a kind of PRP effect; the initial slow responses that gradually speed up as the stimulus spacing increases are due to exactly the same factors that govern the PRP effect in simpler laboratory paradigms. In other words, the automation deficit effect is a form of PRP effect. Our thorough understanding of PRP

effects from the earlier modeling work has allowed us to account for performance of this realistically complex task in quantitative detail.

Conclusions

Our explanation for the automation deficit may have important implications for display and task design. For example, according to this hypothesis, resuming the tactical task could be done more efficiently if it is possible to easily select the highest priority object on the display at that time. That is, suppose the first-changed object currently on the display was coded by making it blink, which would be salient in peripheral vision. Then, the subject could simply look directly at the blinking object in order to ensure that the objects were processed in priority order. Alternatively, the automated version of the task could use a different, less salient way of representing its activity, so that the subject could still profitably monitor for the same perceptual events that are important in the manual version. Not only does the EPIC architecture supply a theoretical framework in which such issues can be explored and resolved in rigorous detail, but EPIC models can also be used to evaluate and predict the effects of the design changes implied by the explanations.

4. GENERAL CONCLUSIONS

EPIC is a computational architecture for constructing models of human cognition and performance that represent the contributions and interactions of perceptual and motor mechanisms as well as cognition in determining the time course of task execution. The examples presented in this article illustrate how EPIC can be applied to a variety of situations in which humans interact with computers both at the level of elementary interactions (e.g., menu operation and data entry) and at the level of high-speed concurrent execution of multiple display-intensive tasks. By accounting for empirical data with high accuracy in an architectural framework, models constructed with EPIC provide explanations for task phenomena with a clarity and precision far beyond the informal theorizing usually deployed in the HCI field. Further work with EPIC should lead to a comprehensive and predictive theoretical account of human performance in complex high-performance multimodal tasks.

At this point, EPIC is a research system that is not in a form suitable for routine use by system or interface designers. However, there is a technology transfer precedent: Earlier work with the CCT production rule models for HCI (Bovair et al., 1990) led to a practical interface design technique (John & Kieras, 1996; Kieras, 1988). Likewise, as the EPIC architecture stabilizes, and experience is gained in applying it to human–system analysis problems, we should be able to devise a simplified approach that will

enable designers to apply EPIC to develop improved human–system interfaces.

Our experience with the EPIC architecture also suggests some meta-level conclusions about the role of cognitive modeling in the science and engineering fields of human performance and human–system interaction:

- Computational models based on human information-processing theory can usefully predict details of human performance in system design and evaluation situations.
- Developing and applying a cognitive model to task situations relevant to real design problems is a demand test of cognitive theory; if the theory successfully represents important properties of human abilities, it should in fact be useful in practical settings.
- Powerful constraints are imposed by quantitatively fitting fixed-architecture models to detailed performance data, which lead to the discovery of plausible task strategies.

In short, a comprehensive, detailed, and quantitative theory of human cognition and performance is the best basis for applied cognitive psychology. Rather than relying only on general psychological principles or on brute-force application of experimental methodology, system design can be best informed by using a theory that addresses phenomena at the same level of detail as design decisions require.

NOTES

Acknowledgments. Thanks are due to James Ballas of the Naval Research Laboratory for his generous assistance in making his task software available and in supplying new analyses of his data.

Support. This work was supported by Office of Naval Research Cognitive Sciences Program Grant N00014–92–J–1173 to David E. Kieras and David E. Meyer and by the Advanced Research Projects Agency (under Order B328 to David E. Kieras).

Authors' Present Addresses. David E. Kieras, Artificial Intelligence Laboratory, Electrical Engineering and Computer Science Department, University of Michigan, Advanced Technology Laboratory Building, 1101 Beal Avenue, Ann Arbor, MI 48109–2110. E-mail: kieras@eecs.umich.edu. David E. Meyer, Department of Psychology, University of Michigan, 525 East University, Ann Arbor, MI 48109–1109. E-mail: demeyer@umich.edu.

HCI Editorial Record. First manuscript received December 1995. Revision received December 1996. Accepted by Wayne D. Gray. Final manuscript received April 24, 1997. — *Editor*

REFERENCES

Anderson, J. R. (1976). *Language, memory, and thought.* Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Anderson, J. R. (1993). *Rules of the mind.* Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

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

Ballas, J. A., Heitmeyer, C. L., & Perez, M. A. (1992b). Evaluating two aspects of direct manipulation in advanced cockpits. *Proceedings of the CHI'92 Conference on Human Factors in Computing Systems,* 127–134. New York: ACM.

Bovair, S., Kieras, D. E., & Polson, P. G. (1990). The acquisition and performance of text editing skill: A cognitive complexity analysis. *Human–Computer Interaction, 5,* 1–48.

Card, S. K. (1984). Visual search of computer command menus. In H. Bouma & D. G. Bouwhuis (Eds.), *Attention and performance X: Control of language processes* (pp. 97–108). London: Lawrence Erlbaum Associates, Inc.

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

Elkind, J. I., Card, S. K., Hochberg, J., & Huey, B. M. (Eds.). (1989). *Human performance models for computer-aided engineering* (National Research Council, Committee on Human Factors). Washington, DC: National Academy Press.

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, artifical intelligence, and cognitive neuroscience* (pp. 299–322). Cambridge, MA: MIT Press.

Gopher, D., & Donchin, E. (1986). Workload: An examination of the concept. In K. R. Boff, L. Kaufman, & J. P. Thomas (Eds.), *Handbook of perception and human performance, Vol. II: Cognitive processes and performance* (pp. 41.1–41.49). New York: Wiley.

Gray, W. D., John, B. E., & Atwood, M. E. (1993). Project Ernestine: A validation of GOMS for prediction and explaining real-world task performance. *Human–Computer Interaction, 8,* 237–309.

Hallett, P. E. (1986). Eye movements. In K. R. Boff, L. Kaufman, & J. P. Thomas (Eds.), *Handbook of perception and human performance* (Vol. 1, pp. 10.1–10.112). New York: Wiley.

Hornof, A. J., & Kieras, D. E. (1997). Cognitive modeling reveals menu search is both random and systematic. *Proceedings of the CHI'97 Conference on Human Factors in Computing Systems,* 107–114. New York: ACM.

John, B. E. (1996). TYPIST: A theory of performance in skilled typing. *Human–Computer Interaction, 11,* 321–355.

John, B. E., & Kieras, D. E. (1996). The GOMS family of user interface analysis techniques: Comparison and contrast. *ACM Transactions on Computer–Human Interaction, 3,* 320–351.

Kieras, D. & Meyer, D. (1995). Predicting human performance in dual-task tracking and decision making with computational models using the EPIC architecture. *Proceedings of the First International Symposium on Command and Control Research and Technology,* 314–325. Washington, DC: National Defense University.

Kieras, D. E. (1988). Towards a practical GOMS model methodology for user interface design. In M. Helander (Ed.), *Handbook of human–computer interaction* (pp. 135–158). Amsterdam: North-Holland Elsevier.

Kieras, D. E., Wood, S. D., & Meyer, D. E. (1997). Predictive engineering models based on the EPIC architecture for a multimodal high-performance human–computer interaction task. *ACM Transactions on Computer–Human Interaction, 4,* 230–275.

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

Laird, J., Rosenbloom, P., & Newell, A. (1986). *Universal subgoaling and chunking.* Boston: Kluwer.

Lee, E., & MacGregor, J. (1985). Minimizing user search time in menu retrieval systems. *Human Factors, 27,* 157–162.

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,* 1408–1412. Santa Monica, CA: Human Factors Society.

McMillan, G. R., Beevis, D., Salas, E., Strub, M. H., Sutton, R., & Van Breda, L. (1989). *Applications of human performance models to system design.* New York: Plenum.

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

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

Nilsen, E. L. (1991). *Perceptual-motor control in human–computer interaction* (Technical Report 37). Ann Arbor: University of Michigan, Cognitive Science and Machine Intelligence Laboratory.

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.

Rosenbaum, D. A. (1980). Human movement initiation: Specification of arm, direction, and extent. *Journal of Experimental Psychology: General, 109,* 475–495.

Rosenbaum, D. A. (1991). *Human motor control.* New York: Academic.

Sears, A., & Shneiderman, B. (1994). Split menus: Effectively using selection frequency to organize menus. *ACM Transactions on Computer–Human Interaction, 1,* 27–51.