CHAPTER 32

MODELING HUMAN PERFORMANCE IN COMPLEX SYSTEMS

K. Ronald Laughery, Jr., Beth Plott, Michael Matessa, and Susan Archer
Alion Science and Technology, MA&D Operation
Boulder, Colorado

Christian Lebiere
Carnegie Mellon University Pittsburgh, Pennsylvania

1 INTRODUCTION

Over the past few decades, human factors and ergonomics practitioners have been called upon increasingly early in the system design and development process. Early inputs from all disciplines result in better and more integrated designs, as well as lower costs, than if one or more disciplines are solely in charge, find out late in the development stage that changes are required, and then call upon the expertise of the other disciplines. Our goal as human factors and ergonomics practitioners should be to provide substantive and well-supported input regarding the human(s), his or her interaction(s) with the system, and the resulting total performance. Total performance includes a number of converging measures, including task latency, type and probability of errors, quality of performance, and workload measures. Furthermore, we should be prepared to provide this input from the earliest stages of system concept development and then throughout the entire system or product life cycle.

To meet this challenge, many human factors and ergonomics tools and technologies have evolved over the years to support early analysis and design. Two specific types of technologies are design guidance (e.g., Boff et al., 1986; O'Hara et al., 1995) and high-fidelity rapid prototyping of user interfaces (e.g., Dahl et al., 1995). Design guidance technologies, in the form of either handbooks or computerized decision support systems, put selected portions of the human factors and ergonomics knowledge base at the fingertips of the designer, often in a form tailored to a particular problem, such as nuclear power plant design or Unix computer interface design. However, design guides have the shortcoming that they do not often provide methods for making quantitative trade-offs in *system* performance as a function of design. For example, design guides may tell us that a high-resolution color display will be better than a black-and-white display, and they may even tell us the value in terms of increased response time and reduced error rates. However, this type of guidance will rarely provide good insight into the value of this improved element of the human's performance to the *overall system's* performance. As such, design guidance has limited value for providing concrete input to *system-level performance prediction*.

Rapid prototyping, on the other hand, supports analysis of how a specific design and task allocation will affect human and system-level performance. The disadvantage of prototyping, as with all human subject experimentation, is that it can be slow and costly. In particular, prototypes of hardware-based systems, such

as aircraft and machinery, are very expensive to develop, particularly at early design stages when there are many widely divergent design concepts. Despite the expense, hardware and software prototyping is an important tool for the human factors practitioner, and its use is growing in virtually every application area.

Although these technologies are valuable to the human factors practitioner, what is often needed is an integrating methodology that can extrapolate from the base of human factors and ergonomics data, as reflected in design guides and the literature, to support system-level performance predictions as a function of design alternatives. This methodology should also bind with rapid prototyping and experimentation in a mutually supportive and iterative way. As has become the case in many engineering disciplines, a prime candidate for this integrating methodology is computer modeling and simulation.

Computer modeling of human behavior and performance is not a new endeavor. Computer models of complex cognitive behavior have been around for over 20 years (e.g., Newell and Simon, 1972; Card et al., 1983) and tools for computer modeling of task-level performance have been available since the 1970s (e.g., Wortman et al., 1978). However, three trends have emerged in the past decade to promote the use of computer modeling and simulation of human performance as a standard tool for the practitioner. First is the rapid increase in computer power and the associated development of easier-to-use modeling tools. People with an interest in predicting human performance through simulation can select from a variety of computer-based tools. For a comprehensive list of these tools, see the Defense Technical Information Center (DTIC) Directory of Design Support Methods (DDSM). The DDSM contains references to human systems integration (HSI) design and interface tools, techniques, databases, guides, and standardization documents. Second is the increased focus by the research community on the development of *predictive* models of human performance rather than simply descriptive models. For example, the goals–operators–methods–selection rules (GOMS) model (Gray et al., 1993) represents the integration of research results into a model for making predictions of how humans will perform in a realistic task environment. Another example is the research in cognitive workload that has been represented as computer algorithms (e.g., McCracken and Aldrich, 1984; Farmer et al., 1995). Given a description of the tasks and equipment with which humans are engaged, these algorithms support assessment of when workload-related performance problems are likely to occur and often include identification of the quantitative impact of those problems on overall system performance (Hahler et al., 1991). These algorithms are particularly useful when embedded as key components in computer simulation models of the tasks and the environment. Third is the integration of those algorithms into cognitive architectures that integrate cognition, perception, and action into a single computational framework that can be applied to a broad range of tasks, from basic laboratory experiments used to validate the architectural mechanisms to predicting operator performance on complex practical tasks (Gray et al., 1997).

Perhaps the most powerful aspect of computer modeling and simulation is that it provides a method through which the human factors and ergonomics team can "step up to the table" with the other engineering disciplines, which also rely on quantitative computer models. What we discuss in this chapter are the methods through which the human factors and ergonomics community can contribute early to system design trade-off decisions.

1.1 Chapter Objectives

In this chapter we discuss some existing computer tools for modeling and simulating human–system performance. It is intended to provide the reader with an understanding of the types of human factors and ergonomics issues that can be addressed with modeling and simulation and some of the tools that are now available to assist the human factors and ergonomics specialist in conducting model-based analyses and an appreciation of the level of expertise and effort that will be required to use these technologies. We begin with two caveats. The first is that we are not yet at a point where computer modeling of human behavior allows sufficiently accurate predictions that no other analysis method (e.g., prototyping) is needed. In the early stages of system concept development, high-level modeling of human–system interaction may be all that is possible. As the system moves through the design process, human factors and ergonomics designers will often want to augment modeling and simulation predictions with prototyping and experimentation. In addition to providing high-fidelity system performance data, these data can be used to constrain, enhance, and refine the models. This concept of human performance modeling supporting and being supported by experimentation with human subjects is represented in Figure 1. In essence, simulation provides the human factors and ergonomics practitioner with a means of extending the knowledge base of human factors and of amplifying the effectiveness of limited experimentation.

The second caveat is that the technologies discussed here are evolving rapidly. We can be certain that every tool discussed is undergoing constant change and that new modeling tools are being developed. We are discussing computer-based tools, and we expect the pace of change in these tools to mirror the pace in other software tools, such as word processors, spreadsheets, presentation and productivity tools, and Internet-based applications. These detailed discussions of several of the modeling tools are included to facilitate better understanding of human performance modeling tools. We encourage the reader to follow citations in this chapter to assess the current state of any tool. Most of these modeling tools have large, active user communities that maintain websites to provide introductory tutorials, software downloads, validated models, and published papers. These resources are invaluable both for the experienced modeler trying to stay abreast of recent developments and the novice user attempting to get up to speed on a new technology.

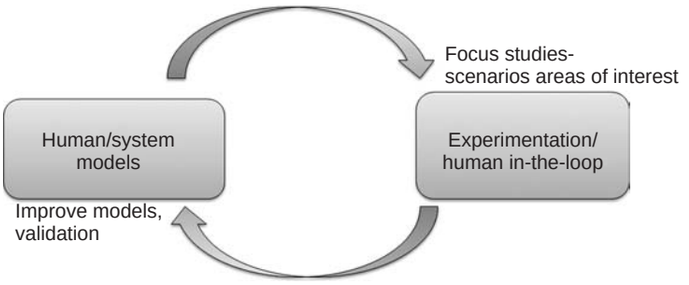

Figure 1 Synergy between modeling and experimentation.

2 QUESTIONS ADDRESSED BY HUMAN PERFORMANCE MODELS

Below are a few classes of problems to which human–system modeling has been applied:

- How long will it take a human or team of humans to perform a set of tasks as a function of system design, task allocation, and individual capabilities?
- What are the performance trade-offs for different combinations of design, task allocation, and individual capability selections?
- What are the workload demands on the human as a function of system design and automation?
- How will human performance and resulting system performance change as the demands of the environment change?
- How many people are required on a team to ensure safe, successful performance?
- How should tasks be allocated to optimize performance?
- How will environmental stressors such as heat, cold, or vibration affect human–system performance?

The list above is a sample rather than an exhaustive list. The tools we discuss in this chapter are inherently flexible and we consistently discover that these tools can be used to solve problems that the tool developers never conceived. To assess the potential of simulation to answer questions, in every potential human performance modeling project we should first determine the specific questions that the project is trying to answer. Then we can conduct a critical assessment of what is important in the human–machine system being modeled. This will define the required content and fidelity of the model. The questions that should be considered about the system include:

1. *Human Performance Representation*. What time or duration of performance is important? How is human performance initiated, and what resolution of behavior is required? What aspects of human performance, including task management, load management, and goal management, are expected? How much is known and constrained about the knowledge and strategies that human users bring to bear on this task?
2. *Equipment Representation.* What equipment is used to accomplish the task? To what level of functional and physical description can and should equipment be represented? Is it operable by more than one human or system component?
3. *Interface Requirements.* What information needs to be conveyed to the humans and when? Is transformation of information required? How often is information updated and monitored?
4. *Control Requirements.* What processes need to be controlled by the human and to what level of resolution? How much attention is required by the human to perform control changes?
5. *Logical and Physical Constraints*. How is performance supported through equipment operability and procedural sequences? What alarms and alerts should be represented?
6. *Simulation Driver*. What makes the system function? The occurrence of well-defined events (e.g., a procedure), the passage of time (e.g., the control of a vehicle), or a hybrid of both?

In using human performance models, perhaps the most significant task of the human factors practitioner is to determine what aspects of the human–machine system to include in the model and what to leave out. By defining the purpose of the model and then answering the questions above, the human factors practitioner will get a sense of what is important in the system and therefore what may need to be represented in a model. Many modeling studies have failed because of the inclusion of too many factors that, although a part of human–system performance, were not system performance drivers. Consequently, the models become overly complex and expensive to develop. In our experience, it is better to begin with a model with too few aspects of the system represented and then add to it than to begin a modeling project by trying to model everything. The first approach may succeed, whereas the second is often doomed. It is also important that the

level of detail is consistent with the types of data that are available.

Additionally, the human factors practitioner should consider the measures of effectiveness of the system that the model should be designed to predict. In building the model, it is important to remember that the goal will be to predict measures of human performance that will affect system performance. Therefore, a clear definition of what is important to performance is necessary. The following aspects of performance measures should be considered:

1. *Success Criteria*. What operational success measures are important to the system? Can these be stated in relative terms or must they be measured in absolute terms?
2. *Range of Performance to Be Studied*. What experimental variables are to be explored by the model? How important is it to establish a range of performance for each experimental condition as a function of the stochastic (i.e., random) behavior of the system?

By asking the foregoing questions prior to beginning a modeling project, the human factors practitioner can develop a better sense of what is important in the system in terms of both aspects that drive system performance and the measures of effectiveness that are truly of interest. Then, and only then, can a human performance modeling project begin with a reasonable hope of success.

In the remainder of this chapter we discuss two classes of modeling tools for human performance simulation, then report on recent efforts to unify those two complementary classes in order to leverage their strengths and alleviate their shortcomings. After discussing each class of modeling tool, we provide specific examples of a modeling tool and then provide case studies about how these tools have been used in answering real human performance questions.

3 CLASSES OF SIMULATION MODELS

Human performance can be highly complex and involve many types of processes and behavior. Over the years many models have been developed that predict sensory processes (e.g., Gawron et al., 1983), aspects of human cognition (e.g., Newell, 1990), and human motor response (e.g., Fitts' law). The current literature in the areas of cognitive engineering, error analysis, and human–computer interaction contains many models, descriptions, methodologies, metaphors, and functional analogies. However, in this chapter we are not focusing on the models of these individual elements of human behavior but rather, on models that can be used to describe human performance in systems. These human–system performance models typically include some of these elemental behavioral models as components but provide a structural framework that allows them to be integrated with each other and put in the context of human performance of tasks in systems.

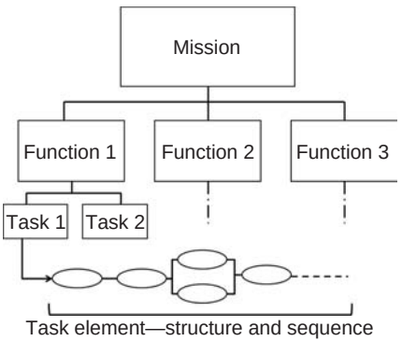

Figure 2 Reductionist models of human performance.

We separate the world of human–system performance models into two general categories that can be described as reductionist models and first-principle models. *Reductionist models* use human–system task sequences as the primary organizing structure, as shown in Figure 2. The individual models of human behavior for each task or task element are connected to this task-sequencing structure. We refer to it as reductionist because the process of modeling human behavior involves taking the larger aspects of human–system behavior (e.g., "perform the mission") and then reducing them successively to smaller elements of behavior (e.g., "perform the function," "perform the tasks"). This continues until a level of decomposition is reached at which reasonable estimates of human performance for the task elements can be made. One can also think of this as a top-down approach to modeling human–system performance. The example of this type of modeling that we use in this chapter is *task network modeling*, where the basis of the human–system model is a task analysis.

First-principle models of human behavior are structured around an organizing framework that represents the underlying goals, principles, and mechanisms of human performance (Figure 3). Tools that support first-principle modeling of human behavior have structures embedded in them that represent elemental aspects of human performance. For example, these models might directly represent processes such as goal-seeking behavior, task scheduling, sensation and perception, cognition, and motor output. In turn, those processes might invoke fundamental actions such as shifts of attention, memory retrieval, and conflict resolution among competing courses of action. To use tools that support first-principle modeling, one must describe how the system and environment interacts with the human processes being modeled. In this chapter we focus on the adaptive control of thought–rational (ACT-R) cognitive architecture (Anderson and Lebiere, 1998).

It is worth noting that these two modeling strategies are not mutually exclusive and, in fact, can be mutually supportive in any given modeling project. Often, when one is modeling using a reductionist approach, one needs models of basic human behavior to represent behavioral phenomena accurately and therefore must draw on

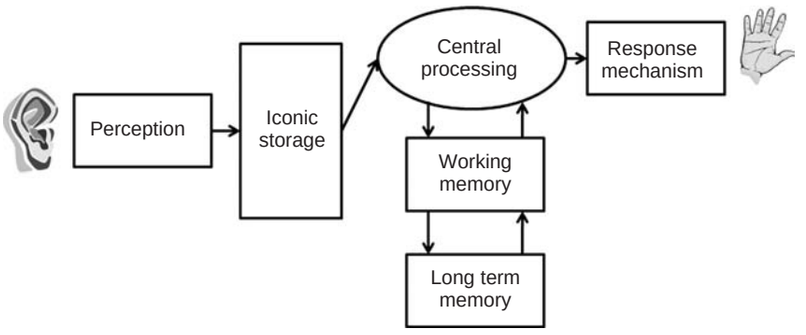

Figure 3 First-principle models of human performance.

elements of first-principle models. Alternatively, when one is modeling human–system performance using a first-principle approach, some aspects of human–system performance and interrelationships between tasks may be more easily defined using a reductionist approach. Both classes of model have been used to model individual and team performance. It is also worth noting that recent advances in human performance modeling tool development are blurring the distinctions between these two classes (e.g., Hoagland et al., 2001). Increased emphasis on interoperability between models has caused researchers and developers to focus on integrating reductionist and first-principle models. In the final section of this chapter we present one such attempt at integrating the ACT-R cognitive architecture with the Improved Performance Research Integration Tool (IMPRINT).

4 REDUCTIONIST APPROACH: TASK NETWORK MODELING

One technology that has proven useful for predicting human–system performance is *task network modeling.* In a task network model, human performance is decomposed into tasks. The fidelity of this decomposition can be selective, with some functions being decomposed several levels and others just one or two. This is, in human factors engineering terms, the task analysis. The sequence of tasks is defined by constructing a *task network*. This concept is illustrated in Figure 4, which presents a sample task network for driving while talking on a cell phone.

Task network modeling is an approach to modeling human performance in complex systems that has evolved for several reasons. First, it is a reasonable means for extending the human factors staple: the task analysis. Task analyses organized by task sequence are the basis for the task network model. Second, task network models can include sophisticated submodels of the system hardware and software to create a closed-loop representation of relevant aspects of the human–machine system. Third, task network modeling is relatively easy to use and understand. Recent advancements in task network modeling technology have made this technology more accessible to human factors practitioners. Finally, task network modeling can provide efficient, valid, and useful input to many types of issues. With a task network model, the human factors engineer can examine a design (e.g., control panel redesign) and address questions such as "How much longer will it take to perform this procedure?" and "Will there be an increase in the error rate?" Generally, task network models can be developed in less time and with substantially less effort than would be required if a prototype were developed and human subjects used. However, as stated before, for revolutionary designs, modeling may not alleviate the need for empirical data collection.

Task network models of human performance have been subjected to validation studies with favorable results (e.g., Lawless et al., 1995; Engh et al., 1998). However, as with any modeling approach, the real level at which validation must be considered is with respect to a particular model, not with respect to the general approach.

4.1 Components of a Task Network Model

To represent complex, dynamic human–system behavior, many aspects of the system may need to be modeled in addition to simply task lists and sequence. In this section we use the task network modeling tool *Micro Saint Sharp* as an example. The basic ingredient of a Micro Saint Sharp task network model is the task analysis as represented by a network or series of networks. The level of system decomposition (i.e., how finely we decompose the tasks) and the amount of the system that is simulated depend on the particular problem. For example, in a power plant model, one can create separate networks for each of the operators and one for the power plant itself. Although the networks may be independent, performance of the tasks can be interrelated through shared variables. The relationships among different components of the system, represented by different segments of the network, can then communicate through changes in these shared variables. For example, when an operator manipulates a control, this may initiate an "open valve" task in a network representing the plant. This could ripple through to a network representing other operators and subsystems and their

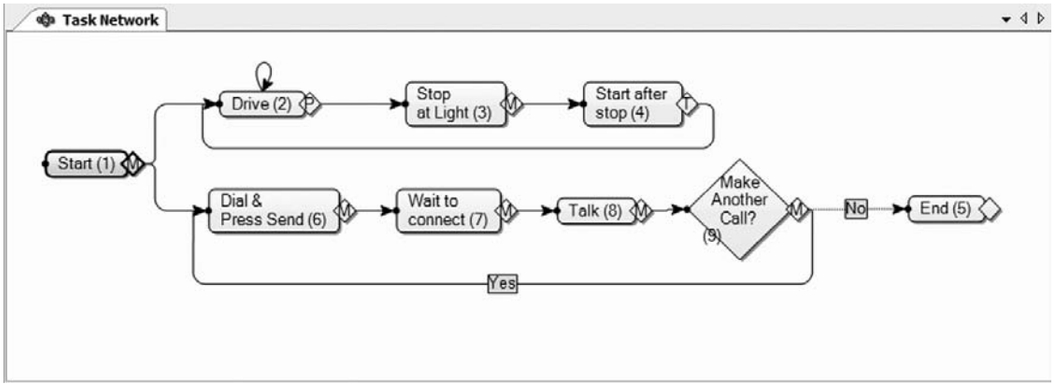

Figure 4 Task network model representing a human driving while talking on a cell phone.

response to the open valve. This basic task network is built in Micro Saint Sharp via a point-and-click drawing palette. Through this environment, the user creates a network as shown in Figure 5. Networks can be embedded within networks, allowing for hierarchical construction. In addition, the shape of the nodes on the diagram can be chosen to represent specific types of activity.

To reflect complex task behavior and interrelationships, more detailed characteristics of the tasks need to be defined. By double clicking on a task, the user opens up the task description window, as shown in Figure 6. Below are descriptions of each of the items on the tabs in this window.

- *Task ID*. This value is an arbitrary number for task referencing.
- *Task Name*. This parameter contains a text string used to identify the task.

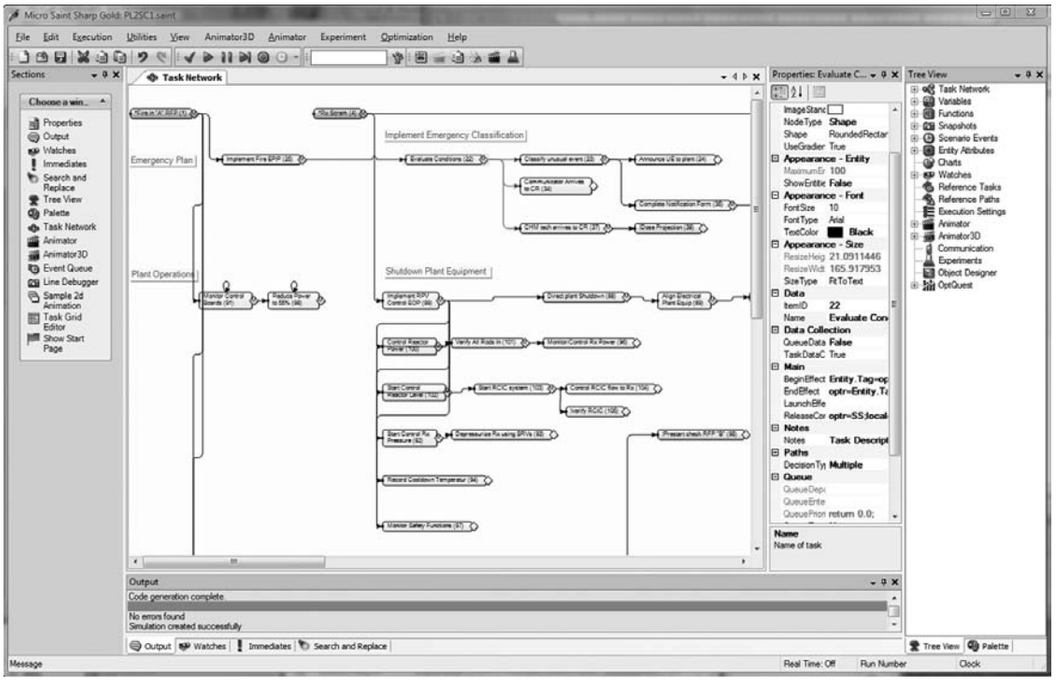

Figure 5 Main window in Micro Saint Sharp for task network construction and viewing.

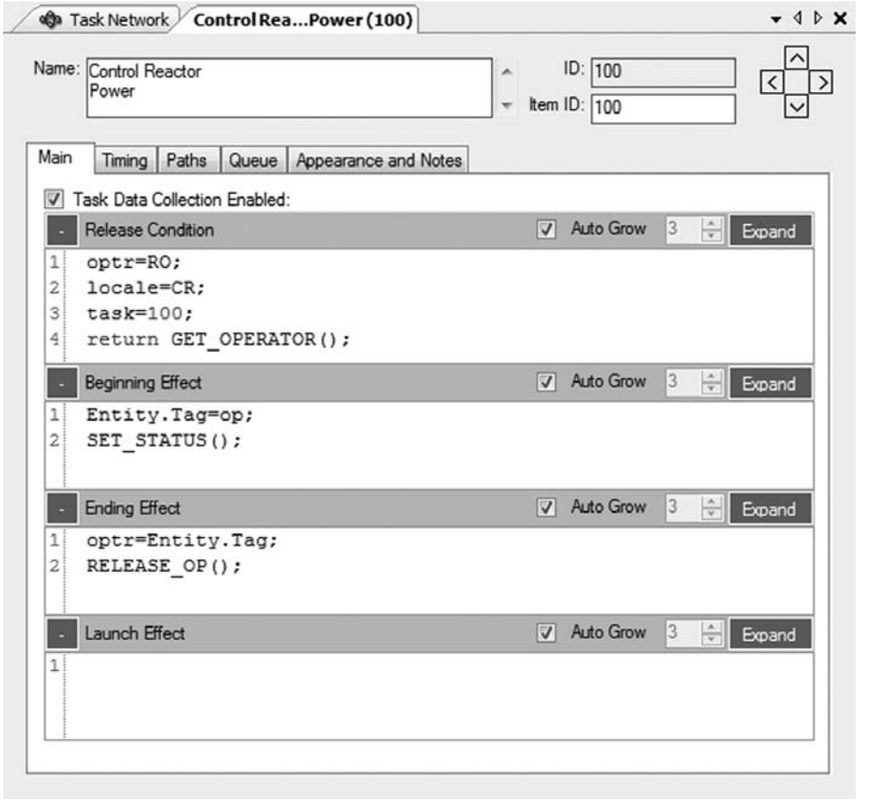

Figure 6 User interface in Micro Saint Sharp for providing input on a task.

- *Time Distribution*. Micro Saint Sharp conducts Monte Carlo simulations with task performance times sampled from a distribution as defined by this option (e.g., normal, beta, exponential).
- *Mean Time*. This parameter defines average task performance time for this task. This can be a number, equation, or algorithm, as can all values in the fields described below.
- *Standard Deviation*. This value contains the standard deviation of the task performance time, assuming that the user has chosen a distribution that is parameterized by a standard deviation.
- *Release Condition*. Data in this field determine when a task begins executing. For example, a condition stating that this task will not start before an operator is available might be represented by a release condition such as the following:

Q1

$$\text{OperatorBusy} == \text{false};$$

In other words, for the task to begin, the value of the variable "OperatorBusy" must be false. This task would wait until the condition was true before beginning execution, which would probably occur as a result of the operator completing the task he or she is currently performing.

- *Beginning Effect*. This field permits the user to define how the system will change as a result of the commencement of this task. For example, if this task used an operator that other tasks might need, we could set the following condition to show that the operator is unavailable while he or she performed this task:

$$\text{OperatorBusy} = \text{true};$$

Assignment and modification of variables in beginning effects are one principal way in which tasks are interrelated.

- *Launch Effect*. This data element is similar to a task beginning effect but is used to launch high-resolution two- (2D) and/or three- (3D) dimensional animation of the task.
- *Ending Effect*. This field contains the definition of how the system will change as a result of the completion of this task. From the previous example, when this task was complete and the

operator became available, we could set the ending effect as follows:

OperatorBusy = false;

At this point, another task waiting for an operator to become available could begin. Ending effects are another important way in which tasks can be interrelated through the assignment and modification of variables.

Another notable aspect of the task network diagram window shown in Figure 5 is the diamond-shaped icon that follows every task. This icon encapsulates data that describe the paths and the associated logic that will be executed when this task is completed. Often, this logic represents a human decision-making process. In that case, the branches align to potential courses of action that the modeled human could select. To define the decision logic, the Micro Saint Sharp user would use the "Paths" tab on the task description dialogue, as shown in Figure 7. There are three general types of decisions to model:

- *Probabilistic*. In probabilistic decisions, the human will begin one of several tasks based on a random draw weighted by the probabilistic branch value. These weightings can be dynamically calculated to represent the current context of the decision. For example, this decision type might be used to represent human error likelihoods and would be connected to the subsequent tasks that would be performed.
- *Tactical*. In tactical decisions, the human will begin one of several tasks based on the branch with the highest "value." This could be used to model the many types of rule-based decisions that humans make, as illustrated in Figure 7.
- *Multiple*. This would be used to begin several tasks at the completion of this task, such as when one human issues a command that begins other crew members' activities.

The expression fields in Figure 7 represent the values associated with each branch. The values can be numbers, expressions, or complicated algorithms defining the probability (for probabilistic branches) or the desirability (for tactical and multiple branches) of taking each branch in the network. Again, any value on this screen can be not simply numbers but also variables, algebraic expressions, logical expressions, or groups of algebraic and logical expressions that would, essentially, form a subroutine. As the model executes, Micro Saint Sharp includes a parser that evaluates the expressions included in the branching logic when it is encountered in the task network flow. This results in a dynamic network in which the flow through the tasks can be controlled with variables that represent equipment state, scenario

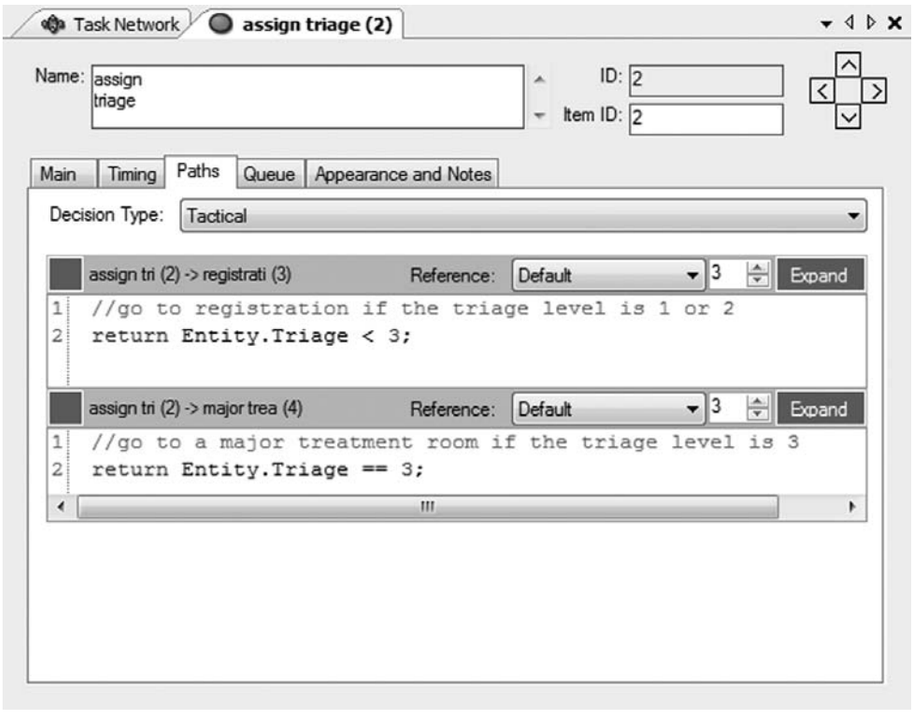

Figure 7 User interface in Micro Saint Sharp for defining task-branching decision logic.

context, or the task loading of the humans in the system, to name a few examples. It is the power of this parser that provides many task network models with the ability to address complex problems.

The research community has been inspired by the Micro Saint Sharp architecture, and several efforts have enhanced the capability of the task network model to more accurately represent theoretical advances. Specifically, Warwick and Santamaria (2009) extended the available decision types described above to a "recognition-primed" decision type (Klein, 1998). This new decision type represents human decisions that are experience driven and in which the subsequent action choices are driven by recognition of aspects of the scenario, rather than by a rule-based process. To implement this decision type, the model develops and maintains an ongoing representation of the human's memory, which either can be preloaded prior to model start or can be populated (i.e., trained) as the model proceeds. Preliminary validation work on this model has been encouraging and provides opportunities for more complete representations of "natural" human behavior.

The Command, Control, and Communications—Techniques for Reliable Assessment of Concept Execution (C3TRACE) tool has also expanded the available decision types to allow branching based on message communication type (digital, face to face, written, etc.) or decision quality (Plott et al., 2004).

There are other aspects of task network model development. Some items define a simulation scenario defining continuous processes within the model and queues in front of tasks. Further details of these features can be obtained from the Micro Saint Sharp *User's Guide* (Alion Science and Technology, 2009). As a model is being developed and debugged, the user can execute the model to test it and collect data. The user can rearrange, open, and close a variety of windows to represent a variety of display modes providing differing levels of information during execution. The simulation speed can also be controlled to include pausing after every simulated task. Typically, during execution the user will display the task network on the screen, and tasks that are currently executing will be highlighted. In this mode, the analyst can get a very clear picture of what events are occurring in what sequence in the model, greatly aiding debugging. Figure 8 presents a sample display during model network animation. Additionally, 2D and 3D animator modes are available. In these modes, the user can create a graphical representation of the system. Changes on the graphical background can be tied to the task flow, providing a powerful method to communicate the model's findings to stakeholders. Once a model is executed and data are collected, the analyst has a number of alternatives for data analysis. The data created during a model execution can be reviewed within Micro Saint Sharp or can be exported to statistical and graphics packages for post processing.

As stated before, the basis for task network models of human performance is the mainstay of human engineering analysis, the task analysis. Much of the information discussed above is generally included in the task analysis. Task network modeling greatly increases

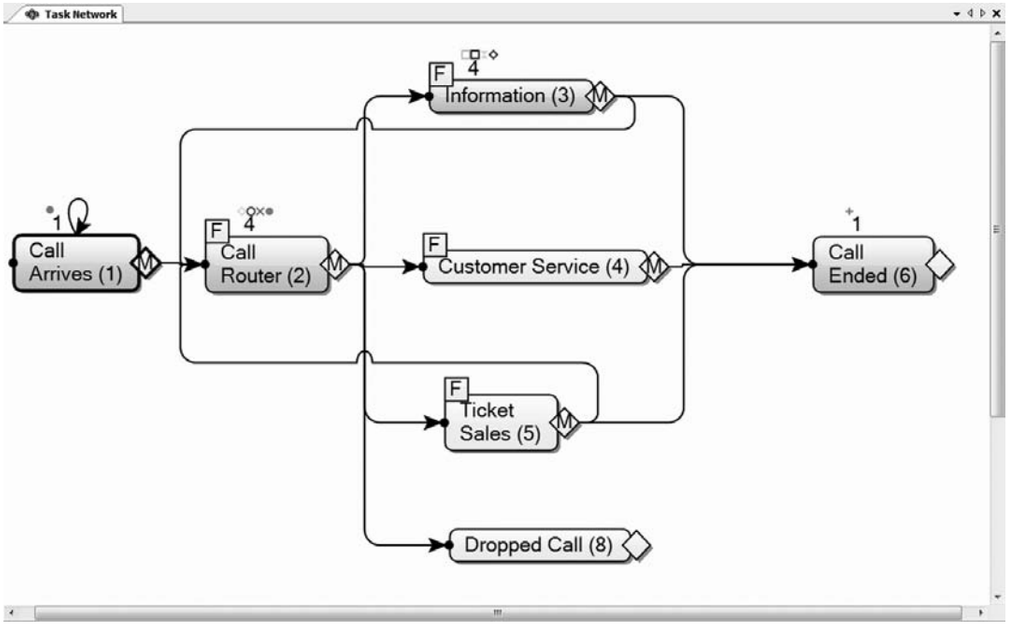

Figure 8 Task network animation during model execution in Micro Saint Sharp.

the power of task analysis since the ability to simulate a task network with a computer permits *prediction* of human performance rather than simply the *description* of human performance that a task analysis provides. What may not be as apparent, however, is the power of task network modeling as a means of modeling human performance in *systems*. Simply by describing the system's activities in this step-by-step manner, complex models of the system can be developed where the human's interaction with the system can be represented in a closed-loop manner. The preceding discussion, in addition to being an introduction to the concepts, is also intended to support the argument that task network modeling is a mature technology ready for application in a wide range of problem domains.

4.2 Task Network Model of a Process Control Operator

This simple hypothetical example illustrates how many of the basic concepts of task network modeling can be applied to studying human performance in a process control environment. It is intended to illustrate many of the concepts described above. The simple human task that we want to model is of an operator responding to an annunciator. The procedure requires that the operator compare readings on two meters. Based on the relative values of these readings, the operator must either open or close a valve until the values on the two meters are nearly the same. The task network in Figure 9 represents the operator activities for this model. Also, to allow the study of the effects of different plant dynamics (e.g., control lags), a simple one-node model of the line in which the valve is being opened is included in Figure 10.

The operator portion of the model will run the "monitor panels" task until the values of the variables "meter1" and "meter2" are different. The simulation could begin with these values being equal and then precipitate a change in values based on what is referred to as a *scenario event* (e.g., an event representing the effects of a line break on a plant state). This event could be as simple as

$$\text{meter } 1 = \text{meter } 1 + 2.0;$$

or as complex as an expression defining the change in the meter as a function of line break size, flow rates, and so on. An issue that consistently arises in model construction is how complex the plant system model should be. If the problem under study is purely operator performance, simple models will usually suffice. However, if overall plant behavior is of interest, the models of plant dynamics, such as meter values, are more important. Again, we recommend the "start simple" approach whenever possible.

When the transient occurs and the values of meter1 and meter2 start to diverge, the annunciator signal will trigger. This annunciator would be triggered in the plant

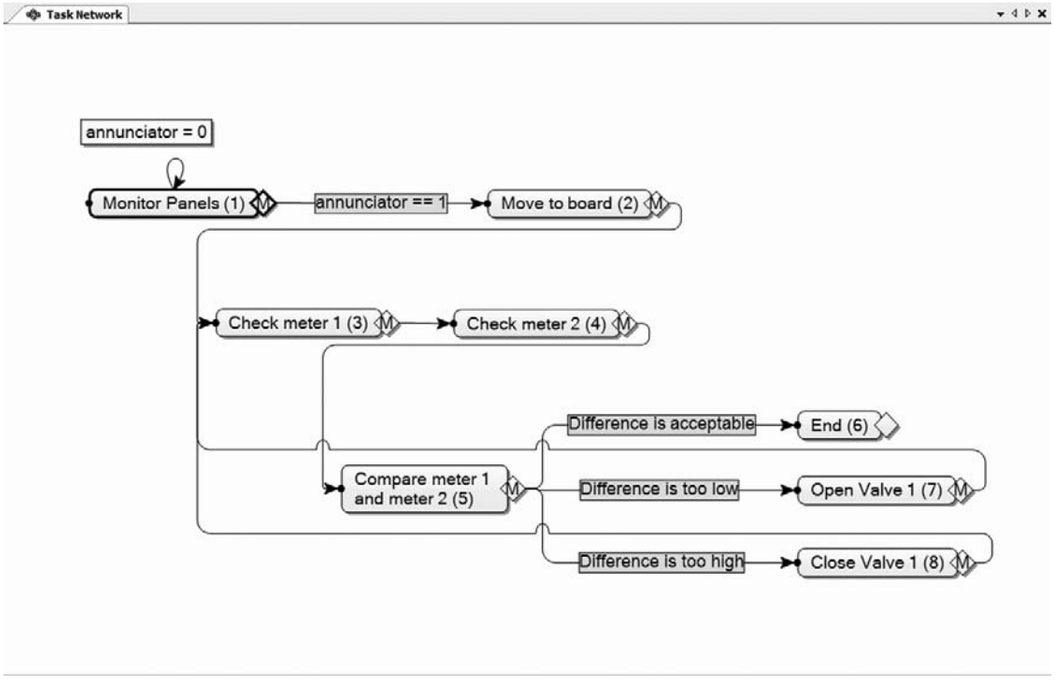

Figure 9 Task network model of a process control operator responding to an annunciator.

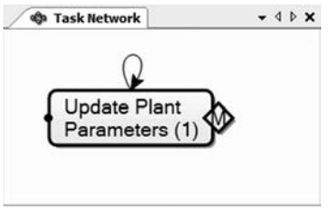

Figure 10 Simple one-node model of the plant integrated with the detailed operator model.

portion of the model by a task-ending effect such as

$$\text{if (meter 1 ! = meter 2) then annunciator} = 1;$$

Once the plant model sets the value of the variable annunciator to 1, the operator will begin to move to the appropriate board. Then the operator will continue through a loop to check the values for meter1 and meter2 and open valve 1, close valve 1, or make no change. The determination of whether to make a control input is determined by the difference in values between the two meters. If the value is less than the acceptable threshold, the operator would open the valve further. If the value is greater than the threshold, the operator would close the valve. This opening and closing of the valve would be represented by changes in the value of the variable valve1 as a task-ending effect of the tasks open valve1 and close valve1. In this simple model, operators do not consider rates of change in values for meter1 and therefore would get into an operator-induced oscillation if there were any response lag. A more sophisticated operator model could use rates of change in the value for meter1 in deciding whether to open or close valves.

Again, this is a very small model reflecting simple operator activity on one control via a review of two displays. However, it illustrates how large models of operator teams looking at numerous controls and manipulating many displays could be built via the same building blocks used in this model. The central concepts of a task network and shared variable reflecting human–system dynamics remain the same.

Given a task network model of a process control operator in a "current" control room, how might the model be modified to address human-centered design questions? Some examples are (1) modifying task times based on changes in the time required to access a new display; (2) modifying task times and accuracies based on changes in the content and format of displays; (3) changing task sequence, eliminating tasks, and/or adding tasks based on changes in plant procedures; (4) changing allocation of tasks and ensuing task sequence based on reallocation of tasks among operators; and (5) changing task times and accuracies based on stressors such as sleep loss or the effects of circadian rhythm. This is not intended as a definitive list of all the ways that these models may be used to study design or operations concepts but should illustrate how these models can be used to address design and operational issues.

4.3 Use of Task Network Modeling to Address Specific Design Concerns

In this section we examine two case studies in the use of task network simulation for studying human performance issues. The first case study explores how task network modeling can be used to assess task allocation issues in a cognitively demanding environment. The second example explores how task network modeling has been used to extend laboratory and field research on human performance under stress to new task environments. We should state clearly that these examples are intended to be representative of the types of issues that task network modeling can address as well as approaches to modeling human performance with respect to these issues. *They are not intended to be comprehensive with respect to either the issues that might be addressed or the possible techniques that the human factors practitioner might apply.* Simulation modeling is a technology whose application leaves much room for creativity on the part of the human factors practitioner with respect to application areas and methods. These two case studies are representative.

4.3.1 Crew Workload Evaluation

Perhaps the greatest contributor to human error in many systems is the extensive workload placed on the human operator. The inability of the operator to cope effectively with all of his or her information and responsibilities contributes to many accidents and inefficiencies. In recognition of this problem, new automation technologies have been introduced to reduce workload during periods of high stress. Some of these technologies are in the form of enhanced controls and displays, some are in the form of tools that "push" information to the operator and alert the operator in order to focus attention, and still others consist of adaptive tools that "take over" tasks when they sense that the operator is overloaded. Unfortunately, these technical solutions often introduce new tasks to be performed that affect the visual, auditory, and/or psychomotor workload of the operators.

Recently, new concepts in crew coordination have focused on better management of human workload. This area shows tremendous promise and is benefiting from efforts of human factors researchers. However, their efforts are hindered because there are limited opportunities to examine empirically the performance of different combinations of equipment and crew composition in a realistic scenario or context. Additionally, high workload is not typically caused by a single task but by situations in which multiple tasks must be performed or managed simultaneously. It is not simply that the quantity of tasks can lead to overload, but it also depends on the composition of those tasks. For example, two cognitive tasks being performed in parallel are much more effortful than a simple motor task and an oral communication task being performed together. The occurrence of these situations will not typically be discovered through normal human engineering task analysis or subjective workload analysis until there is a system to be tested. That is often too late to influence design. To rectify this problem, there has been a significant amount of

recent research and development aimed at human workload *prediction* models. Predictive models allow the designers of a system to estimate operator workload *without human subject experimentation*. From this and other research, a solid theoretical basis for human workload prediction has evolved, as is described in Wickens (1984).

In this section we discuss a study using task network modeling to predict the impact of task allocation on human workload. Although these examples are posed in the context of the design of a military system, the same techniques have been used in nonmilitary applications such as process control and user–computer interface design.

4.3.2 Modeling the Workload of a Future Command and Control Process

The Army command and control (C2) community is concerned with how new information technology and organizational changes projected for tomorrow's battlefield will affect soldier tasks and workload. To address this concern, an effort was undertaken to model soldier performance under current and future operational conditions. In this way, the impact of performance differences could be quantitatively assessed so that equipment and doctrine design could be influenced in a timely and effective manner.

In one C2 project, the primary concern was to determine how tasks should be allocated and automated such that a C2 team could evaluate all the relevant data and make decisions within an environment with particularly high time pressure. Specifically, the effort was to address the following key questions:

- How many crew members do you need?
- How do you divide tasks among jobs?
- How does decision authority flow?
- Can the crew meet decision timeline requirements?
- Is needed information usable and accessible?

Task network modeling was used to study crew member, task, and scenario combinations in order to examine these questions. Figure 11 shows the top-level diagram of the task network. Essentially, the crew members receive and monitor information about the system and the environment until an event occurs that pushes them out of the 10,000 and 20,000 networks into either a series of planning tasks or a series of evaluation, decision, direction, and execution tasks. The purpose of the planning task is to update tactical battle plans based on new information received from the system or the environment. Receipt of new intelligence data about the enemy's intention or capability is an example of an event that would cause crew members to undertake planning tasks. Similarly, receipt of information from the system about resource limitations might trigger the crew members to proceed down the alternative path (through evaluate to execute). Specifically, limited resources might cause crew members to evaluate whether the engagement is proceeding appropriately (30,000), decide how to adjust system parameters (40,000), direct the appropriate response to the correct level of command (50,000), and then execute the order (60,000). Upon completion, crew members would return to monitoring the system and situation.

Each rectangle in the task network shown in Figure 11 actually consists of a network of tasks. An example of the tasks that belong to network 10000 are shown in Figure 12. As described earlier, the tasks in network 10000 are linked by probabilistic and tactical decisions. Each of the tasks in the C2 task network

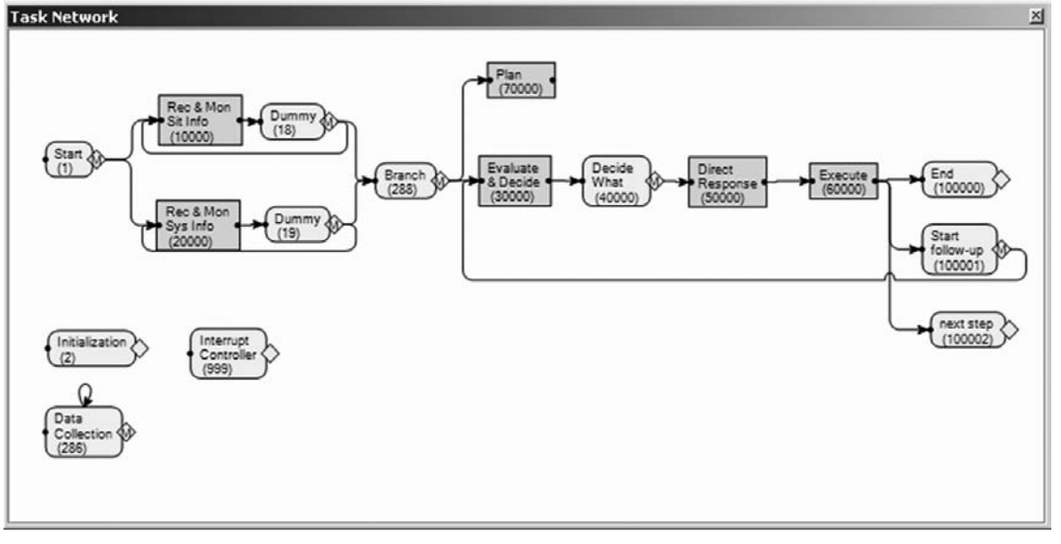

Figure 11 Upper level task network.

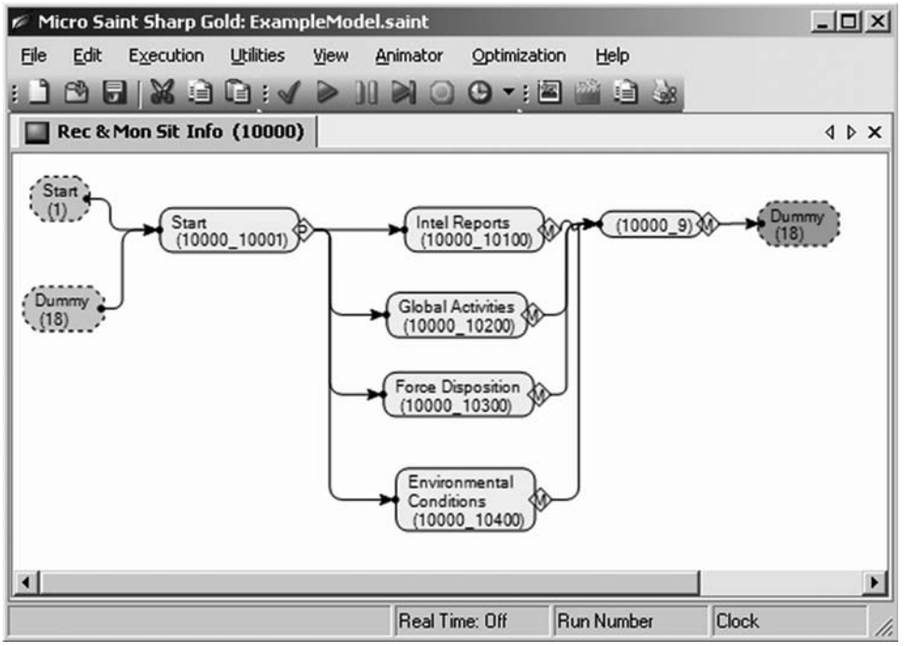

Figure 12 Second-level task network.

is associated with several items of human performance data:

- *Task Performance Time.* These data consist of a mean, standard deviation, and distribution. The data were collected from a combination of three sources: (1) human factors literature (e.g., Fitt's law), (2) empirical studies during operator-in-the-loop simulator exercises, and (3) subject matter experts.
- *Branching Logic*. Although the task network indicates a general process flow, this particular model was designed to respond to scenario events. Because of that design decision, each task includes logic to determine the following task. For example, if the scenario is very intense and multiple target tracks are available, crew members would follow a different task flow than if they were performing routine system checks.
- *Release Rules*. Logic controlling the number and types of parallel tasks each crew member can perform is contained in each task's release condition.

Since one purpose of the model was to examine various task allocation strategies, the model was designed to incorporate several measures of crew member workload. The basis of this technique is an assumption that excessive human workload is not usually caused by one particular task required of the operator. Rather, the human having to perform several tasks simultaneously leads to overload. Since the factors that cause this type of workload are intricately linked to these dynamic aspects of the human's task requirements, task network modeling provides a good basis for studying how task allocation and sequencing can affect operator workload.

However, task network modeling is not inherently a model of human workload. The only relevant output common to all task network models is the time required to perform a set of tasks and the sequence in which the tasks are performed. Time information alone would suffice for some workload evaluation techniques, such as Siegel and Wolf (1969), whereby workload is estimated by comparing the time available to perform a group of tasks to the time required to perform the tasks. Time available is driven by system performance needs, and time required can be computed with a task network model. However, it has long been recognized that this simplistic analysis misses many aspects of the human's tasks that influence both perceived workload and ensuing performance. At the very least, this approach misses the fact that some pairs of tasks can be performed in combinations better than other pairs of tasks.

One of the most promising theories of operator workload, which is consistent with task network modeling, is the multiple-resource theory proposed by Wickens (see Wickens et al., 1983). Simply stated, the multiple-resource theory suggests that humans have several different resources that can be tapped simultaneously and with varying levels of inter-resource conflict and competition. Depending on the nature of the information

processing tasks required of a human, these resources would have to process information sequentially (if different tasks require the same types of resources) or possibly in parallel (if different tasks required different types of resources). There are many versions of this multiple-resource theory in the workload literature (e.g., McCracken and Aldrich, 1984; Archer and Adkins, 1999). In this chapter we provide a discussion of the underlying methodology of the basic theory.

Multiple-resource workload theory is implemented in a task model in a fairly straightforward manner. First, each task in the task network is characterized by the workload demand required in each human resource, often referred to as a *workload channel*. Examples of commonly used channels include auditory, visual, cognitive, and psychomotor. Particular implementations of the theory vary in the channels that are included and the fidelity with which each channel is measured (high, medium, low vs. seven-point scale). As an example, the scale for visual demand is presented in Figure 13.

Similar scales have been developed for the auditory, cognitive, fine-motor, gross-motor, speech, and tactile channels. Using this approach, each operator task can be characterized as requiring some amount of each of the seven types of resources, as represented by a value (typically between 1 and 7). All operator tasks can be analyzed with respect to these demand values. In performing a set of tasks pursuant to a common goal (e.g., engage an enemy target), crew members frequently must perform several tasks simultaneously, or at least nearly so. For example, they may be required to monitor a communication network while visually searching a display for target track. Given this, the workload literature indicates that the crew member may either accept the increased workload (with some risk of performance degrading) or begin dumping tasks perceived as less important. To factor these two issues into task network simulations, two approaches can be incorporated: (1) evaluate combined operator workload demands for tasks that are being performed concurrently and/or (2) determine when the operator would begin dumping tasks due to overload.

During a task network simulation, the model of the crew may indicate that they are required to perform several tasks simultaneously. The task network model evaluates total attentional demands for each human resource (e.g., visual, auditory, fine motor, gross motor, speech, tactile, and cognitive) by combining the attentional demands across all tasks that are being performed simultaneously. Intra- and interresource conflict values are then computed that indicate how much the different resources compete with each other. The conflict score is then used to increase the total attentional demand. This combination leads to an overall workload demand score for each crew member.

To implement this approach in Micro Saint Sharp, the task-beginning effect can be used to increment variables that represent the current workload score in each resource. Then, while the tasks are being performed, these variables track attentional demands. When the tasks are completed, the task-ending effects can decrement the values of these variables accordingly. Therefore, if these workload variables were recorded and then plotted as the model runs, the output would look something like as shown in Figure 14. This result can be used to identify points of high workload throughout the scenario being modeled. The human factors practitioner can then review the tasks that led to the points of

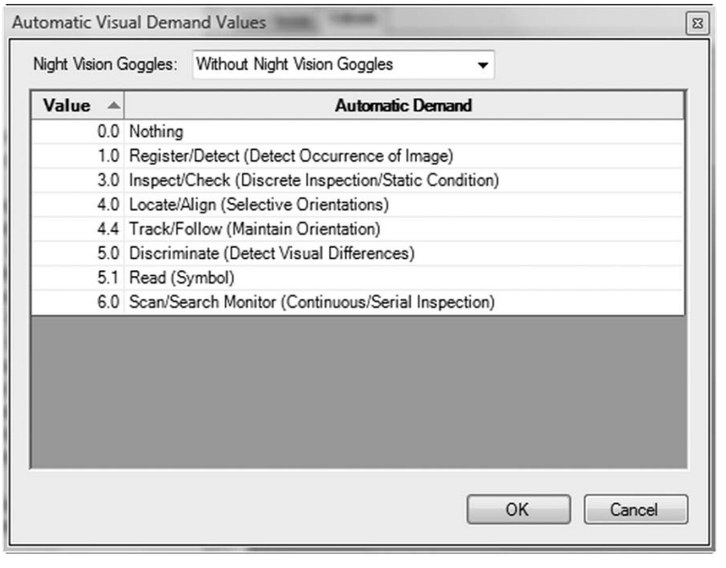

Figure 13 Visual workload scale.

high workload and determine whether they should be reallocated or redesigned in order to alleviate the peak. This is a common approach to modeling workload.

Once the task networks were verified with knowledgeable crew members, they became part of the human factors team's analytical test bed. Figure 15 shows the overall method that can be used to examine aspects of crew member performance across a wide variety of operational scenarios and crew configuration concepts. The center of this diagram, labeled the task network, represents the tasks that the crew performs. The network itself, representing the flow of the tasks, does not change between model runs. Rather, the model has been parameterized so that an event scenario stimulates the network. The left side of the diagram illustrates the types of data that are used to drive the task network model. In this case, those data include crew configurations, or allocations of tasks to different crew members and automation devices, as well as scenario events. The scenario events represent an externally generated time-ordered list of the events that trigger the crew members to perform tasks in the task network. The right side of Figure 15 represents the types of outputs that can be produced from this task network model. One of the primary outputs is a crew member workload graph, such as that shown in Figure 14. Another is operator utilization, as shown in Figure 16.

4.3.3 Extensions to Other Environments

The workload analysis methodology described above has been developed into a stand-alone task network modeling tool by the Army Research Laboratory (ARL) Human Research and Engineering Directorate (HRED) as part of the IMPRINT (Archer and Adkins, 1999). IMPRINT integrates task network modeling software with features that specifically support the multiple-resource theory of workload discussed above. It provides the human factors practitioner with an environment that supports the analysis of task assignment to crew members based on four factors:

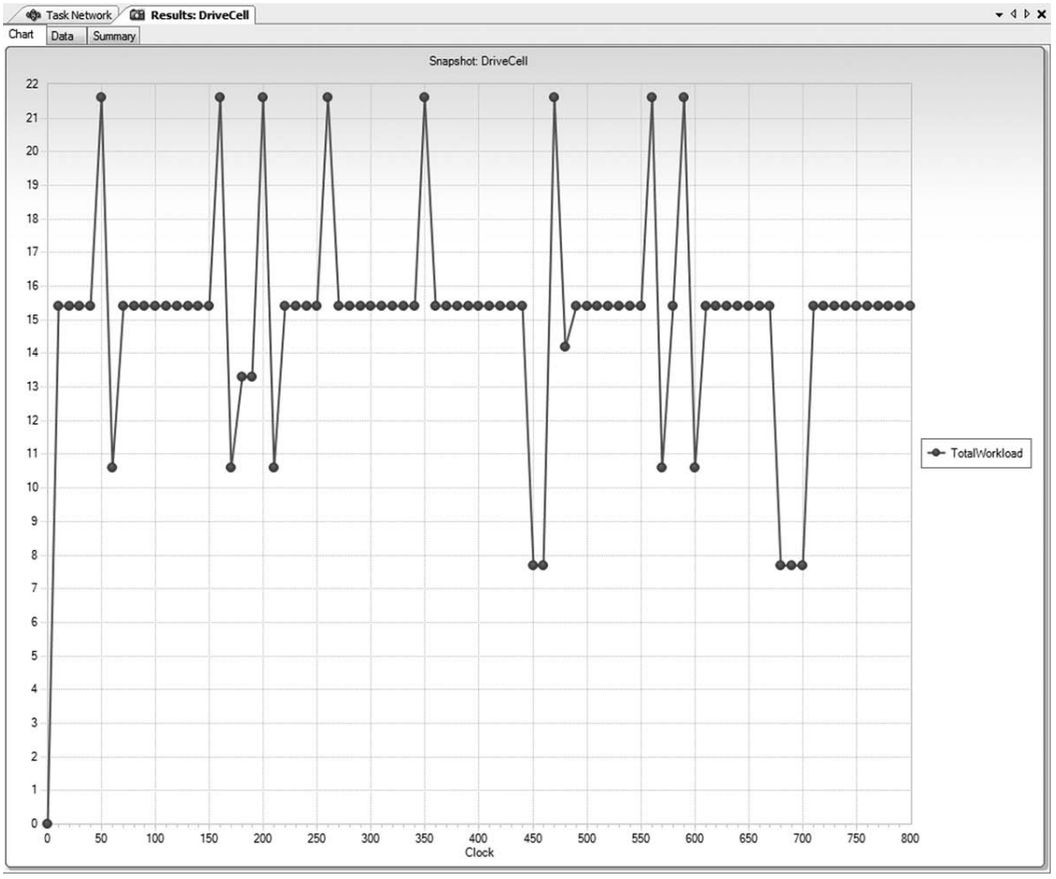

Figure 14 Workload output from a task network model.

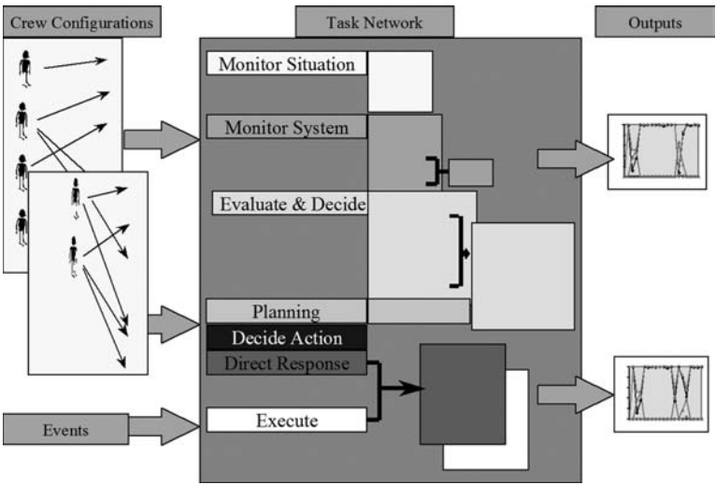

Figure 15 Overall method for examining workload in a complex system.

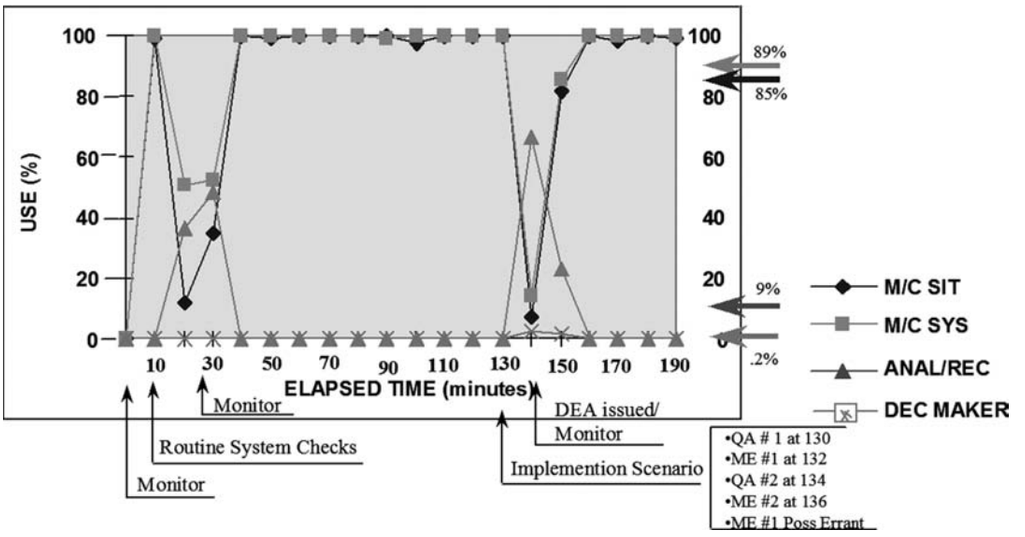

Figure 16 Percent utilization taken over 10-min intervals.

1. *Workload of Crew Members.* Tasks should be assigned to minimize the amount of time that crew members will spend in situations of excessive workload.
2. *Time Performance Requirements.* Tasks must be assigned and sequenced so that they are completed within the available time. This consideration is essential since time constraints often will drive the need to perform several tasks simultaneously.
3. *Likelihood of Successful Performance and Consequences of Failure.* Tasks must be assigned and sequenced so that they can be completed within a specified accuracy measure.
4. *Access to Controls and Displays.* Tasks cannot be assigned to crew members that do not have access to the necessary controls and displays.

Of course, there are numerous theoretical questions regarding this simplistic approach to assessing workload in an operational environment. However, even the use of this simple approach has been shown to provide useful

insight during design. For example, in a study conducted by the Army (Allender, 1995), a three-man crew design was evaluated using a task network model. The three-man model was constructed using data from a prototype four-man system. From this model-based analysis, the three-man design was found to be unworkable. Later, experimentation using human subjects verified that the model's workload predictions were sufficiently accurate to point the design team in a valid direction.

IMPRINT also includes built-in constructs for simulating *workload management* strategies that operators would employ to accommodate points of high operator workload (Plott, 1995). The ultimate result of simulating the workload management strategies is that the operator task network being modeled is *dynamic*. In other words, the task sequence, operator assignments, and individual task performance may change in response to excessive operator workload as the task network model executes. These changes may be as simple as one operator handing tasks off to another operator to reduce workload to an acceptable level or as complex as the operator beginning to time share tasks in order to complete all the tasks assigned, potentially with associated task performance penalties. Ultimately, the tool provides an estimate of system-level performance as a result of these realistic workload management strategies. This innovation in modeling provides greater fidelity in efforts that model human behavior in the context of system performance, particularly in high-workload environments such as complex system control and management.

4.3.4 Extending Research Findings to New Task Environments

Task network modeling was used by LaVine et al. (1995) to extend laboratory data and field data collected on one set of human tasks to predicting performance on similar tasks. The problem of extending laboratory or field human performance data to other tasks has plagued the human engineering community for years. We know intuitively that human performance data can be used to predict performance for similar tasks. However, it is often the case that the task whose performance we want to predict is similar in some ways but different in others. The approach described below uses a skill taxonomy to quantify task similarity and therefore provides a means for determining how other tasks will be affected when exposed to a common stressor on human performance. Once functional relationships are defined between a skill type and a stressor, task network modeling is used to determine the effect of the stressor on performance of a complex task that uses many of these skills simultaneously.

The specific approach below is being used by the U.S. Army to predict crew performance degradation as a function of a variety of stressors. It is not intended to represent a universally acceptable taxonomy for simulating human response to stress. The selection of the best taxonomy would depend on the particular tasks and stressors being studied. What this example is intended to illustrate is another way that task network modeling can be used to predict human performance by *making a series of reasonable assumptions* that can be played together in a model for the purpose of making predictions that would be impossible to make otherwise. The methodology for predicting human performance degradation as a function of stressors consists of three parts: (1) a *taxonomy* for classifying tasks according to basic human skills, (2) *degradation functions* for each skill type for each stressor, and (3) *task network models* for the human-based system whose performance is being predicted. Conceptually, either laboratory or field data can be used to develop links between a human performance stressor (e.g., heat, fatigue) and basic human skills. By selecting a skills taxonomy that is sufficiently discriminating to make this assumption reasonable, one can assume that the effects of the stressor on all tasks involving the skill will be approximately the same. The links between the level of a stressor (e.g., fatigue) and resulting skill performance (e.g., the expected task time increase from fatigue) are defined mathematically as the degradation function. The task network model is the means for linking these back to complex human–system performance.

Taxonomy The basic premise behind the taxonomy is that the tasks that humans perform can be broken down into basic human skills or atomic tasks (Roth, 1992). The taxonomy that was used by Roth consists of five skill types described by Roth as follows:

1. *Attention:* the ability to attend actively to a stimulus complex for extended periods of time in order to detect specified changes or classes of changes that indicate the occurrence of some phenomenon that is critical to task performance
2. *Perception:* the ability to detect and categorize specific stimulus patterns embedded in a stimulus complex
3. *Psychomotor skill:* the ability to maintain one or more characteristics of a situation within a set of defined conditions over a period of time, either by direct manipulation or by manipulating controls that cause changes in the characteristics
4. *Physical skill:* the ability to accomplish sustained, effortful muscular work
5. *Cognitive skill:* the ability to apply concepts and rules to information from the environment and from memory in order to select or generate a course of action or a plan (includes communicating the course of action or plan to others)

These five skills covered most of the tasks that were of interest to the Army for this study and still provided a manageable number of categories for an analyst to use.

Degradation Functions The degradation functions quantitatively link skill performance to the level of a stressor. The degradation functions can be developed from any data source, including standard test batteries or actual human tasks. Through statistical analysis, one can build skill degradation functions for each taxon. These functions map the performance decrement expected on

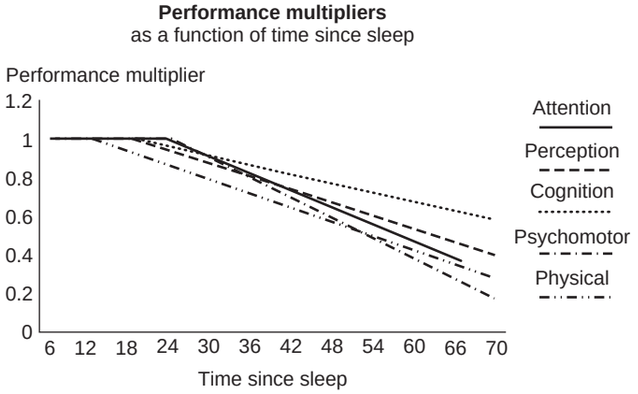

Figure 17 Performance degradation functions associated with each of the human skills from the taxonomy.

a skill based on the parameters of the performance-shaping factor (e.g., time since sleep). An example of these functions is presented in Figure 17.

Incorporating the Degradation Functions into Task Network Models to Predict Overall Human–System Performance Degradation The key to making this approach useful to predicting complex human performance is the task network model of the new task. In the task network model of the human's activities, all tasks are defined with respect to the percentage of each skill required from the taxonomy. For example, the following are ratings for tasks faced by a console operator responding to telephone contacts:

Detect ring	50% attention, 50% perception
Select menu item using a mouse	40% attention, 60% psychomotor
Interpret customer's request for information	100% cognitive

In building the task network model, mathematical expressions can be developed that degrade a specific task's performance through an arithmetic weighting of skill degradation multipliers that are derived from the degradation functions. For example, if the fatigue parameter was "time since sleep" and the value of that parameter was "36 hours since sleep," the task time performance multipliers would be as follows in the example above:

Attention performance multiplier	0.82
Perception performance multiplier	0.808
Cognition performance multiplier	0.856
Psychomotor performance multiplier	0.784
Physical performance multiplier	0.727

Based on these multipliers and the task weightings above, the specific task effects would be:

- Detect ring (50% attention, 50% perception)

$$\text{Task multiplier} = 0.5 \times 0.82 + 0.5 \times 0.808 = 0.814$$

- Select menu item using a mouse (40% attention, 60% psychomotor)

$$\text{Task multiplier} = 0.4 \times 0.82 + 0.6 \times 0.784 = 0.7984$$

- Interpret customer's request for information (100% cognitive)

$$\text{Task multiplier} = 0.856$$

In a model of the complex tasks being examined by LaVine et al. (1995), the task networks consisted of several dozen or even several hundred tasks. Through the approach described above, each task in a model exhibited a unique response to a stressor depending on the particular skills that it required. The task network model then provided the means for relating the individual task performance to overall human–system performance as a function of stressor level (e.g., the time to perform a complex series of tasks involving decision making and error correction). Through this type of analysis, LaVine et al. were able to develop curves such as that shown in Figure 18 relating human performance to a stressor. These relationships would have been virtually impossible to develop experimentally.

Again, there were a number of simplifying assumptions that were made in this research. However, by being willing to accept these assumptions, LaVine et al. were able to characterize how complex human–system performance would be affected by a variety of stressors over a wide range in a relatively short time. As such, they were able to estimate the effects of stressors that would have otherwise been pure guesswork.

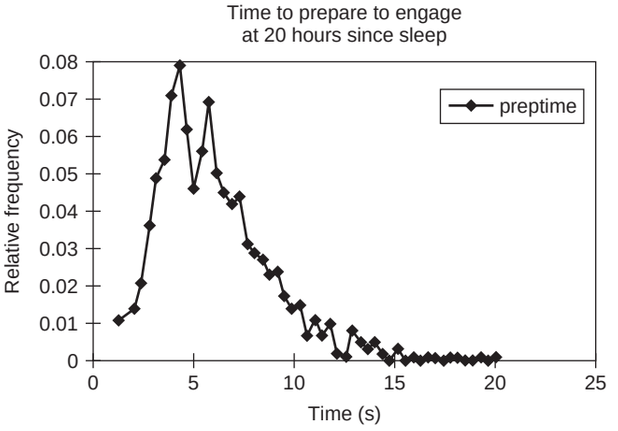

Figure 18 Frequency distribution of expected human performance as a function of time since sleep that was derived using task network modeling.

4.3.5 Incorporation of Advanced Attention Model

In recent years advanced theories of selective attention have been advanced to the point that they can be embedded into human performance models to more accurately predict how a human might perform in a visually complex environment. As the fidelity of human performance models improve, so too does the need for integrating complex human attention models in order to predict how operators will allocate their visual attention. Attention models that have shown preliminary success when being linked to human performance models are the SEEV (salience, effort, expectancy, and value) model (Horrey et al., 2006) and the newer N-SEEV (noticing-SEEV) model (Wickens and McCarley, 2008). SEEV is a computational, plausible model that accounts for how four quantifiable elements do and/or should drive operator's attention around a complex working environment. SEEV models the operator's visual scanning pattern attending to a given "area of interest" (AOI) that supports the tasks that need to be performed. The SEEV and N-SEEV algorithms are described in detail in Chapter 5.

The Man–Machine Integration Design and Analysis System (MIDAS) has recently been updated to incorporate the SEEV algorithm (Gore et al., 2009). The integration of the SEEV model into MIDAS allows dynamic scanning behaviors by calculating the probability that the operator's eye will move to a particular AOI given the tasks the operator is engaged in within the multitask context. It also better addresses allocation of attention in dynamic environments such as flight and driving tasks. In MIDAS, effort, expectancy, and value are assigned values between 0 and 1, while salience is left unconstrained. Effort, expectancy, and value drive the human operator's visual attention around the displays. However, if a salient event occurs, then $P(\text{AOI})$ may be offset by the display exhibiting the salient event until the display location of the salient event has been fixated and detected. The improved predictive capability of information-seeking behavior that resulted from the implementation of the validated SEEV model leaves MIDAS better suited to predict performance in complex human–machine systems.

4.3.6 Summary

Once again, the above are intended to serve as examples, not a catalog of problems or approaches that are appropriate for task network modeling. Task network modeling is an approach to extend task and systems analysis to make predictions of human–system performance. The creative human factors and ergonomics practitioner will find many other useful applications and approaches.

5 FIRST-PRINCIPLE APPROACH: ADAPTIVE CONTROL OF THOUGHT–RATIONAL COGNITIVE ARCHITECTURE

The other fundamental approach to modeling human performance is based on the mechanisms that underlie and cause human behavior. Since this approach is based on fundamental principles of the human and his or her interaction with the system and environment, we have designated them as *first-principle models*. By integrating these models with models of the system and environment, the human factors specialist can predict the full behavior of large-scale interactive human–machine systems. The ACT-R cognitive architecture (Anderson and Lebiere, 1998) is a production system theory that models the steps of cognition by a sequence of production rules that fire to coordinate retrieval of information from the environment and from memory. It is a cognitive architecture that can be used to model a wide range of human cognition. It has been used to model tasks from memory retrieval (Anderson et al., 1998) to visual search (Anderson et al., 1997). The range of models developed, from those purely concerned with internal

cognition to those focused on perception and action, makes ACT-R a plausible candidate to model complex tasks involving the interaction of one (or more) human operator with complex systems with the goal of evaluating the design of those systems. In all domains, ACT-R is distinguished by the detail and fidelity with which it models human cognition. It makes claims about what occurs cognitively every few hundred milliseconds in performance of a task. ACT-R is situated at a level of aggregation above those of basic brain processes (targeted by other modeling approaches, such as neural networks) but considerably below such complex tasks as air traffic control. The new version of the theory has been designed to be more relevant to tasks that require deploying significant bodies of knowledge under conditions of time pressure and high information-processing demand. This is because of the increased concern with the temporal structure of cognition and with the coordination of perception, cognition, and action.

5.1 ACT-R

ACT-R is a unified architecture of cognition developed over the last 30 years at Carnegie Mellon University. At a fine-grained scale it has accounted for hundreds of phenomena from the cognitive psychology and human factors literature. The most recent version, ACT-R 6.0 (Anderson et al., 2007), is a modular architecture composed of interacting modules for declarative memory, perceptual systems such as vision and audition modules, and motor systems such as manual and speech modules, all synchronized through a central production system (see Figure 19). This modular view of cognition is a reflection both of functional constraints and of recent advances in neuroscience concerning the localization of brain functions. ACT-R is also a hybrid system that combines a tractable symbolic level that enables the easy specification of complex cognitive functions with a subsymbolic level that tunes itself to the statistical structure of the environment to provide the graded characteristics of cognition such as adaptivity, robustness, and stochasticity.

The central part of the architecture is the production module. A production can match the contents of any combination of buffers, including the goal buffer, which holds the current context and intentions; the retrieval buffer, which holds the most recent chunk retrieved from declarative memory; the visual and auditory buffers, which hold the current sensory information; and the manual and vocal buffers, which hold the current state of the motor and speech module. The highest rated matching production is selected to effect a change in one or more buffers, which in turn triggers an action in the corresponding module(s). This can be an external action (e.g., movement) or an internal action (e.g., requesting information from memory). Retrieval from memory is initiated by a production specifying a pattern for matching in declarative memory. Each chunk competes for retrieval, with the most active chunk being selected and returned in the retrieval buffer. The activation of a chunk is a function of its past frequency and recency of use, the degree to which it matches the pattern requested, plus stochastic noise. Those factors confer memory retrievals, and behavior in general, desirable "soft" properties such as adaptivity to changing circumstances, generalization to similar situations, and variability (Anderson and Lebiere, 1998).

The current goal is a central concept in ACT-R, which as a result provides strong support for goal-directed behavior. However, the most recent version of the architecture is less goal focused than its predecessors

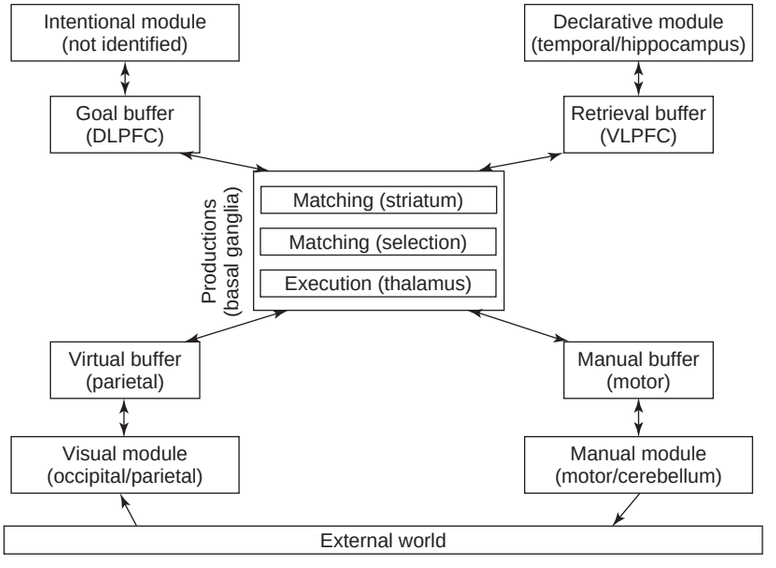

Figure 19 Modular view of ACT-R cognitive architecture.

by allowing productions to match to any source of information, including the current goal, information retrieved from declarative memory, objects in the focus of attention of the perceptual modules, and the state-of-the-action modules. The content of many of those buffers, especially the perceptual buffers, might have changed not as a function of an internal request but as a result of an external event happening, perhaps unexpectedly, in the outside world. This emphasis on asynchronous pattern matching of a wide variety of information sources better enables ACT-R to operate and react efficiently in a dynamic fast-changing world through flexible goal-directed behavior which gives equal weight to internal and external sources of information.

There are three main distinctions in the ACT-R architecture. First, there is the procedural–declarative distinction that specifies two types of knowledge structures: chunks for representing declarative knowledge and productions for representing procedural knowledge. Second, there is the symbolic level, which contains the declarative and procedural knowledge, and the subsymbolic level of neural activation processes that determine the speed and success of access to chunks and productions. Finally, there is a distinction between the performance processes by which the symbolic and subsymbolic layers map onto behavior and the learning processes by which these layers change with experience.

Human cognition can be characterized as having two principal components: (1) the knowledge and procedures codified through specific training within the domain and (2) the natural cognitive abilities that manifest themselves in tasks as diverse as memory, reasoning, planning, and learning. The fundamental advantage of an integrated architecture like ACT-R is that it provides a framework for modeling basic human cognition and integrating it with specific symbolic domain knowledge of the type specified by domain experts (e.g., rules specifying what to do in a given condition, a type of knowledge particularly well suited for representation as production rules). However, performance described by symbolic knowledge is mediated by parameters at the subsymbolic level that determine the availability and applicability of symbolic knowledge. Those parameters underlie ACT-R's theory of memory, providing effects such as decay, priming, and strengthening and make cognition adaptive, stochastic, and approximate, capable of generalization to new situations and robustness in the face of uncertainty. They also can account for the limitations of human performance, such as latencies to perform tasks and errors that can originate from a number of sources. Finally, they provide a basis for representing individual differences such as those in working memory capacity, attentional focus, motivation, and psychomotor speed as well as the impact of external behavior moderators such as fatigue (Lovett et al., 1999; Taatgen, 2001; Gunzelmann et al., 2009) through continuous variations of those subsymbolic architectural parameters that affect performance in complex tasks.

Because they influence quantitative predictions of performance so fundamentally, we describe in some more detail the subsymbolic level in which continuously varying quantities are processed, often in parallel, to produce much of the qualitative structure of human cognition. These subsymbolic quantities participate in neural-like activation processes that determine the speed and success of access to chunks in declarative memory as well as the conflict resolution among production rules. ACT-R also has a set of learning processes that can modify these subsymbolic quantities. Formally, activation reflects the log posterior odds that a chunk is relevant in a particular situation. The *activation* A_i of a declarative chunk i is computed as the sum of its base-level activation B_i plus its context activation:

$$A_i = B_i + \sum_j W_j S_{ji}$$

In determining the context activation, W_j designates the attentional weight given the focus element j. An element j is in the focus, or in context, if it is part of the current goal chunk (i.e., the value of one of the goal chunk's slots); S_{ji} stands for the strength of association from element j to chunk i. ACT-R assumes that there is a limited capacity of source activation and that each goal element emits an equal amount of activation. Source activation capacity is typically assumed to be 1 (i.e., if there are n source elements in the current focus each receives a source activation of $1/n$). The associative strength S_{ji} between an activation source j and a chunk i is a measure of how often i was needed (i.e., retrieved in a production) when chunk j was in the context. Associative strengths provide an estimate of the log likelihood ratio measure of how much the presence of a cue j in a goal slot increases the probability that a particular chunk i is needed for retrieval to instantiate a production. The *base-level activation* of a chunk is learned by an architectural mechanism to reflect the past history of use of a chunk i:

$$B_i = \mathrm{In} \sum_{j=1}^{n} t_j^{-d} \approx \mathrm{In} \frac{nL^{-d}}{1-d}$$

where n stands for the number of references to chunk i, t_j stands for the time elapsed since the j reference to chunk i, d is the memory decay rate, and L denotes the lifetime of a chunk (i.e., the time since its creation). As Anderson and Schooler (1991) have shown, this equation produces the power law of forgetting (Rubin and Wenzel, 1990) as well as the power law of learning (Newell and Rosenbloom, 1981). When retrieving a chunk to instantiate a production, ACT-R selects the chunk with the highest activation A_i. However, some stochasticity is introduced in the system by adding Gaussian noise of mean zero and standard deviation σ to the activation A_i of each chunk. In order to be retrieved, the activation of a chunk needs to reach a fixed retrieval threshold τ that limits the accessibility of declarative elements. If the Gaussian noise is approximated with a sigmoid distribution, the *probability P* of chunk *i to be retrieved* by a production is

$$\mathrm{P} = \frac{1}{1 + e^{-(\mathrm{A_i} - \tau)/\mathrm{s}}}$$

where $s = \sqrt{3}\sigma/\pi$. The activation of a chunk i is related directly to the latency of its retrieval by a production p. Formally, *retrieval time* T_{ip} is an exponentially decreasing function of the chunk's activation A_i:

$$T_{ip} = Fe^{-fA_i}$$

where F is a time scaling factor. In addition to the latencies for chunk retrieval as given by the retrieval time equation, the total time of selecting and applying a production is determined by executing the actions of a production's action part, whereby a value of 50 ms is typically assumed for elementary internal actions. External actions, such as pressing a key, usually have a longer latency determined by the ACT-R/PM perceptual–motor module (Byrne and Anderson, 2001). In summary, subsymbolic activation processes in ACT-R make a chunk active to the degree that past experience and the present context (as given by the current goal) indicate that it is useful at this particular moment.

Just as subsymbolic activation processes control which chunk is retrieved from declarative memory, the process of selecting which production to fire at each cycle, known as conflict resolution, is also determined by subsymbolic quantities called utility that are associated with each production. The utility of a production is defined as

$$U_i(n) = U_i(n-1) + \alpha[R_i(n) - U_i(n-1)]$$

where $U_i(n)$ is the utility of a production i after its nth application, $U_i(n-1)$ is its utility after its $(n-1)$st application, $R_i(n)$ is the reward the production receives, and α is the learning rate. Just as for retrieval, *conflict resolution* is a stochastic process through the injection of noise in each production's utility, leading to a probability of selecting a production i given by

$$Probability(i) = \frac{e^{U_i/\sqrt{2s}}}{\sum_j e^{U_j/\sqrt{2s}}}$$

where the summation is over all the productions which are currently able to fire. The production with the highest utility (after noise is added) will be the one chosen to fire. Similar computations are at work in other modules, such as the perceptual–motor modules. Especially important are the parameters controlling the time course of processing as one attempts to execute a complex action or as one shifts visual attention to encode a new stimulus (Byrne and Anderson, 2001). Recent work has simplified the process of specifying the sequence of steps for complex actions by allowing ACT-R modelers to demonstrate actions on interfaces (John et al., 2004; Matessa and Mui, 2009). ACT-R can not only predict direct quantitative measures of performance such as latency and probability of errors but, from the same mechanistic basis, can also arise more global, indirect measures of performance, such as cognitive workload. Although ACT-R has traditionally shied away from such meta-awareness measures and concentrated on matching directly measurable data such as external actions, response times, and eye movements, it is by no means incapable of doing so. For the purpose of the task described below, Lebiere (2001) proposed a measure of cognitive workload in ACT-R grounded in the central concept of unit task (Card et al., 1983). *Workload* is defined as the ratio of time spent in critical unit tasks to the total time spent on the task. *Critical unit tasks* are defined as tasks that involve actions, such as a goal to respond to a request for action with a number of mouse clicks, or tasks that involve some type of pressure, such as a goal to scan a display result from the detection of an event onset. The ratio is scaled to fit the particular measurement scale used in the self-assessment report. Lebiere (2001) describes possible elaborations of this basic measure.

5.2 AMBR

In this section we describe in some detail the constraints and requirements of the process of developing an ACT-R model for a task of moderate complexity and the range of quantitative predictions that one can expect from such a model. The task is a synthetic air traffic control simulation that was developed for the agent-based modeling of behavior representation (AMBR) comparison (Pew and Gluck, 2004) that arose from a report (Pew and Mavor, 1998) that highlighted the need for more robust, realistic human performance models (HPMs) for use in simulations for training and system acquisition

The AMBR project was designed to advance the state of the art in cognitive and behavioral modeling, especially models of integrative performance, requiring the coordination of memory, learning, multitasking, interruption handling, and perceptual and motor systems in order to scale more effectively to real-world environments. The program provided a structure to gather human performance data and evaluate the accuracy and predictiveness of the models. The AMBR program was organized as a series of comparisons among alternative modeling approaches including ACT-R but also the Air Force Research Laboratory's DCOG (Eggleston et al., 2001), CHI Systems, Inc.'s COGNET/iGEN (Zachary et al., 2001), and George Mason University's EASE (Chong, 2001).

The task designed to elicit the desired behaviors is a synthetic air traffic control simulation. This domain requires a controller to manage one sector of airspace, especially the transition of aircraft into and out of the sector. Scenarios can vary the number, speed, altitude, and type of aircraft requesting access to the sector and can be complicated by having them arrive from multiple directions and adjoining sectors. This is a rich enough infrastructure to create a variety of scenarios having variable task load levels and varying levels of planning complexity. Figure 20 displays a screen shot of the simulation. The main part of the screen on the left contains a graphical representation of the entire airspace, with the part controlled by the human or model agent contained in the central yellow square. The rest of the airspace is divided by the yellow lines in four

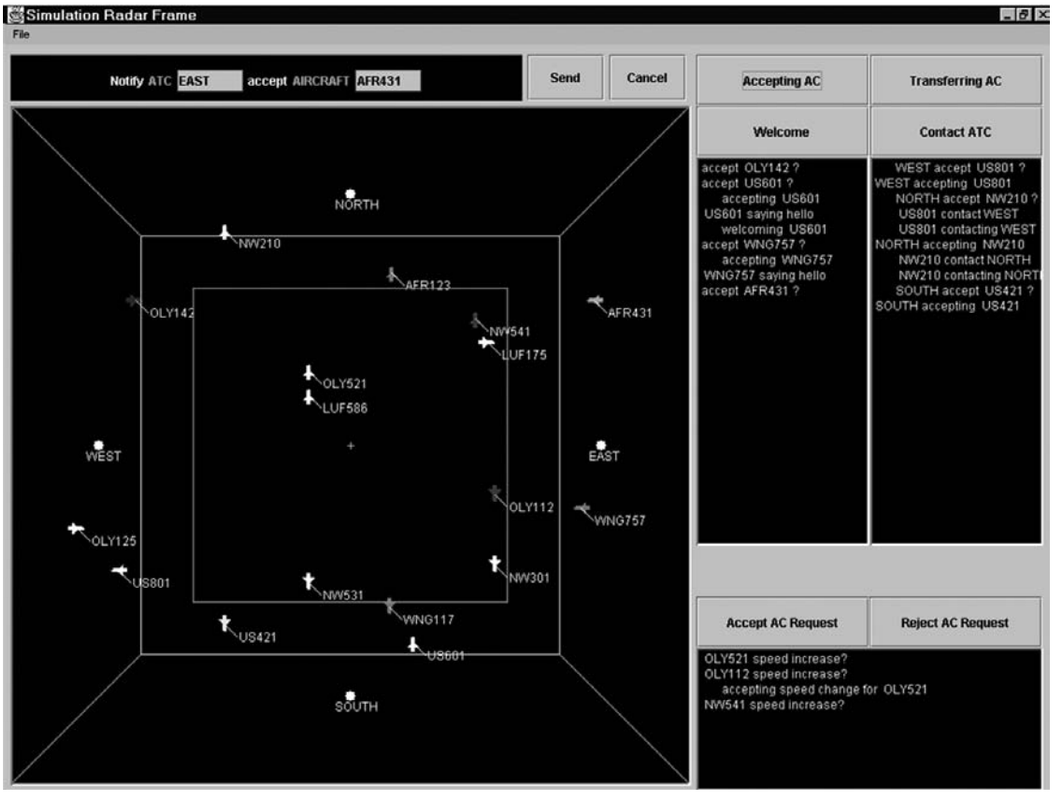

Figure 20 Screen shot of the AMBR simulation.

regions, north, east, south, and west, each managed by a separate controller. At any point during the simulation a number of airplanes (the exact number being a parameter controlling the difficulty of the task) are present in the airspace, flying through the central region or entering or exiting it. The task of the central controller is to exchange messages with the airplanes (each tagged with its identifying code, e.g., UAL344) and neighboring controllers to manage their traversal of its airspace. Those messages are displayed in the text windows on the right of the screen, with each window dedicated to a specific message category. The top left window concerns messages sent when a plane is entering the central controller's region, while the top right window concerns messages sent when a place is exiting the central region. Both windows include messages exchanged between controllers as well as messages between the central controller and the plane itself. The bottom window concerns messages from and to planes requesting a speed increase, which should be granted unless that plane is overtaking another plane, which is the only airspace conflict that this simplified task allows.

A single event involves a number of messages being exchanged, all of which are appended to the relevant text window. For example, in the case of a plane about to enter the central region, a message requesting permission to enter will first be sent to the central controller from the controller of the neighboring region from which the plane originates. The central controller must reply to the other controller in a timely manner to accept the plane, then contact the plane to welcome it to the airspace. Those two cannot be performed in immediate succession but, instead, require waiting for the first party contacted (in this case the other controller) to reply before taking the final action. This delay allows for the interleaving of unit tasks but also requires the maintenance of the currently incomplete tasks in working memory. Messages from other tasks can arrive when a task is being processed, thus requiring some search of the text window to identify the messages relevant to a task. A message is composed by clicking a button above the relevant text window (e.g., accepting AC), then clicking in the graphical window on the intended recipient (e.g., another controller) and optionally the target of the message (i.e., a plane, unless it is the intended recipient, in which case this is omitted), then the send button above the graphic window. The message being composed is displayed at the top left of the display in a text window.

To measure performance on the task objectively, penalties were assessed for a variety of failures to act

in a timely manner. To evaluate the impact of system design, a decision support condition contrasted with a support condition were implemented to dissociate two aspects of multitasking behavior. In the standard condition, subjects had to parse the messages printed in the text windows on the right side of the screen to determine which planes needed attention and which functions needed to be performed on them. In the assisted condition, planes that require assistance were color coded in the graphical display on the left side of the screen according to the task that needed to be performed (green for accept, blue for welcome, orange for transfer, yellow for contact, magenta for speed change, and red for holding). This helped the subjects track visually which tasks needed to be attended to and removed any necessity to parse the text windows on the left, a complex and time-consuming task. Therefore, it dissociated the maintenance and updating of the queue of to-be-attended tasks from the resolution of conflicts between high-priority tasks. Two sets of scenarios were created: One set was provided to the developers as a model on which to base their designs, and another set was reserved to be used at the time of the competitive validation (i.e., the fly-off). Human performance data on the first set of scenarios were provided to the developers to fine-tune their model. The data from the second set of scenarios were withheld until after the fly-off for comparison with the model performance. The range of behavior requirements of both sets had the same scope, but the ways in which those behaviors were exercised were not identical, to test the robustness and predictiveness of the models.

5.3 Model Development

If it is to justify its structural costs, a cognitive architecture should facilitate the development of a model in several ways. It should limit the space of possible models to those that can be expressed concisely in its language and work well with its built-in mechanisms. It should provide for significant transfer from models of similar tasks, either directly in the form of code or more generally in the form of design patterns and techniques. Finally, it should provide learning mechanisms that allow the modeler to specify in the model only the structure of the task and let the architecture learn the details of the task in the same way that human cognition constantly adapts to the structure of its environment. These architectural advantages not only reduce the amount of knowledge engineering required and the number of trial-and-error development cycles, providing significant savings in time and labor, but also improve the predictiveness of the final model. If the "natural" model (derived a priori from the structure of the task, the constraints of the architecture, and the guidelines from previous models of related tasks) provides a good fit to the empirical data, one can be more confident that it will generalize to unforeseen scenarios and circumstances than if it is the result of post hoc knowledge engineering and data analysis. That is the approach that we adopted in developing a model of this task and indeed, more generally, our design and use of the ACT-R architecture.

Of course, in domains involving a large body of expertise, it makes sense to encode in the cognitive model the accepted knowledge of the field. But in synthetic tasks or in tasks involving new system design, specific established knowledge is usually inexistent or inaccessible. Thus, we did not try to reverse engineer the subjects' strategies but instead tried to develop the simplest and most natural model for the architecture. We organized the model around a few goal types with their associated productions. Goal types correspond closely to the unit tasks in human−computer interaction (Card et al., 1983) as well as to the tasks in task network models (e.g., Allender et al., 1995). Five goal types, called color-goal, text-goal, scan-text, scan-screen, and process, were defined, together with a total of 36 very simple productions. Goals were simple and would hold just a few elements, such as the aircraft currently being handled together with related information such as its position and the action to be performed, in accordance with architectural constraints. Overall, such model development need not take more than a few days. Two basic modes of human interaction with the simulation were defined: one in which the operator had to rely mostly on text messages scrolling in windows to identify events that required action (the text condition) and one in which aircraft on the radar screen that required action would turn a color corresponding to the action (the color condition). The simulation also had three speeds (low, medium, and high) that controlled how much time the subjects would have (10, 7.5, and 5 min, respectively) to perform a given number of actions.

The goal type color-goal was the top goal for the color condition. Five productions were defined that applied to that goal. They scanned the radar screen continuously, identified an aircraft that had turned color, mapped the color into the required action by relying on five simple memory chunks encoding the instructions that the subjects were given regarding the color-action mappings, then created a goal to perform the given action on the aircraft. The goal-type process executed the sequence of mouse clicks required to perform the action. Twelve productions were defined to handle the five possible actions. This required clicking on a button identifying the action, then on the aircraft, then perhaps on a neighboring controller, then finally on the send button.

As expected, the text condition was both more difficult for the subjects and slightly more complicated for the model. The goal type text-goal was the top goal for the text condition. Four productions were defined to cycle through the three text windows and the radar screen looking for aircraft requiring action by creating goals of type scan-text and scan-screen, respectively. A goal of type scan-text would handle the scanning of a single text window for a new message from another controller requesting action. A production was defined to scan the window systematically for such a message. If one was found, another production would attempt to retrieve a memory of handling such a request. Memories for such requests would be created automatically by the architecture when the corresponding goal was completed, but their availability was subject to their subsymbolic parameters, which were in turn subject to decay as well as reinforcement. If no memory could

be retrieved, the window would be scanned for another message, indicating completion. If none could be found, a process goal would be created to perform the action requested. Note that this is the same goal as in the color condition. A key component of the model was an additional production that would detect the onset of a new message in another window and interrupt the current goal to scan that window instead. This allowed the model to be sensitive to new events and handle them promptly. Scanning the radar screen was accomplished in a similar manner by goals of type scan-screen and their eight associated productions.

Finally, all the architectural parameters that control the performance of the simulation were left at their default values provided by previous models. A key aspect of our methodology, which is also pervasive in ACT-R modeling, is the use of Monte Carlo simulations to reproduce not only the aggregate subject data (such as the mean performance or response time) but also the variation that is a fundamental part of human cognition. Especially when evaluating system design, it is essential not only to capture an idealized usage scenario but as broad a range of performance as possible. In that view, the model does not represent an ideal or even average subject, but instead, each model run is meant to be equivalent to a subject run, in all its variability and unpredictiveness. For that to happen, it is essential that the model not merely be a deterministic symbolic system but also be able to exhibit meaningful nondeterminism. To that end, randomness is incorporated in every part of ACT-R's subsymbolic level, including chunk activations, which control their probability and latency of retrieval; production utilities, which control their probability of selections; and production efforts, which control the time that they spent executing.

Moreover, as has been found in other ACT-R models (e.g., Lerch et al., 1999), that randomness is amplified in the interaction of the model with a dynamic environment: Even small differences in the timing of execution might mean missing a critical deadline, which results in an error condition, which requires immediate attention, which might cause another missed deadline, and so on. To model the variation as well as the mean of subject performance, the model was always run as many times as there were subject runs. For that to be a practical strategy of model development, it is essential that the model run very fast, ideally significantly faster than real time. Our model ran up to five times faster than real time on a desktop PC, making it possible to run a full batch of 48 scenarios in about an hour an a half, enabling a relatively quick cycle of model development.

5.4 Modeling Results

Because the variability in performance between runs, even of the same subject, is a fundamental characteristic of this task, we ran as many model runs as there were subject runs. Figure 21 compares the mean performance in terms of penalty points for subjects and model for color (left three bars) and text (right three bars) condition by increasing workload level. The model matches the

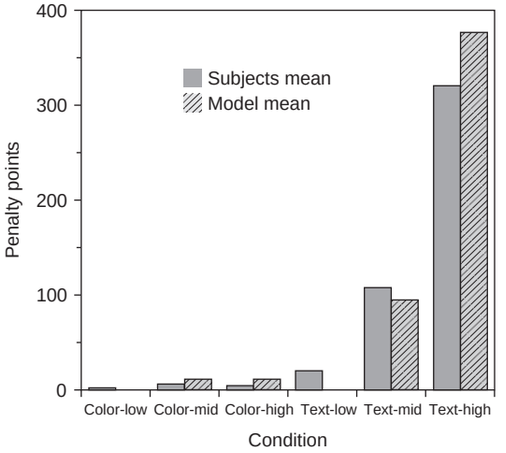

Figure 21 Mean performance as a function of workload and system design.

data quite well, including the strong effects of the color-versus- text condition and of workload for the unaided (text) condition.

Because ACT-R includes stochasticity in chunk retrieval, production selection, and perceptual/motor actions, and because that stochasticity is amplified by the interaction with a highly dynamic simulation, it can reproduce a large part of the variability in human performance, as indicated by Figure 22, which plots the individual subject and model runs for the two conditions that generated a significant percentage of errors (text condition in medium and high workload). The range of performance in the medium-workload condition is reproduced almost perfectly other than for two outliers, and a significant portion of the range in the high condition is also reproduced, albeit shifted slightly too upward. It should be noted that each model run is the result of an identical model that differs from another only in its run time stochasticity. The model neither learns from trial to trial nor is modified to take into account individual differences.

The model not only reproduces the subject performance in terms of total penalty points but also matches well to the detailed subject profile in terms of penalties accumulated under eight different error categories, as plotted in Figure 23. It should be emphasized that those errors were not engineered in the model but, instead, resulted directly from the limitations of the cognitive architecture applied to a demanding, fast-paced dynamic task.

The model also fits the mean response times (RTs) for each condition, as shown in Figure 24, which plots the detailed pattern of latencies to perform a required action for each condition and number of intervening events (i.e., number of planes requiring action between the time of a given plane requiring action and the time the action is actually performed). The model predicts very accurately the degradation of RT as more

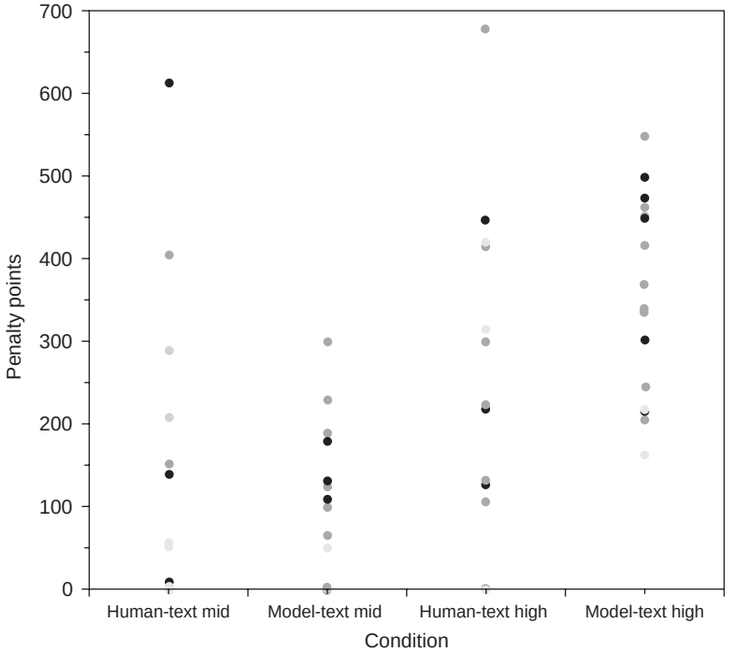

Figure 22 Mean performance as a function of workload and system design.

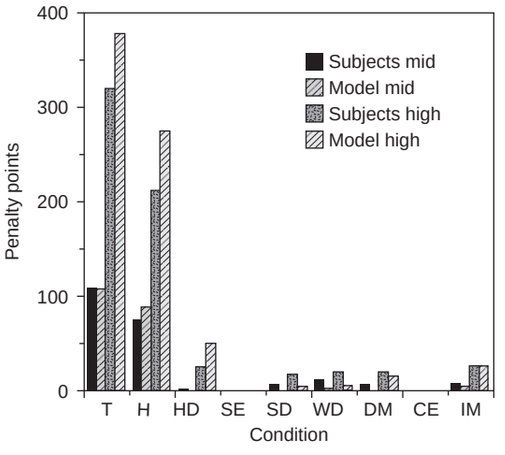

Figure 23 Penalty points for a variety of error categories.

events compete for attention, including the somewhat counterintuitive exponential (note that RT is plotted on a log scale) increase in RT as a function of number of events rather than a more straightforwardly linear increase. The differences in RT between conditions are primarily a function of the time taken by the perceptual processes of scanning radar screen and text windows.

Finally, the model reproduces the subjects' answers to the self-reporting workload test administered after each trial. As shown in Figure 25, the simple definition of workload described in Section 5.3 captures the main workload effects, specifically effects of display condition and schedule speed. The latter effect results from reducing the total time to execute the task (i.e., the denominator) while keeping the total number of events (roughly corresponding to the numerator) constant, thereby increasing the ratio. The former effect results from adding to the process tasks the message-scanning tasks resulting from onset detection in the text condition, thus increasing the numerator while keeping the denominator constant, thereby increasing the ratio as well. Another quantitative effect that is reproduced is the higher rate of impact of schedule speed in the text condition (and the related fact that workload in the slowest text condition is higher than workload in the fastest color condition). This is primarily a result of task embedding [i.e., the fact that a process task can be (and often is) a subgoal of another critical unit task (scanning a message window following the detection of an onset in that window)], thus making the time spent in the inner critical task count twice.

Lebiere (2004) reports the results of a second phase of the AMBR comparison in which the model had to learn how to categorize airplanes properly based

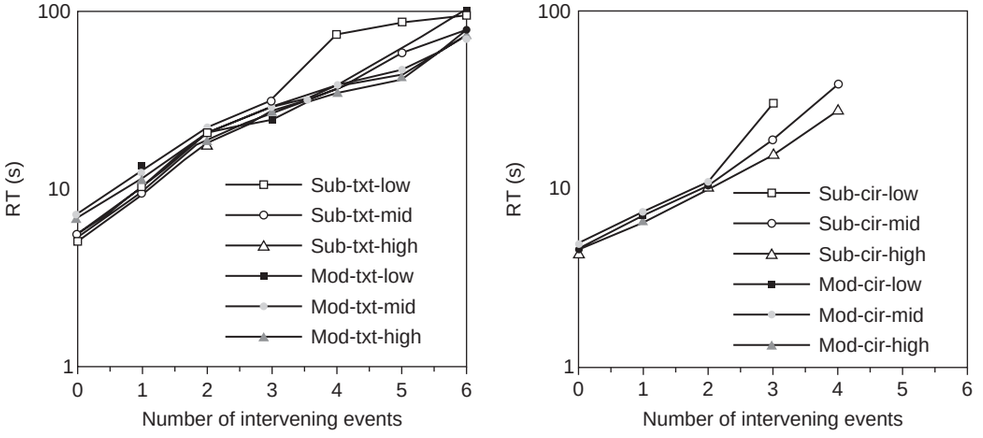

Figure 24 Response time as a function of intervening events.

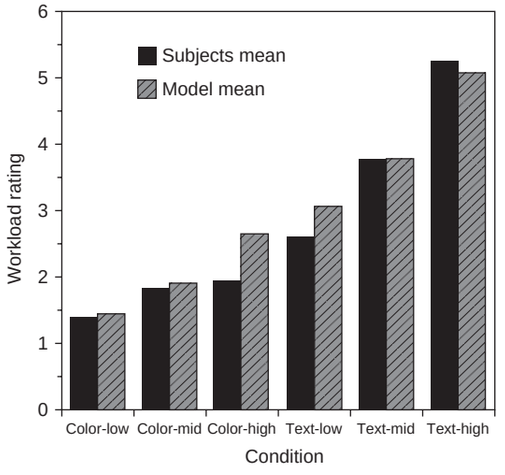

Figure 25 Workload levels for various speed and display conditions.

on a simple pass–fail feedback. This model is similar to the one described here but leverages even more extensively the subsymbolic aspects of the architecture, especially the learning equations described in the ACT-R introductory section, to perform the learning task as a constrained component of the entire task. In summary, the advantages of this model are that it is relatively simple, required almost no parameter tuning or knowledge engineering, provides a close fit to both the mean and variance of a wide range of subject performance measures as well as workload estimates, and suggests a straightforward account of multitasking behavior within the existing constraints of the ACT-R architecture.

6 INTEGRATION OF APPROACHES

Because ACT-R and IMPRINT were targeted at different behavioral levels, they complement each other perfectly. IMPRINT is focused on the task level, how high-level functions break down into smaller scale tasks, and the logic by which those tasks follow each other to accomplish those functions. ACT-R is targeted at the "atomic" level of thought, the individual cognitive, perceptual, and motor acts that take place at the subsecond level. As shown in Figure 19 and in the previous example, the current goal is a central concept in ACT-R which corresponds directly to the concept of unit task. At each cycle, a production will be chosen that best applies to the goal, knowledge might be retrieved from declarative memory, and perceptual and motor actions may be taken. Those cycles will repeat until the current goal is solved, at which point it is popped and another one is selected. The ACT-R theory specifies in detail the performance and learning that takes place at each cycle within a specific goal but has comparatively little to say about the selection of those goals. Since goals in ACT-R correspond closely to tasks in IMPRINT, that weakness matches IMPRINT's strength perfectly. Conversely, since IMPRINT requires the characteristics of each task to be specified as part of the model, ACT-R can be used to generate those detailed characteristics in a psychologically plausible way without requiring extensive data collection. Thus, an integrated ACT-R/IMPRINT is structured along as pictured in Figure 26.

An IMPRINT model specifies the network of tasks used to accomplish the functions targeted by the model (e.g., landing a plane and taxiing safely to the gate). The network specifies how higher order functions are decomposed into tasks and the logic by which these tasks are composed together. As input, it takes the distribution of times to complete the task and the accuracy with which the task is completed. It can also take as input

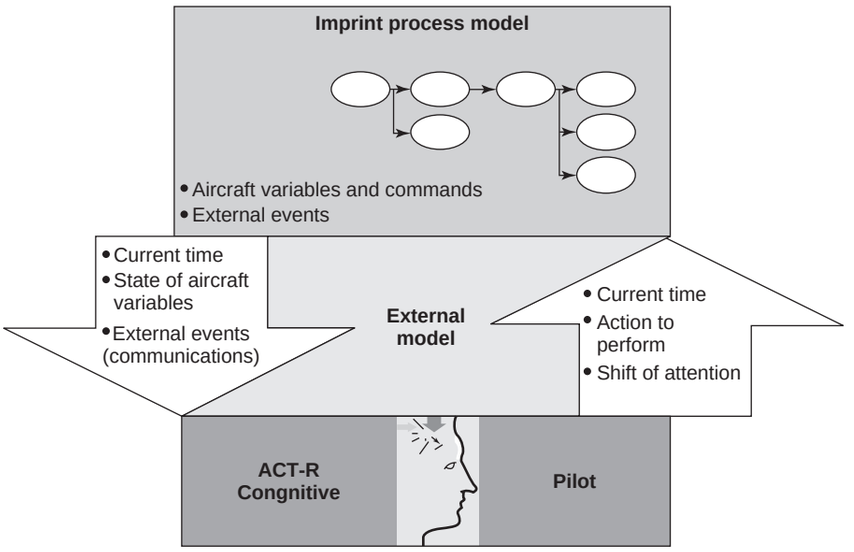

Figure 26 Integrated ACT-R/IMPRINT model.

the workload generated by each task. Additional inputs include events generated by the simulation environment. Finally, a number of additional general parameters, such as personnel characteristics, level of training, and familiarity and environmental stressors, can be specified. IMPRINT specifies the performance function by which these parameters modulate human performance. The outputs include mission performance data such as time and accuracy as well as aggregate workload data.

An ACT-R model specifies the knowledge structures, such as declarative chunks and production rules that constitute the user knowledge relevant to the tasks targeted by the model. It also specifies the goal structures reflecting the task structure and the architectural and prior knowledge parameters that modulate the model's performance. For each goal on which ACT-R is focused (i.e., made the current goal), it generates a series of subsecond cognitive, perceptual, and motor actions. The result of those actions is the total time to accomplish the goal as well as how the goal was accomplished, including any error that might result. Errors in ACT-R originate from a broad range of sources. They include memory failures, including the failure to retrieve a needed piece of information or the retrieval of the wrong piece of information; choice failures, including the selection of the wrong production rule; and attentional failures, such as the failure to detect the salient piece of information by the perceptual modules. Although those errors could arise because of faulty symbolic knowledge (either declarative or procedural), it is often not the case, especially in domains that involve highly trained crews. More often, those errors occur because the subsymbolic parameters associated with chunks or productions do not allow the model to access them reliably or quickly enough to be deployed in the proper situation.

Moreover, because those parameters vary stochastically and their effect is amplified by the interaction with a dynamic environment, those times and errors will not be deterministic but will vary with each execution, as is the case for human operators. Thus, the ACT-R model for a particular goal can be run whenever IMPRINT selects the corresponding task to generate the time and error distribution for that task in a manner that reflects the myriad cognitive, perceptual, and motor factors that enter into the actual performance of the task. As seen in the previous example, ACT-R can also generate workload estimates for each goal that reflect the cognitive demands of the actions taken to perform that particular subtask, then pass those estimates to IMPRINT, which can then combine them into global workload estimates for the entire task. ACT-R and IMPRINT have been unified in a single integrated development environment with the Human Behavior Architecture (HBA) tool (Warwick et al., 2008).

6.1 Sample Applications

As a practical application of the IMPRINT and ACT-R integration, a complex and dynamic task was selected for a modeling effort. Researchers with the National Aeronautics and Space Administration (NASA) were interested in developing models of pilot navigation while taxiing from a runway to a gate. Research on pilot surface operations had shown that pilots can commit numerous errors during taxi procedures (Hooey and Foyle, 2001). NASA was hoping to reduce the number and scope of pilot error during surface operations by

using information displays that would improve the pilots' overall situation awareness.

NASA researchers provided the IMPRINT and ACT-R modeling teams with data describing pilot procedures during prelanding and surface taxi operations. These data included videotapes of pilots in the NASA Ames Advanced Concept Flight Simulator (ACFS), which is a simulated cockpit capable of duplicating pilot taxiing operations. A detailed, scaled map of Chicago's O'Hare airport was also provided, which included runway signage. Other types of documentation were provided to give the IMPRINT and ACT-R modeling team the information necessary to duplicate runway taxiing behavior by pilots.

The IMPRINT and ACT-R modeling teams used the scaled map of Chicago's O'Hare airport to estimate the time between runway taxi turns. IMPRINT handled the higher level, task-oriented parts of the taxiing and landing operations (i.e., turning, talking on radio, looking at instrumentation), while ACT-R handled the more cognitive and decision-making parts of the task (i.e., remembering where to turn, remembering the taxi route). By using the scaled map of the airport, the IMPRINT and ACT-R teams were able to determine the amount of time between each taxi turn (based on an estimated plane speed that was correlated with the simulated speeds from the videotape data) and then use those data to estimate the decay rate for the list of memory elements (i.e., runway names) that the pilot would have to remember.

Using this integrated architecture allowed the team to represent a complex, dynamic task, and by exploiting each architecture's strengths, the modeling process was enhanced and streamlined. The resulting model could account for a broad range of possible taxiing errors within a constrained first-principle framework, as was the case for the stand-alone AMBR model, but in addition benefited by integration with the task network model, which provided a convenient task-based organizing framework to minimize the authoring requirements for the cognitive model as well as to provide a high-productivity tool to simulate the environment and aircraft with which the cognitive model interacts.

Craig et al. (2002) performed a similar integration of ACT-R into the combat automation requirements tool (CART)* model (Brett et al., 2002), a task network model used in the acquisition process of the joint strike fighter. The task to be performed was target acquisition, more specifically, management of the shoot list, which allows a pilot to select potential targets to be identified by high-resolution radar. Using a methodology similar to that described above, specific subtasks were identified for which additional cognitive fidelity was required and reimplemented in the form of ACT-R goals and associated production rules. ACT-R then interacted with the CART model, providing plausible performance for cognitive subtasks such as prioritizing targets and recalling items identified previously.

* The reader should note that the CART model capabilities are now subsumed into the IMPRINT tool.

7 SUMMARY

In this chapter we have reviewed the need for simulating performance of complex human-based systems as an integral part of system design, development, testing, and life-cycle support. We have also defined two fundamentally different approaches to modeling human performance, a reductionist approach and a first-principle approach. Additionally, we have provided detailed examples of two modeling environments that typify these two approaches along with representative case studies. Finally, we described an integrated tool that attempts to leverage the advantages of both approaches into an efficient and principled modeling package.

As we have stated and demonstrated repeatedly throughout this chapter, the technology for modeling human performance in systems is evolving rapidly. Furthermore, the breadth of questions being addressed by models is expanding constantly. Necessity being the mother of invention, we encourage the human factors practitioner to consider how computer simulation can provide a better and more cost-effective basis for human factors analysis and in turn stimulate further developments in modeling and simulation tools to better serve their needs.

REFERENCES

Alion Science and Technology (2009), *Micro Saint Sharp Version 3.0 User's Guide*, Alion Science and Technology/MA&D Operation, Boulder, CO.

Allender, L. (1995, December), personal communication.

Allender, L., Kelley, T., Salvi, L., Headley, D. B., Promisel, D., Mitchell, D., Richer, C., and Feng, T. (1995), "Verification, Validation, and Accreditation of a Soldier-System Modeling Tool," in *Proceedings of the 39th Human Factors and Ergonomics Society Meeting*, October 9-13, San Diego, CA, Human Factors and Ergonomics Society, Santa Monica, CA, pp. 1219–1223.

Anderson, J. R. (2007). *How Can the Human Mind Occur in the Physical Universe?* Oxford University Press, New York.

Anderson, J. R., and Lebiere, C. (1998), *The Atomic Components of Thought*, Lawrence Erlbaum Associates, Mahwah, NJ.

Anderson, J. R., and Schooler, L. J. (1991), "Reflections of the Environment in Memory," *Psychological Science*, Vol. 2, pp. 396–408.

Anderson, J. R., Matessa, M., and Lebiere, C. (1997), "ACT-R: A Theory of Higher Level Cognition and Its Relation to Visual Attention," *Human-Computer Interaction*, Vol. 12, No. 4, pp. 439–462.

Anderson, J. R., Bothell, D., Lebiere, C., and Matessa, M. (1998), "An Integrated Theory of List Memory," *Journal of Memory and Language*, Vol. 38, pp. 341–380.

Anderson, J. R., Qin, Y., Jung, K-W., & Carter, C. S. (2007). "Information-Processing Modules and their Relative Modality Specificity," *Cognitive Psychology*, 54(3), pp. 185–217.

Archer, S. G., and Adkins, R. (1999), *Improved Performance Research Integration Tool (IMPRINT) Analysis Guide*, Army Research Laboratory Technical Report, Aberdeen Proving Ground, MD.

Boff, K. R., Kaufman, L., and Thomas, J. P. (1986), *Handbook of Perception and Cognition*, Wiley, New York.

Brett, B. E., Doyal, J. A., Malek, D. A., Martin, E. A., Hoagland, D. G., and Anesgart, M. N. (2002), *The Combat Automation Requirements Testbed (CART) Task 5 Interim Report: Modeling a Strike Fighter Pilot Conducting a Time Critical Target Mission*, AFRL-HE-WP-TR-2002-0018, Wright-Patterson Air Force Base, OH.

Byrne, M. D., and Anderson, J. R. (2001), "Serial Modules in Parallel: The Psychological Refractory Period and Perfect Time-Sharing," *Psychological Review*, Vol. 108, pp. 847–869.

Card, S. K., Moran, T. P., and Newell, A. (1983), *The Psychology of Human-Computer Interaction*, Lawrence Erlbaum Associates, Mahwah, NJ.

Chong, R. (2001), "Low-Level Behavioral Modeling and the HLA: An EPIC-Soar Model of an Enroute Air-Traffic Control Task," in *Proceedings of the 10th Conference on Computer Generated Forces and Behavior Representation*, Norfolk, VA, pp. 27–35, Orlando, FL, Division of Continuing Education, University of Central Florida.

Craig, K., Doyal, J., Brett, B., Lebiere, C., Biefeld, E., and Martin, E. (2002), "Development of a Hybrid Model of Tactical Fighter Pilot Behavior Using IMPRINT Task Network Model and ACT-R," in *Proceedings of the 11th Conference on Computer Generated Forces and Behavior Representation*, Orlando, FL.

Dahl, S. G., Allender, L., Kelley, T., and Adkins, R. (1995), "Transitioning Software to the Windows Environment: Challenges and Innovations," in *Proceedings of the 1995 Human Factors and Ergonomics Society Meeting*, Human Factors and Ergonomics Society, Santa Monica, CA, October, pp.1224–1227.

Eggleston, R. G., Young, M. J., and McCreight, K. L. (2001), "Modeling Human Work through Distributed Cognition," in *Proceedings of the 10th Annual Conference on Computer Generated Forces and Behavior Representation*, Norfolk, VA.

Engh, T., Yow, A., and Walters, B. (1998), *An Evaluation of Discrete Event Simulation for Use in Operator and Crew Performance Evaluation*, report to the Nuclear Regulatory Commission, NRC, Washington, DC.

Farmer, E. W., Belyavin, A. J., Jordan, C. S., Bunting, A. J., Tattershall, A. J., and Jones, D. M. (1995), *Predictive Workload Assessment: Final Report*, DRA/AS/MMI/CR95100/, Defence Research Agency, Farnborough, Hampshire, England, March.

Gawron, V. J., Laughery, K. R., Jorgensen, C. C., and Polito, J. (1983), "A Computer Simulation of Visual Detection Performance Derived from Published Data," in *Proceedings of the 2nd International Ohio State University Aviation Psychology Symposium*, Columbus, OH, April, pp. 465–471.

Gore, B. F., Hooey, B. L., Wickens, C. D., and Scott-Nash, S. (2009), "A Computational Implementation of a Human Attention Guiding Mechanism in MIDAS v5," in *Proceedings of the Human Interaction International 2009 Annual Conference*, San Diego, CA.

Gray, W. D., Young, R. M., and Kirschenbaum, S. S. (1997), Special Issue: Cognitive Architectures and Human-Computer Interaction, *Human-Computer Interaction*, Vol. 12, No. 4.

Gunzelmann, G., Byrne, M. D., Gluck, K. A., & Moore, L. R. (2009). "Using computational cognitive modeling to predict dual-task performance with sleep deprivation," *Human Factors*, Vol. 51, No. 2, pp. 251–260.

Hahler, B., Dahl, S., Laughery, R., Lockett, J., and Thein, B. (1991). "CREWCUT: A Tool for Modeling the Effects of High Workload on Human Performance," paper presented at the 35th Annual Human Factors Society Meeting, San Francisco.

Hoagland, D. G., Martin, E. A., Anesgart, M., Brett, B. S., Lavine, N., and Archer, S. G. (2001), "Representing Goal-Oriented Human Performance in Constructive Simulations: Validation of a Model Performing Complex Time-Critical-Target Missions," in *Proceedings of the Simulation Interoperability Workshop*, Orlando, FL, Spring.

Hooey, B. L., and Foyle, D. C. (2001), "A Post-Hoc Analysis of Navigation Errors during Surface Operations: Identification of Contributing Factors and Mitigating Strategies," in *Proceedings of the 11th Symposium on Aviation Psychology*, Ohio State University, Columbus, OH.

Horrey, W. J., and Wickens, C. D. (2003), "Multiple Resource Modeling of Task Interference in Vehicle Control, Hazard Awareness and In-Vehicle Task Performance," in *Proceedings of the Second International Driving Symposium on Human Factors in Driver Assessment, Training and Vehicle Design*, Park City, Utah, pp. 7–12.

Horrey, W. J., Wickens, C. D., and Consalus, K. P. (2006) "Modeling drivers' visual attention allocation while interacting with in-vehicle technologies," in *Journal of Experimental Psychology: Applied*, Vol. 12, No. 2, pp. 67–86.

John, B. E., Prevas, K., Salvucci, D. D., & Koedinger, K. (2004). "Predictive human performance modeling made easy." in *CHI 2004, ACM Conference on Human Factors in Computing Systems, CHI Letters* 6(1).

Klein, G. (1998), *Sources of Power: How People Make Decisions*, The MIT Press, Cambridge, MA.

LaVine, N. D., Peters, S. D., and Laughery, K. R. (1995), *A Methodology for Predicting and Applying Human Response to Environmental Stressors*, Micro Analysis and Design, Inc., Boulder, CO, December.

Lawless, M. L., Laughery, K. R., and Persensky, J. J. (1995), *Micro Saint to Predict Performance in a Nuclear Power Plant Control Room: A Test of Validity and Feasibility*, NUREG/CR-6159, Nuclear Regulatory Commission, Washington, DC, August.

Lebiere, C. (2001), "A Theory-Based Model of Cognitive Workload and Its Applications," in *Proceedings of the 2001 Interservice/Industry Training, Simulation and Education Conference (I/ITSEC)*, National Defense Industrial Association (NDIA), Arlington, VA.

Lebiere, C. (2004), "Constrained Functionality: Application of the ACT-R Cognitive Architecture to the AMBR Modeling Comparison," in *Modeling Human Behavior with Integrated Cognitive Architectures: Comparison, Evaluation, and Validation*, R. W. Pew and K. A. Gluck, Eds., Lawrence Erlbaum Associates, Mahwah, NJ.

Lerch, F. J., Gonzalez, C., and Lebiere, C. (1999), "Learning under High Cognitive Workload," in *Proceedings of the 21st Conference of the Cognitive Science Society*, Lawrence Erlbaum Associates, Mahwah, NJ, pp. 302–307.

Lovett, M. C., Reder, L. M., and Lebiere, C. (1999), "Modeling Working Memory in a Unified Architecture: An ACT-R Perspective," in *Models of Working Memory*, A. Miyake and P. Shah, Eds., Cambridge University Press, Cambridge, MA.

Matessa, M., and Mui, R. (2009), "GRBIL: A Tool for Dynamic Operator and Environment Modeling," in *Advanced Decision Architectures for the Warfighter: Foundations and Technology*, P. McDermott and L. Allender, Eds., Army Research Laboratory, Boulder, CO, pp 237–254.

McCracken, J. H., and Aldrich, T. B. (1984), *Analysis of Selected LHX Mission Functions: Implications for Operator Workload and System Automation Goals*, Technical Note ASI 479-024-84(B), Anacapa Sciences, Ft. Rucker, AL, June.

Newell, A. (1990), *Unified Theories of Cognition*, Harvard University Press, Cambridge, MA.

Newell, A., and Rosenbloom, P. S. (1981), "Mechanisms of Skill Acquisition and the Power Law of Practice," in *Cognitive Skills and Their Acquisition*, J. R. Anderson, Ed., Lawrence Erlbaum Associates, Mahwah, NJ, pp. 1–56.

Newell, A., and Simon, H. A. (1972), *Human Problem Solving*, Prentice-Hall, Englewood Cliffs, NJ.

O'Hara, J. M., Brown, W. S., Stubler, W. F., Wachtel, J. A., and Persensky, J. J. (1995), *Human-System Interface Design Review Guideline: Draft Report for Comment*, NUREG-0700 Rev. 1, U.S. Nuclear Regulatory Commission, Washington, DC.

Pew, R. W., and Gluck, K. A. (2004). *Modeling Human Behavior with Integrated Cognitive Architectures: Comparison, Evaluation, and Validation*, Lawrence Erlbaum Associates, Mahwah, NJ.

Pew, R. W., and Mavor, A. S. (1998), *Modeling Human and Organizational Behavior: Application to Military Simulations*, National Academy Press, Washington, DC.

Plott, B. (1995), *Software User's Manual for WinCrew, the Windows-Based Workload and Task Analysis Tool*, U.S. Army Research Laboratory, Aberdeen Proving Ground, MD.

Plott, B., Quesada, S., Kilduff, P., Swoboda, J., and Allender, L. (2004) "Using an Information-Driven Decision-Making Human Performance Tool to Assess U.S. Army Command, Control, and Communication Issues," in *Proceedings from the Human Factors and Ergonomics Society 48th Annual Meeting*.

Roth, J. T. (1992), *Reliability and Validity Assessment of a Taxonomy for Predicting Relative Stressor Effects on Human Task Performance*, Technical Report 5060-1, prepared under contract DNA001-90-C-0139, Micro Analysis and Design, Boulder, CO, July.

Rubin, D. C., and Wenzel, A. E. (1990), "One Hundred Years of Forgetting: A Quantitative Description of Retention," *Psychological Review*, Vol. 103, pp. 734–760.

Siegel, A. I., and Wolf, J. A. (1969), *Man-Machine Simulation Models*, Wiley-Interscience, New York.

Taatgen, N. A. (2001), "A Model of Individual Differences in Learning Air Traffic Control," in *Proceedings of the 4th International Conference on Cognitive Modeling*,Gray, W. Altmann, E., Cleeremans, and Schunn, C. Eds., Lawrence Erlbaum Associates, Mahwah, NJ, pp. 211–216.

Warwick, W., Archer, R., Hamilton, A., Matessa, M., Santamaria, A., Chong, R., Allender, L., and Kelley, T. (2008). "Integrating Architectures: Dovetailing Task Network and Cognitive Models," in *Proceedings of Behavior Representation in Modeling and Simulation (BRIMS '08)*, Orlando, FL, University of Central Florida..

Warwick, W., and Santamaria, A. (2009), "Modeling the Recognition Primed Decision," in *Advanced Decision Architectures for the Warfighter: Foundations and Technology*, P. McDermott and L. Allender, Eds., final report for DAD19-01-2-0009, Army Research Laboratory Advanced Decision Architectures Collaborative Technology Alliance, Boulder, CO, pp. 273–288.

Wickens, C. D. (1984), *Engineering Psychology and Human Performance*, Charles E. Merrill, Columbus, OH.

Wickens, C. D., and McCarley, J. S. (2008), *Applied Attention Theory*, Taylor & Francis, Boca Raton, FL.

Wickens, C. D., Sandry, D. L., and Vidulich, M. (1983), "Compatibility and Resource Competition between Modalities of Input, Central Processing, and Output," *Human Factors*, Vol. 25, pp. 227–248.

Wortman, D. B., Duket, S. D., Seifert, D. J., Hann, R. L., and Chubb, A. P. (1978), *Simulation Using SAINT: A User-Oriented Instruction Manual*, AMRL-TR-77-61, Aerospace Medical Research Laboratory, Wright-Patterson Air Force Base, OH, July.

Zachary, W., Santarelli, T., Ryder, J., Stokes, J., and Scolaro, D. (2001), "Developing a Multi-Tasking Cognitive Agent Using the COGNET/iGEN Integrative Architecture," in *Proceedings of the 10th Annual Conference on Computer Generated Forces and Behavior Representation*, Norfolk, VA.

Queries in Chapter 32

Q1. We have kept this equation as per manuscript "==". Please confirm is this correct?