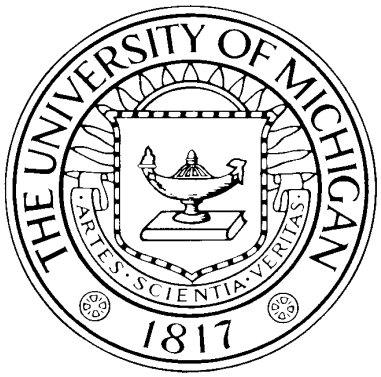
The Role of Cognitive Task Analysis in the Application of Predictive Models of Human Performance

David E. Kieras

and

David E. Meyer

University of Michigan



EPIC Report No. 11 (TR-98/ONR-EPIC-11)

March 5, 1998

This research was supported by the Office of Naval Research, Cognitive Science Program, under Grant Numbers N00014-92-J-1173 & N00014-96-1-0467. Reproduction in whole or part is permitted for any purpose of the United States Government. Requests for reprints should be sent to: David E. Kieras, Artificial Intelligence LaboratoryElectrical Engineering & Computer Science Department, University of Michigan, 1101 Beal Avenue, Ann Arbor, MI 48109-2110, kieras@eecs.umich.edu, or David E. Meyer, Department of Psychology, University of Michigan, 525 East University, Ann Arbor, MI 48109-1109, demeyer@umich.edu

The Role of Cognitive Task Analysis in the Application of Predictive Models of Human Performance

David E. Kieras and David E. Meyer

University of Michigan

Abstract

Predictive modeling of human performance as long been applied in human factors engineering. In the meantime, computational cognitive architecture models have developed a theoretically coherent and sophisticated basis for advanced predictive modeling of human performance. Key is a distinction between fixed task-independent architectural mechanisms and task-specific strategies that control the architecture. Applying the new cognitive modeling approaches to system design problems requires a method for identifying the task strategy, but new results suggest that humans choose task strategies that incorporate optional features that are not based on either architectural constraints or task demands. These optional features strongly influence performance, but cannot be identified by conventional task analysis methods. A heuristic is presented for obtained useful task performance predictions based on characterizing the fast-possible and slowest-reasonable combinations of optional task strategy aspects. The heuristic is illustrated with results from a somewhat complex laboratory task.

Introduction

Two Traditions in Human Performance Modeling

Two communities have been interested in modeling and predicting human performance. The *Human Factors Engineering* community has long applied methods for predicting human performance for the purpose of arriving at better designs of systems. For example, past applications have been based on methods and modeling tools such as SAINT and HOS (McMillan, Beevis, Salas, Strub, Sutton, & Van Breda, 1989; Elkind, Card, Hochberg, & Huey, 1989). These approaches are based on analyzing the task that the system operator performs, using systematic task analysis methods that have developed over many years of practical experience in system analysis and design (Kirwan & Ainsworth 1992, Beevis, Bost, Doering, Nordo, Oberman, Pain, Schuffel, & Streets, 1992). In addition, modeling tools use well-established theoretical concepts from human information-processing to generate performance predictions. Overall, this approach has been successful enough that considerable effort has been expended to implement computer-based tools for constructing and using models of human performance in system design.

Concomitantly, the *Cognitive Psychology Research* community has developed a new generation of concepts for modeling human cognition and performance, which show promise of enabling the modeling of human performance to be done with considerably more detail and precision, and also in a more theoretically unified and coherent framework than traditional human information-processing theory. These more advanced approaches are based on computational modeling packages that implement an overall structure for human cognition, a *cognitive architecture*, analogous to the hardware architecture of a computer. Within such a framework, models for a specific task, or type of task, can be implemented. Since the focus has been on developing the scientific basis of the architectures, rather than practical application, the tasks chosen for study

have not typically involved actual systems, and systematic methods of task analysis have not been applied.

Practical Value of Predictive Modeling

The further development of predictive modeling methodology in Human Factors work should have a strong impact on system design because it will enable the design of systems that will perform well with considerably less cost and greater success than the current standard methodology that centers on empirical user testing. User testing is necessarily slow and expensive, and to be most accurate, requires a fairly complete prototype or mockup of the system under design. In contrast, predictive models can produce results on human performance while a future system is still in the earliest design phase, as when SAINT-family models are applied during functional analysis and function allocation. To use the models at their greatest level of precision requires only a detailed design specification; no mockups, prototypes, or human testing are necessary in order to evaluate a design early enough to get the system design in the right ballpark before the necessary and costly next steps of actual user testing. Current experience with newer approaches to performance modeling such as MIDAS (Hoecker, Roth, Corker, Lipner, & Bunzo, 1994; Smith & Tyler, 1997), and GOMS (John & Kieras, 1996a,b) is especially encouraging. It would seem that applying the new cognitive architectures from the Cognitive Psychology Reseach community to the work of Human Factors Engineering would be an obvious and direct step. However, the two communities have been severely compartmentalized; despite the shared interest in human performance modeling, there is a huge gap in both theory and practice between the researchers building sophisticated cognitive models and the human factors designers of large-scale systems.

Purpose of this Chapter

This chapter is an effort to begin bridging the gap. We will focus on certain problems that arise in applying a modern cognitive modeling approach to predicting performance in somewhat complicated tasks. Our cognitive modeling work has revealed a deficiency in both task analysis methodology and cognitive modeling methodology, and we propose an initial solution to the problem, and point to how the more general issues could be addressed.

The remainder of this chapter first describes the general approach for using cognitive architecture in system design, including what is required in order to apply a cognitive architectures to predict human performance in a system design setting. Then is presented the critical problem of identifying the task strategy to be used in the model, which is complicated by the presence of optional aspects of how the task is performed. We then present a solution to the problem of strategy options, the *bracketing heuristic*, and a test of its application. We conclude with remarks about how the bracketing heuristic can be applied, the requirements it places on task analysis, and the relation of cognitive modeling and task analysis in general.

Using Cognitive Architectures in System Design

Task-Independent Architecture and Task-Specific Procedures

Currently, the three most important cognitive architectures relevant to this chapter are ACT-R (Anderson, 1993), SOAR (Laird, Newell, & Rosenbloom, 1987), and EPIC (Kieras & Meyer, in press; Kieras, Wood, & Meyer 1997, Meyer & Kieras, 1997a, b, in press). All of these involve representing the "how to do it" knowledge (procedural knowledge) for a task with a set of production rules, a simple and elegant formalism. A production rule representation is a set of IF-THEN rules which the architecture mechanisms "run" in order to perform the task. Thus, the

structures and mechanisms postulated in the architecture represent the fixed or constant mechanisms of human cognition and performance, while the production rules represent the task-specific "programming," analogous to the distinction between computer hardware and software. In contrast to earlier information-processing models of human performance, the theoretical significance of these architectures is that they make a clear distinction between the fixed versus the task-specific aspects of behavior, and so provide a software-reuse capability in constructing models of human performance — if the architecture is accurate, then only the new task-specific components need to be specified to generate performance predictions for a new system.

Current proposed architectures differ in their detailed assumptions about the components of human cognition and performance. However such differences are not relevant for the purposes of this chapter. Thus, this chapter makes use of only the EPIC architecture. Nevertheless, the main points herein apply equally to models constructed with any other architecture for modeling human performance in detail and predictively.

Requirements for Architecture-Based Predictive Modeling

More specifically, in order to construct and apply an architectural model of human performance, three elements are necessary:

- ***A specified architecture.*** There must be a specified architecture that represents the fixed, constant, human abilities that generalize across tasks. The architecture can be extremely simple, as in those underlying GOMS methodology (see John & Kieras, 1996b), or very complex, as in ACT-R (Anderson, 1993) or SOAR (Laird, Newell, & Rosenbloom, 1987). In addition, the architecture must include some way to represent task-specific parameters of architectural components in addition to task-specific procedural knowledge. For example, a specific task might involve specialized symbols on a display that will take some amount of time to recognize. These recognition times will need to be represented in the architecture by numerical values of parameters chosen for the task domain, even if the recognition is assumed to be handled by standard components of the architecture.

- ***A representation of task strategy.*** There must be some representational scheme, such as production rules, through which the assumed procedural knowledge for performing a task, here called a *task strategy*, can be stated and used along with the architecture to generate predictions about task activities and performance. This representation is essentially the "programming language" for the architecture; it is the major mechanism that adapts the fixed architecture to deal with specific tasks.

- ***A strategy-identification methodology.*** Third, there must be a methodology for identifying what strategy will be used to perform the task. This strategy can then be expressed in the representational scheme. The strategy-identification methodology, and the problems in developing it, are one major focus of this chapter.

If all three of these elements can be developed and fielded appropriately, cognitive architecture models could be used routinely in system design to predict human performance for a particular system design. The steps in this routine application would be: (1) Determine and represent the task-specific processes and their parameters; (2) Identify a strategy for performing the task, and then program the architecture to follow the task strategy. (3) Run the resulting model through a selected set of representative task situations to obtain predicted human performance. (4) Evaluate a proposed system design by comparing the predicted performance against either performance requirements, or the performance predicted for alternative system designs. (5) If necessary, revise

the system design, and repeat the assessment.

The Strategy-Identification Problem

This proposed process for routine application of cognitive modeling sounds straightforward, and simply echoes conventional wisdom in the application of simpler models (e.g. Card, Moran, & Newell, 1983, John & Kieras, 1996a; McMillan et al., 1989; Elkind, Card, Hochberg, & Huey, 1989). However, identifying the task strategy is the stumbling block. Even in putatively simple laboratory tasks, the task strategy can be very non-obvious, and the difficulties of understanding expert performance in the real world are legion. The extreme detail and precision of modern cognitive architectures exacerbates the problem because it is even harder to identify a detailed strategy than a coarse-grained one. Finally, we know from certain long-standing formal results in computation theory that veridical identification of task strategy is in principle impossible (Moore, 1956), so strategy identification is at best only a heuristic process. What heuristic knowledge can be brought to bear on strategy identification from cognitive modeling and human factors?

Cognitive Modeling Practice

In most academic cognitive modeling research, the methodology for identifying the task strategy is intuitive, informal, and normally unstated. Typically, the researcher makes an intuitive guess about what the task strategy is, and then sees if the predictions generated from the assumed strategy and architecture fit the data. If not, the researcher modifies the strategy or the architecture and repeats until a good fit is obtained between the data and the predictions.

During the data-fitting process, the researcher tinkers with a model until its predictions fit the data. The model "predicts" only in the statistical sense of the term; conceptually, the process is fundamentally post-hoc. The scientific lessons learned from this heuristic exercise are based on the vicissitudes experienced in achieving the fit. For example, if the model can be made to fit only with implausible assumptions, the plausibility of the proposed architecture is seriously weakened. If modifications to the architecture are required for a good fit, then the question becomes whether these changes are general and lead to a more accurate architecture, or whether they are purely ad-hoc. On the other hand, if applying the architecture to a variety of task domains reveals recurring patterns of strategies, then important lessons have been learned about the "software" of human performance as well as the "hardware" (see Meyer & Kieras, in press). Ultimately, the research endeavor is deemed successful if the lessons learned from the data-fitting exercise are generalizable and useful.

Application Requires True Prediction

However, in trying to predict performance of a human-machine system during system design, the truth is unknown; there are no data to be fit because the system has not yet been constructed. The modeler must construct a model that *truly* predicts the human-machine performance of a system that does not exist - real prediction is required, not a statistical account of existing data. Granted, some basic parameters may require empirical measurement, such as how long it takes to interpret symbols on a display screen. Nonetheless, the predictive model must produce results similar to what full-scale measurement of an actual system would yield, but using a simulated system and simulated users. Consequently, the task strategy cannot be determined simply by post-hoc data-fitting tinkering. Rather, the analyst must try to *guess* the task strategy in advance by considering the user interface of the proposed system and the overall task requirements that the user is trying to meet.

These need for a priori determination of the task strategy poses great problems. Practical system design involves tasks of high complexity, and the strength of cognitive modeling supposedly stems from its ability to deal with complex tasks. Yet it would seem unlikely that cognitive modelers can perform seat-of-the-pants task analysis fast enough, accurately enough, or reliably enough to yield a practically useful model in a usefully short time. Thus a crucial future objective for both the scientific and practical application of a cognitive architecture is to develop a priori strategy-identification methods that are powerful, accurate, and reliable enough for practical application.

Will Task Analysis Methodology help?

Given that task analysis in human factors practice has a long and successful history (Kirwan & Ainsworth, 1992; Beevis et al., 1992), conventional task analysis methods should be applicable to identifying task strategies for use in cognitive models. For example, task-flow models like SAINT can be applied directly to a fairly high-level and undetailed task analysis (Laughery, 1989). More detailed models like HOS (Harris, Iavecchia, & Dick, 1989) can be routinely applied when the exact sequence of user actions or user procedures is specified in selected situations or scenarios. Many of the techniques in sources like Kirwan and Ainsworth (1992) and Beevis et al. (1992) supply methods and notation for identifying and recording task procedures and other aspects of the task situation.

However, there is a historical gulf between cognitive modeling and human factors methodology. The considerable experience in practical task analysis has yet to be applied in a systematic fashion to the construction of computational cognitive models. Conversely, the modern approach to computational cognitive architecture modeling is not at all in the mainstream of human factors practice. In short, human factors task-analysis experts do not build architected cognitive models, and cognitive modelers do not apply the established and principled methods of task analysis.

The Problem of Optional Aspects of Task Strategy

The thesis of this chapter is that fully exploiting modern cognitive architectural modeling requires some innovation in how we do both task analysis and modeling. The reason is that human task strategies can be more difficult to identify than our previous experience with simple architectures and straightforward tasks would lead us to believe. More specifically, in our own work (Meyer & Kieras, 1997a, b), we have seen that task performance is very heavily influenced by *optional* aspects of task strategy: These are aspects of how the task is performed that are at the discretion of the human operator, being constrained neither by the task requirements nor the cognitive architecture. These optional aspects of task strategy can occur and cause powerful effects even in very elementary tasks which have traditionally been assumed to depend solely on the architectural structure of the human information-processing system (see Meyer & Kieras, in press). Moreover,humans can be remarkably subtle, creative, and even confused in exactly how they choose to do a task. These considerations pose the human-factors analyst with a fundamental question: how can performance be predicted when the analyst is forced to guess about how clever or confused the future users of a system will be?

The remainder of the chapter will help answer this question. Here we present an example of an interesting task in which strategy options are involved when modeled with the EPIC architecture. Our work shows how performance can be predicted by *bracketing* the range of performance produced through the different strategy options. However, first a very brief introduction to the EPIC architecture and its representation of task strategies is necessary.

The EPIC Architecture

Figure 1 presents the overall structure of the EPIC architecture. Many details of EPIC are irrelevant to this chapter, and so will not be presented here; the reader is referred to Kieras, Wood, and Meyer(1997), and Kieras and Meyer (in press) for detailed descriptions of the architecture, and more complete information is also available from the authors. For present purposes, it will suffice to say that EPIC postulates separate processors for perceptual, cognitive, and motor systems,

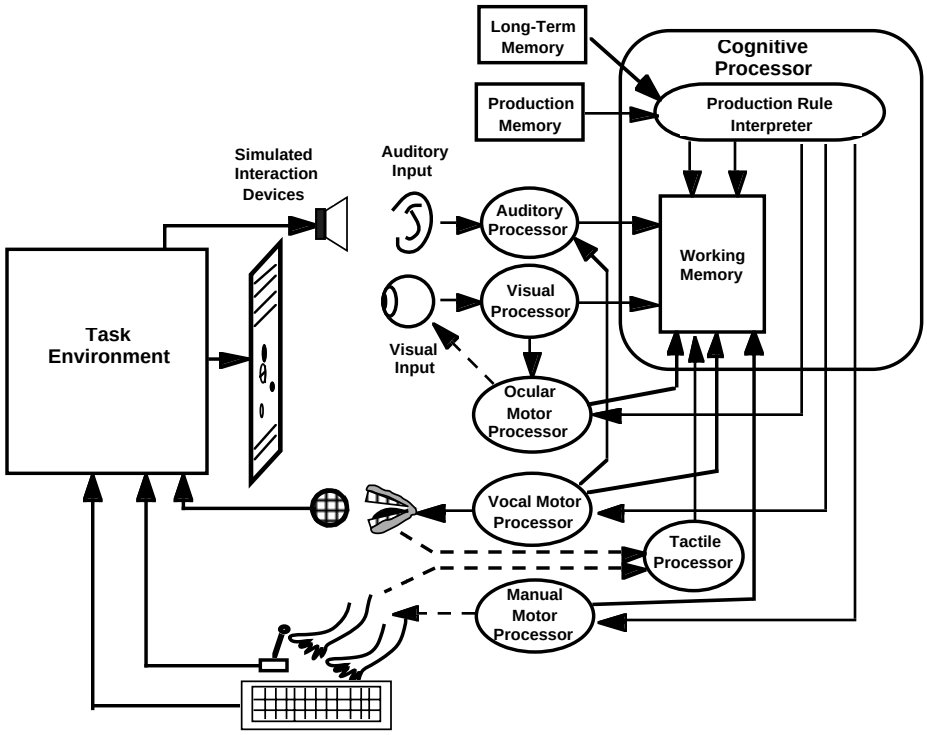


Figure 1. Overall structure of the EPIC architecture simuluation system. Task performance is simulated by having the EPIC model for a simulated human (on the right) interact with a simulated task environment (on the left) via a simulated interface between sensory and motor organs and interaction devices.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

which operate in parallel with each other, and interact to produce simulated human performance. Task strategies are represented as sets of production rules executed by the cognitive processor. The perceptual processors rely on parameter estimates for task-specific encodings such as the time required to recognize display symbols. The distinctive features of EPIC concern some of the specific assumptions made about the cognitive processor, the relatively rich characterization of the perceptual and motor mechanisms that are capable of many parallel processing activities, and the representation of executive processes that oversee multiple-task performance and coordinate parallel activities within a single task. An important objective of our work has been to fit human performance data in precise quantitative detail; we have found that the quantitative properties of data, in conjunction with quantitative constraints imposed by the architecture, serve as very powerful constraints on the task strategies required to produce performance that matches the data.

EPIC's cognitive processor allows fully parallel processing, and the perceptual and motor mechanisms permit a larger set of parallel activities than most architectures have previously entertained. For example, recognition of a visual stimulus object can occur while the eye is being moved to the next object. The physical execution of a movement can be overlapped with the motor programming for the next movement. The hand might be prepared for a movement well in advance of when the movement will be fully specified or executed. Our work has repeatedly revealed how such extreme parallelism is required to fit task data (Meyer & Kieras, in press); if full advantage of this parallelism is taken and task activities are overlapped as much as possible, performance will be much faster than if more conservative strategies are followed.

Identification of Task Strategies in EPIC modeling

In Kieras, Wood, and Meyer(1997) we applied EPIC to model telephone operator task times. We found that not only did we have to identify the required task procedures and implement them as task strategies, but we also had to devise a set of *modeling policies* that specified optional strategy features, such as which processes the task strategy would overlap. Since we could not choose a single policy based on either the task analysis or the EPIC architecture, we explored several of the large number of possibilities by constructing strategies according to selected combinations of modeling policies, and then comparing the model predictions to observed task performance. We discovered that all of the EPIC models using these strategies were usefully accurate in predicting the data, but we also observed that some policies resulted in models that overpredicted the task times, while others underpredicted. We were encouraged: predictions based on the different modeling policies had fallen both above and below the target data, but were also fairly close to them. In what follows, we will describe an extension of this approach, using a much more strategically complex task as an example.

The Ballas Task and Results

Now that the basics of the EPIC architecture have been presented, we present an example of how optional aspects of task strategy appear in our EPIC model for a complex task. Here we have applied EPIC to predicting performance using data collected by James Ballas and his collaborators at the Naval Research Laboratory (Ballas, Heitmeyer, & Perez 1992a, b). The experimental task was a dual-task paradigm in which subjects had to track a target with a joystick, and concurrently classify other targets presented on a radar-like display. Figure 2 shows a sketch of the dual-task display; the *tracking task* is performed in the right-hand window, and the tactical classification task (termed the *tactical task* hereafter) in the left-hand window. During these tasks, "blips" would appear on the tactical display, then change color, whereupon the subject had to classify the blips as being hostile or neutral as fast as possible according to a set of prespecified decision rules. Each response to a blip consisted of two keystrokes, one to identify the blip by its "track number", the second to designate whether it was hostile or neutral. Periodically, an on-board computer would take over the tactical task and classify the blips automatically, leaving the subject free to perform the tracking task by itself. After a time, the computer signaled the subject to resume the tactical task and classify the blips manually.

The basic effect observed by Ballas et al. (1992a,b) appears in Figure 3, which shows the observed reaction times of the first and second keystroke responses for each blip color-change event following the resumption of the manual tactical task. Responses to the blips were slower for a period of time after task resumption, compared to a matched set of blips events during later steady-state performance. Thus there was an interesting *automation deficit* associated with getting back into the tactical task when the subject had to resume performing it manually.

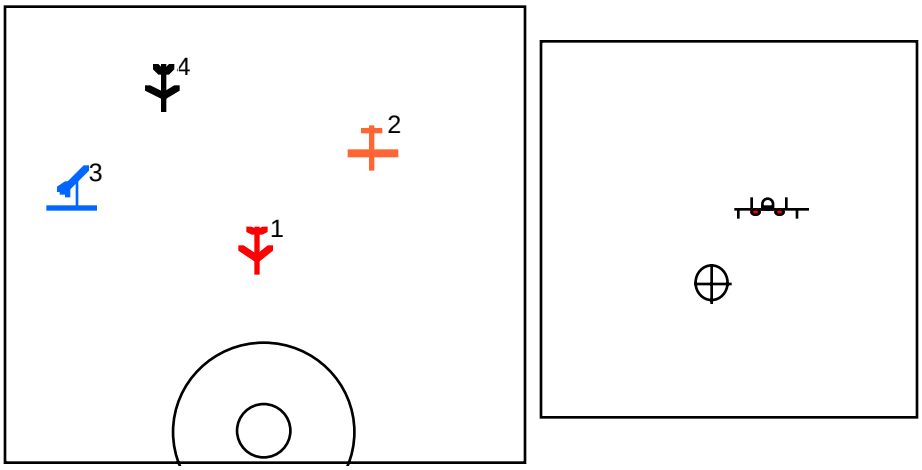


Figure 2. Sketch of the display for the graphical keypad interface in the Ballas et al. task. The tracking task is performed in the window on the right; here the aircraft icon moves around, and the joystick is used to try to keep the cross-hairs on it. The tactical classification task is performed in the window on the left. Here the center of the small circle at the bottom of the display represents the "ownship" point; the "blips" appear in the display and generally move toward the bottom. Initially they are black, but then change color and must be responded to, with red meaning confirm as hostile, blue meaning confirm as neutral, and amber meaning classify based on the speed and direction of the blip. Each blip is identified by a "track" number. Two keystrokes are made on the keypad, one for the hostile/neural designation, the other for the track number.

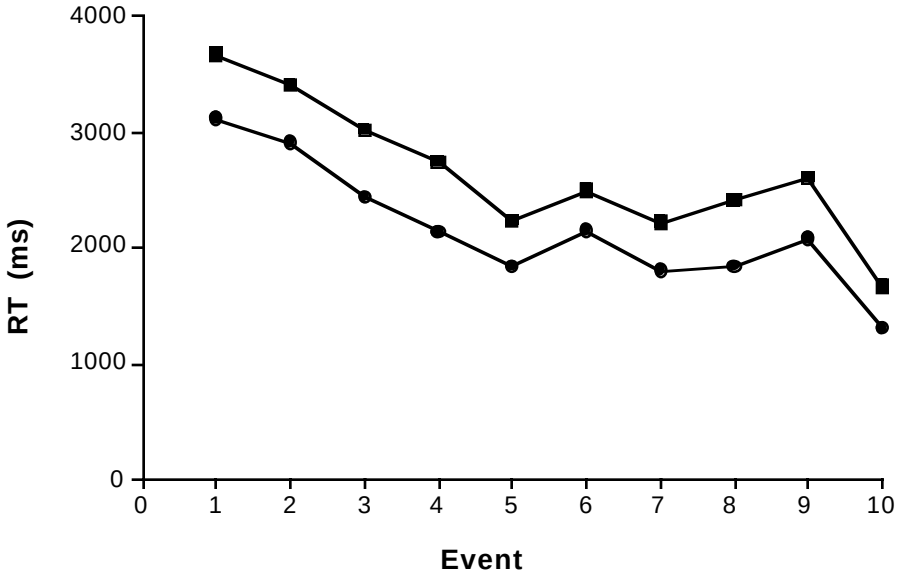


Figure 3. Observed reaction times in the Ballas task for each event after tactical task resumption. The lower curve, plotted with circles, are the times for the first keystroke, the higher curve, plotted with squares, are for the second keystroke. The automation deficit effect appears as longer times for the first few color-change events after the task is resumed (Events 1-3) compared to a matched set of events during steady-state task performance (Events 7-9) .

Ballas Task Strategy

Required and optional aspects of the task. The details of our work on modeling the Ballas task will be presented elsewhere. For present purposes, the main point is that getting a close fit to the empirical data from the model requires a subtle and complex strategy which is modulated in real time depending on the task workload. Figure 4 shows the Ballas task strategy as a hierarchy of tasks and subtasks that we formulated intuitively by the usual cut-and-try approach practiced in most cognitive modeling. Logically, the Ballas task requires alternating between the tracking task and the tactical task according to some task-switching rule, and the tactical task requires choosing a blip and making the appropriate responses to it. However, the specific task-switching criterion, the structural relationships of the task processes, and the extent to which they can be overlapped are constrained by neither the logical requirements of the task nor the EPIC architecture. There are a variety of options that could be chosen by the subjects. We arrived at a specific combination of these options that seems empirically and theoretically plausible by iteratively proposing strategies and comparing their performance to observed data. Of course there is no guarantee that our final inferred strategy is the actual strategy followed by the subjects, but because of how difficult it was to fit the data quantitatively, we doubt that there are many other possibilities within the EPIC architecture.

Strategy representation. Despite these uncertainties about the correct strategy, the representation of possible strategies was straightforward. That is, different possible strategies were programmed with production rules following a set of programming conventions very similar in spirit to the customs of "structured" or "modular" programming, and similar in detail to both the style rules presented by Bovair, Kieras, and Polson(1990) for writing production rule implementations of GOMS models, and also the modeling policies in Kieras, Wood, and Meyer (1997). The strategies were programmed using specified patterns, or templates, that were simply adapted to the specific task requirements as needed. So once the basic organization of the strategies was decided, their production rule representation was created in a very systematic and structured fashion.

Task strategy organization. The strategy shown in Figure 4 can be described in more detail. The dual-task executive supervises the two main subtasks, and also a third auxiliary task that can monitor events on the tactical task display even when the tactical task is not be performed. The dual-task executive applies alternative rules for when to switch tasks, depending on ancillary contextual details. According to one such conservative rule, the dual-task executive waits until some tactical-task blip changes color before suspending tracking and starting the tactical task. Alternatively, the dual-task executive sometimes uses a more enterprising rule to anticipate the color-change events by switching to the tactical task when a threatening blip gets close to the "ownship" circle, and thus is likely to change color soon; the eye is placed on the threatening blip, so the response to its color change can be made more quickly.

Performing the tactical task itself involves an executive process that coordinates three subtasks, one to select a blip for processing, a second to select and produce the hostility designation response for the selected blip, and a third to select and produce the target ID (track number) response for the selected blip. Under EPIC, these three subtasks can be running simultaneously depending on the overlapping policy implemented in the task strategy. If the task strategy overlaps the processing heavily, then performance will be very fast; if not, performance will be substantially slower.

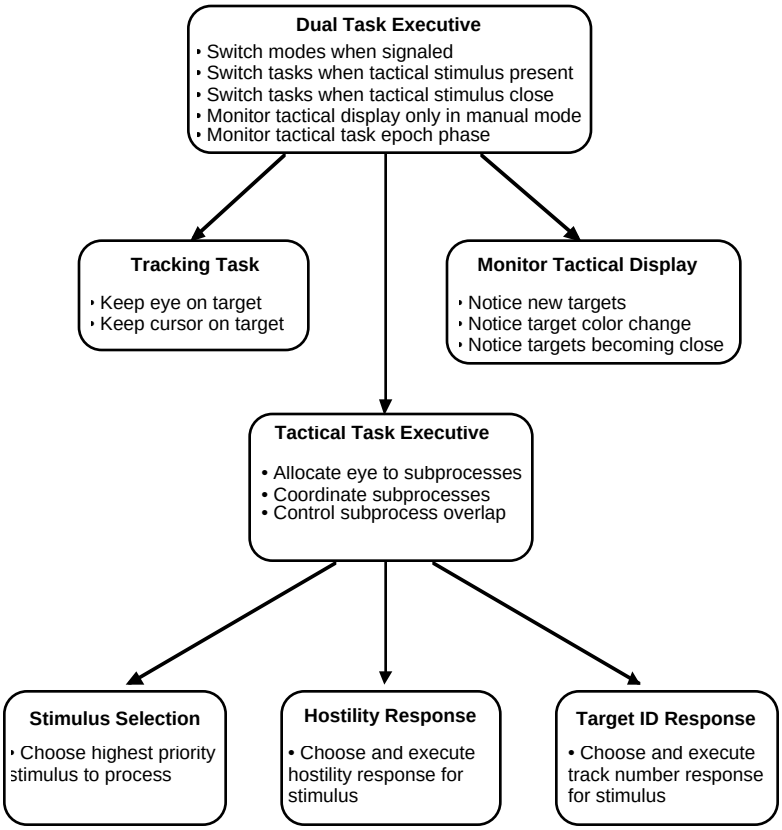


Figure 4. Hierarchical structure of the task strategy used in the models for the Ballas task. Each box represents a task process that performs the functions listed in the box. The executive processes control the task processes below them.

Generally speaking, once basic perceptual and motor delays have had their effect, the performance speed depends primarily on the extent to which the task processes are overlapped by the task strategy.

Dynamic modulation of optional aspects. To fit the observed task data, we had to postulate that subjects followed the task strategy shown in Figure 4, but dynamically modulated the dual-task executive's task-switch rule and the degree of tactical-task process overlap as a function of current workload. During high load periods such as those for events 1-3 and 7-9 in Figure 3, the task strategy is more aggressive with overlapping and anticipation, while during lower-load periods, less overlapping and anticipation is used.

The resulting model using the dynamically modulated task strategy fits the data very well, as shown in Figure 5. This model will be termed the *Fitted Model* in the rest of this chapter. For present purposes, the Fitted Model serves to show that EPIC is a reasonably successful architecture under the usual scientific criteria that allow the task strategy to be iteratively modified to fit the data.

Strategy options complicate prediction. Despite the apparent success of this modeling with EPIC, there is a serious limitation with such work: The fact that one can program models using a cognitive architecture to fit data accurately does not mean that models built with the architecture will be useful in system design. There are many technical obstacles to making model-based predictions in a practical context, but the chief problem of concern in this chapter is dealing

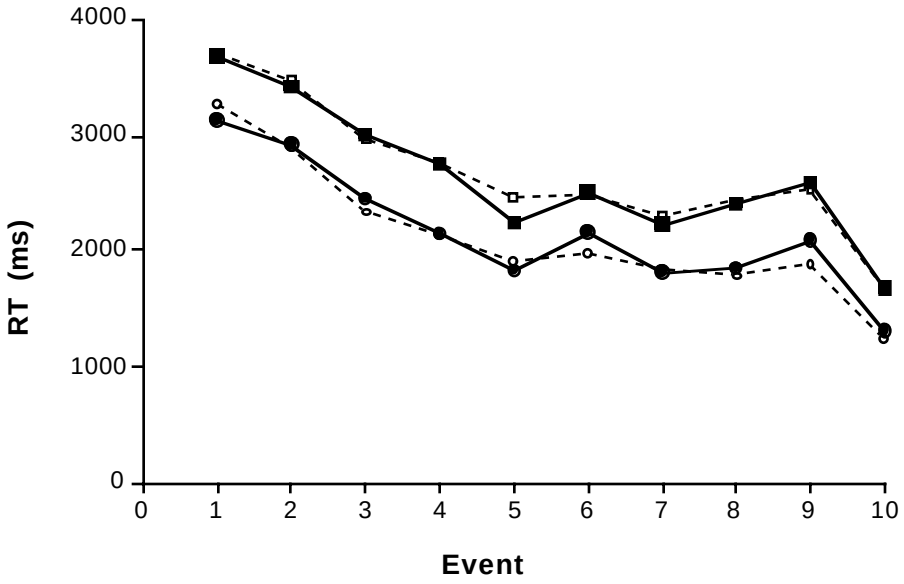


Figure 5. Observed and predicted reaction times in the Ballas task based on the Fitted Model for each event after tactical task resumption. Observed times are shown as solid lines and points, predicted times as dotted lines and open points. The lower curves (circles) are the times for the first keystroke, the higher curves (squares) for the second. The close fit was obtained as a result of the task strategy dynamically modulating the overlapping and anticipation options as a function of workload.

with the optional aspects of the task strategy. These must be resolved before practical model-based predictions are possible. For example, the modulation decisions in the Fitted Model are not dictated by either the task requirements or the architecture constraints, but rather depend on how the subjects chose to do the task strategically. The problem for practical prediction is to determine how enterprising the system operator will be: Will operators use a task strategy that aggressively optimizes performance speed, or will they take a more leisurely approach? In combat, it would be reasonable for operators to try to work rapidly, but what if they have to function for hours at a time? Wouldn't they slow down to a sustainable level of effort? These questions are much more difficult to deal with than the typical task analysis, because their answers cannot be derived in any obvious a priori way from the task requirements, system design, or cognitive architecture. Rather, such strategy issues involve as yet unknown aspects of human metacognition.

The Bracketing Heuristic

Given these considerations, this chapter provides a basic new insight: rather than trying to guess the actual task strategy, it is easier and more reliable to characterize the extremes of the possible task strategies, using the following *bracketing heuristic*: First, identify a *base strategy* for performing the task, dictated by the logical requirements of the task and a systematic approach to representing the strategy in a well organized and structured form, and which incorporates plausible estimates for important task-specific parameters. Second, define a *slowest-reasonable* version of the base strategy; this strategy consists of nominal adherence to the task requirements, but without use of enterprising strategy options. Such a task strategy is neither haphazard nor "lazy," rather, it is deliberate and unhurried. Third, define a *fastest-possible* version of the base strategy, which given the limits represented by the cognitive architecture, exploits that architecture to its fullest, to produce the fastest performance. According to the bracketing heuristic, actual human performance on the fielded system should lie somewhere between the extremes of the fastest-possible and

slowest-reasonable strategy. Exactly where actual performance lies will depend on the level of training, stress, motivation, and fatigue, as well as the extent to which the operators are clever, enterprising, or simply lucky in their choice of strategy. Thus, instead of trying to guess what specific optional strategy the operators will devise, we can simply bracket their performance.

A Test of the Bracketing Heuristic

To test the viability of the bracketing heuristic, we first applied it to the same data set that we used originally to formulate the Fitted Model. Next we applied it to a new set of data collected for a similar version of the task. In each case, we used the Fitted Model as the base strategy, and from it created fastest-possible and slowest-reasonable versions that worked for both cases. In the first case, we were bracketing data that we already had examined extensively, but in the second case, we did not even examine the data until the bracketing predictions had been obtained. Thus, we were able to approximate the situation of using a cognitive model predictively.

The Bracketing Strategies

Of course, the bracketing heuristic has to be elaborated in the context of a multitask situation like the Ballas et al. (1992a, b) task — what is the meaning of "slowest-reasonable" and "fastest-possible" when there are two tasks that must compete for processing resources? For our current test of the bracketing heuristic, we answered this question in terms of the task designated as the highest priority, which was the tactical task. Thus, fastest-possible means that the tactical task is executed as fast as possible regardless of the effect on the lower-priority tracking task. Slowest-reasonable means that the higher priority of the tactical task is honored, but no more so than the overall task instructions explicitly require.

More specifically, the slowest-reasonable task strategy implements a nominal adherence to the task instructions. The instructions imply that the tracking task should be performed until a blip changes color in the tactical task, so under the slowest-reasonable strategy, there is no attempt to anticipate when the tactical task needs attention. Likewise, the instructions imply that when the tactical task is automated, there is no need to monitor the tactical display. The slowest-reasonable strategy therefore lacked the optional enterprising features we have often had to include in our fitted models (Meyer & Kieras, in press). So it omitted movement pre-positioning, advance preparation, and overlapping for the three subprocesses of the tactical task — each response movement had to be complete before the next step in processing for the tactical task began.

In contrast, the fastest-possible task strategy corresponds to the most extreme interpretation of the task instructions. Because the tactical task supposedly has higher priority than the tracking task, this strategy ignores the tracking task if there is anything useful to be done on the tactical task. For example, if there is even a single blip on the tactical display, then under the fastest-possible strategy, the eye is kept on it until it changes color, resulting in faster responding than if the eye had been moved back to the tracking task display. In addition, the tactical display is monitored at all times, even while the tactical task is automated and tracking being performed, because this will speed up identifying relevant blips when the tactical task is resumed. Furthermore, the fastest-possible strategy overlaps the three tactical task subprocesses as much as possible, maximizing advance movement pre-positioning, preparation, and overlapped movement execution.

Bracketing Results

We derived predictions from the two models that implemented the fastest-possible and slowest-reasonable strategies, using the same perceptual parameter estimates as in the Fitted Model. The corresponding predicted and observed times are shown in Figure 6. The observed times are the

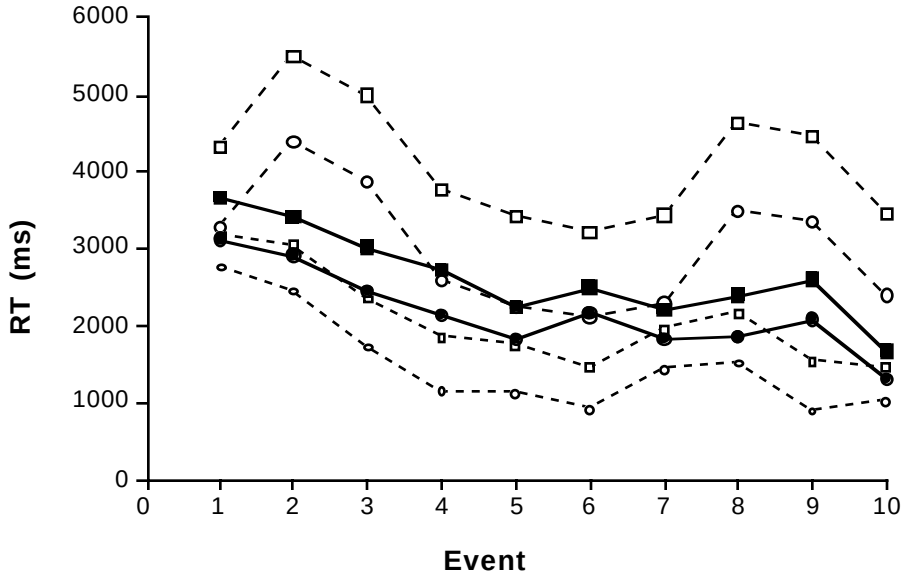


Figure 6. Observed and predicted reaction times in the Ballas task obtained from the bracketing models for each event after tactical task resumption. Observed times are the same data as shown in Figure 3, and are shown as solid lines and points, predicted times as dotted lines and open points. The curves plotted with circles are the times for the first keystroke, the curves plotted with squares are for the second. The slowest-reasonable model is plotted with large open points; the fastest-possible with small open points. Note how the observed times for each keystroke across events are bracketed between the slowest-reasonable and fastest-possible predicted times.

same as those shown before in Figure 3. The predictions based on the fastest-possible and slowest-reasonable strategies do a very good job of bracketing the observed times, both in absolute magnitude and sequential trends across events. We obtained similarly good bracketing results for another version of the Ballas task that involved a touch-screen interface.

After this initial success, we next attempted to bracket a new set of data collected by Ballas and his current co-workers for a task that had the same structure and requirements as the previously modeled one, but which involved different event scenarios and different properties of the targets to be classified. For this purpose, Ballas also collected single-task single-target classification reaction times, in which a single blip was presented, and subjects had to classify it as hostile or neutral by pressing a single key. We used these supplementary data to obtain new estimates of the perceptual processing parameters for the new target properties. This was done with a simple model for doing just this tiny task, and the perceptual processing parameters were adjusted to match the differences in the time to classify the single targets.

Then we generated bracketing predictions from the same fastest-possible and slowest-reasonable models with these perceptual parameters for the new experimental scenarios. Finally, we got the actual data from Ballas's laboratory, and compared the predicted with the observed times.

The predicted and observed reaction times from this a priori bracketing are shown in Figure 7. Again the fastest-possible and slowest-reasonable predictions do a very good job of bracketing the observed performance, both in absolute magnitude and the sequential trends across events. We also obtained similarly good bracketing results for a second scenario in the same experiment. Since we had not previously modeled these data, we are highly encouraged about the value of the bracketing heuristic. We hope to conduct more ambitious applications of it in the future.

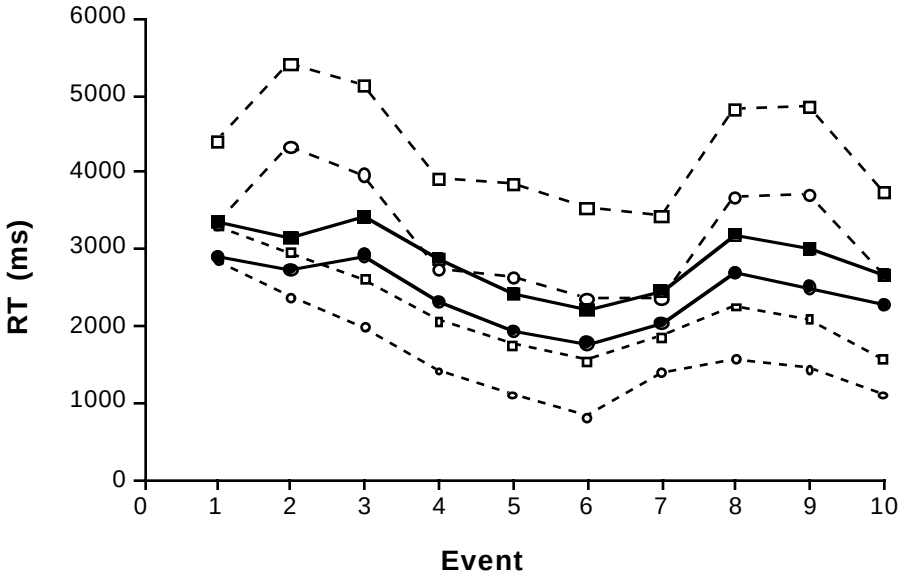


Figure 7. Observed and predicted reaction times in the Ballas task obtained from the bracketing models for each event after tactical task resumption. Observed times are the new data described in the text, and are shown as solid lines and points. Predicted times as shown as dotted lines and open points. The curves plotted with circles are the times for the first keystroke, the curves plotted with squares are for the second. The slowest-reasonable model is plotted with large open points; the fastest-possible with small open points. Note how the observed times for each keystroke across events are bracketed between the slowest-reasonable and fastest-possible predicted times.

Conclusions

Conclusions about the Bracketing Heuristic

Bracketing as a scientific tactic. These results suggest a tactic for future scientific work in the post-hoc model-fitting mode, in which the goal is to arrive at a well-fitting model by iterative testing. The bracketing predictions could help arrive at a well-fitting task strategy more rapidly. For example, as previously described, our Fitted Model strategy in the Ballas task adjusted the overlapping dynamically as a function of workload. We arrived at this strategy by tediously iterating through a large number of models, finally achieving the insight that an apt strategy had to perform at nearly top speed during high workload, and much more leisurely during low workload.

However, we might have achieved this insight much sooner if we had obtained the bracketing predictions first. For example, in Figure 6, during the high-workload portions of the scenario (events 1-3 and 7-9) performance is close to the predictions from fastest-possible strategy, while during the low-workload times, performance is closer to predictions from the slowest-reasonable strategy. These results support our final conclusion about dynamic overlapping adjustment, but they also would have made the need for dynamic adjustment rather obvious at the outset. Future work with EPIC will allow us to test this a priori constrained approach further.

The bracketing heuristic in system design. The bracketing heuristic could be used during system design as a way to obtain performance predictions for a proposed system despite the indeterminacy about what final strategies the human operators will learn and apply. The fastest-possible and slowest-reasonable strategies correspond to the range of variation that could be attributed to differences in levels of training, experience, or motivation on the part of the ultimate

operators of the system.

For two system designs being compared to see which produces the best performance, if they are consistently ordered in terms of the fastest and slowest task strategies, then the best design can be chosen despite of the unknown future effects of operator strategy.

There is a more interesting case of predicting system performance in which there are absolute requirements for the overall speed of performance of the human-machine system. If the bracketing predictions both fall inside the acceptable range of performance, then the design is acceptable. But if the fastest-possible performance is too slow, then the system design is seriously flawed — no human operator will be able to do the task fast enough. In this situation, the analyst can examine the model and its activity to identify the obstacles to faster performance, and then consider alternative system designs to relieve the problem, such as additional automation, or a different user interface design.

On the other hand, if the fastest-possible performance is adequate, but the slowest-reasonable is not, then the system design is marginal — its success will depend on whether the human operators are able or willing to perform according to a more efficient task strategy. This problematic situation could be addressed by considering the demands of the faster strategies and assessing whether they could be easily and consistently trained, and then reliably executed under actual field conditions. Of course the safest and most robust approach would be to change the system design so that even the slowest-reasonable strategy produces acceptable performance

Required Task Analysis for Bracketing Prediction

Mandatory vs. optional requirements of the task. The bracketing heuristic is a way to obtain performance predictions in spite of variations in human task strategies. However, to devise the bracketing strategies, we need to know the mandatory and optional task demands so that the bounds on the possible ways of doing the task can be defined. This requirement is not normally an objective of task analysis — we are not asking "how is the task done?" but rather "what about the task absolutely must be done?" Providing an answer to the latter question requires a much more thoughtful analysis than simply observing and recording how users operate an existing system, or working through a few specific scenarios of how a system might be operated in the future.

Why don't operators work as fast as possible? For the bracketing heuristic to be most useful, we also need a greater understanding of when people will try to achieve the fastest possible performance and when not. Our models of telephone operator performance suggested strongly that some operators had optimized their performance with respect to energetic, "ergonomic", or fatigue-based criteria in their task strategy, rather than maximizing their performance speed. How can we tell when and in what ways to take account of such considerations when predicting performance?

Relation of Task Analysis to Cognitive Modeling

Using computational models to express a task analysis. Many commonly used task analysis notations are difficult to check for accuracy and completeness due to their informality and reliance on interpretation by the human analyst. In contrast, expressing the result of a task analysis as a computational model would go far toward making task analysis more accurate and complete. By attempting to construct the model, the analyst will gain more insight into what the task requirements actually are. By attempting to run the model, the analyst will be able to check whether the analysis accurately captures a correct understanding of the task. In short, a computational model is an excellent target for task analysis methods. If the task analysis is complete and accurate enough to specify a running computational model of performance, then the essential features of the

task have almost certainly been captured. Furthermore, with appropriate tools, a model-based task analysis approach can reduce the gap between task analysis, system design, performance evaluation, and system implementation: both the task analysis and the system design are represented in the computational model, which can both predict performance and potentially help generate the actual user interface implementation (see Byrne, Wood, Sukaviriya, Foley, and Kieras, 1994).

What is a "formal" task analysis? Traditional task analysis can be fairly informal, because the knowledge and intuition of the audience can make up for vagueness and incompleteness in the analysis itself. But computational modeling requires a very formal representation of a task, and so suggests a distinction between what is and what is not formalizable in a task analysis. That is, understanding a user's task is inherently an informal, intuitive process in which the task analyst attempts to arrive at an understanding of the user's task and situation. Once this understanding has been achieved, it must be recorded and communicated in some notational scheme that should be at least formal enough to be standardized and documented. The task analysis methods developed in human factors thus consist essentially of suggestions about what to observe, measure, or record about the user's activities, and useful human-readable notations for recording the information so collected. The methods are nothing more, nor less, than guides to make the analysis process more systematic, dependable, and communicable, but the analyst's intuition and expertise is still the actual source of the understanding of the user's task. Thus, having a computational cognitive model to express the results of the task analysis would extend the value and testability of this intuitively-derived information by representing it in a more rigorous and executable form.

How reliable is task analysis? Conventional applications of task analysis make only limited claims to rigor and objectivity; it suffices that the analysis helps to develop a good system design. However, if task analysis is to be coupled with computational predictive models for rigorous engineering in system design, it helps to know more about how much the results of task analysis depend on the idiosyncrasies of the analyst and the vagaries of the analysis process. Unfortunately, there has been very little work on assessing the reliability of standard task analysis methods. Clearly, a computational model can be no more reliable than the task analysis used to construct it, and the power of a computational model can be especially dangerous if the reliability of the task analysis from with it came is suspect. This concern has led some researchers to question the value of modeling approaches such as GOMS, but they have failed to understand that the basic reliability problem lies with the task analysis, not with the modeling approach. High priority should be given to studying the reproducibility and reliability of task analysis methods.

Generative models are essential. A major contribution of computational modeling approaches is its potential for overcoming a serious practical limitation in most task analysis approaches. The general pattern seen in typical current task analysis practice is that the analyst considers only a small number of task situations or scenarios, and either records at a detailed level how users operate an existing system, or forecasts at a gross level how a future system might be operated. However, fundamental human performance abilities and limitations can only be addressed at the detailed level of analysis, and for a valid assessment of system performance, a large variety of scenarios must be analyzed. The reason why only a small number of scenarios are studied and detailed analysis is avoided is that a typical detailed task analysis method requires enumerating, by hand, every operator action in every specific scenario for every system design under consideration. The time and labor required are simply prohibitive.

However, computational models like those described earlier in this chapter use a representation of procedural knowledge that allows task strategy knowledge to be expressed as general procedures, rather than simply as lists of action sequences in specific scenarios. Such models are *generative*, in that that can *generate* all possible action sequences in all possible instances of the

task subsumed by the strategy. Thus, once programmed with a general strategy, a computational model can be run to automatically generate predicted operator behavior and performance for any number of representative situations. Modifications to the system design typically require only small modifications to the model, and the same situation descriptions can be then be rerun to quickly obtain a new set of predictions. In addition, the explicit representation of the procedural knowledge in a general form allows the prospective system to be studied and analyzed in considerable depth, such as assessing interface consistency and the potential for transfer of training. The advantages of generative models are presented in more detail in Kieras, Wood, and Meyer(1997), and John and Kieras (1996b).

Can task analysis support modeling directly? Clearly, the effort to apply predictive human performance modeling to system design requires more theoretical and practical work on computational cognitive architectures and overcoming the technical difficulties in programming models using these architectures. However, the real bottleneck is in the task analysis process. It would seem that traditional cognitive task analysis at least focused on the proper subject matter, and so it should dove-tail nicely with the needs of model building. For example, so-called *Hierarchical Task Analysis* (HTA, see Kirwan & Ainsworth, 1992, p. 104ff), the most popular single task analysis method, represents task procedures in ways very similar to GOMS models, which it predates by many years and vastly surpasses in amount of practical application and guidance available to the analyst. However, we don't really know whether the standard task analysis methods will work in support of model building because in fact there is little past experience. What is missing, and badly needed, is a demonstration that one can start with a conventional task analysis such as HTA and then proceed systematically to a usefully accurate computational cognitive model, with no "hand-waving" in between. If so, then prospects are good that task analysis methods and computational models can be combined to increase the range and power of the tools available to help design more effective systems.

Acknowledgement

This work was supported by grants to the authors from the Office of Naval Research, Grant No. N00014-92-J-1173, and Grant No. N00014-96-1-0467 in collaboration with James Ballas, Naval Research Laboratory.

References

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

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. Naval Research Laboratory, Washington, D. C.

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.

Beevis, D., Bost, R., Doering, B., Nordo, E., Oberman, F., Papin, J-P., I., H. Schuffel, & Streets, D. (1992). Analysis techniques for man-machine system design. (Report AC/243(P8)TR/7). Brussels, Belgium: Defense Research Group, NATO HQ.

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.

Byrne, M.D., Wood, S.D, Sukaviriya, P., Foley, J.D, and Kieras, D.E. (1994). Automating Interface Evaluation. In *Proceedings of CHI*, 1994, Boston, MA, USA, April 24-28, 1994). New York: ACM, pp. 232-237.

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

Harris, R., Iavecchia, H.P, & Dick, A.O. (1989). The Human Operator Simulator (HOS-IV). In G.R. McMillan, D. Beevis, E. Salas, M.H. Strub, R. Sutton, & L. Van Breda (Eds.). *Applications of human performance models to system design*. New York: Plenum Press. 275-280.

Hoecker, D.G., Roth, E.M., Corker, K.M., Lipner, M.H., and Bunzo, M.S. (1994). Man-machine Design and Analysis System (MIDAS) Applied to a Computer-Based Procedure-Aiding System. In *Proceedings of the 38th Annual Meeting of the Human Factors and Ergonomics Society*. Memphis, TN, Oct. 1994.

John, B. E., & Kieras, D. E. (1996a). Using GOMS for user interface design and evaluation: Which technique? *ACM Transactions on Computer-Human Interaction*, **3**, 287-319.

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

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

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.

Kirwan, B., & Ainsworth, L. K. (1992). *A guide to task analysis*. London: Taylor and Francis.

Laird, J. E., Newell, A., and Rosenbloom, P.S. (1987) Soar: An architecture for general intelligence. *Artificial Intelligence*, **33**, 1-64.

Laughery, K. R. (1989). Micro SAINT - A tool for modeling human performance in systems. In G.R. McMillan, D. Beevis, E. Salas, M.H. Strub, R. Sutton, & L. Van Breda (Eds.). *Applications of human performance models to system design*. New York: Plenum Press. 219-230.

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 Press.

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 control processes and human multiple-task performance: Part 2. Accounts of Psychological Refractory-Period Phenomena. *Psychological Review*. **104**, 749-791.

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

Moore, E. F. (1956). Gedanken-experiments on sequential machines. In C. E. Shannon, & J. McCarthy (Eds.), *Automata studies*. Princeton, NJ: Princeton University Press.

Smith, B.R., & Tyler, S.W. (1997). The design and application of MIDAS: A constructive simulation for human-system analysis. Paper presented at the Second Simulation Technology and Training Conference, March 17-20, 1997, Canberra, Australia. Available online at http://www-midas.arc.nasa.gov.