

21 Model-based Design of Human Interaction with Complex Systems

CHRISTINE M. MITCHELL

21.1 INTRODUCTION

This chapter proposes a methodology for designers of human interaction with complex systems. It contains a generic characterization of the proposed methodology—model-based design—and a description of a *specific* methodology, which uses the operator function model (OFM) (Mitchell, 1996a). The chapter begins with an overview of the current state of design for human–system interaction in complex systems and presents a set of critical design issues that model-based design can help address.

21.2 HUMAN INTERACTION WITH COMPLEX SYSTEMS: THE SYSTEMS, TASKS, AND USERS

Human–system interaction addresses the set of issues concerning the manner in which a user or operator interacting with a system carries out activities to meet system or user objectives. The particular forms or styles of human–system interaction govern the extent to which both the overall system and users meet objectives in efficient and safe ways and reduce the occurrence of human error.

In many systems, human–*system* interaction includes human–*computer* interaction and its associated design issues. Given the proliferation of computers throughout the home, office, and industrial workplace, humans often interact with systems through a computer medium. Thus, humans “program” a computer that in turn instructs the system. Human–system interaction via a computer medium occurs in the kitchen (we program the microwave for a given power level and time), family room (we program the television/VCR to record a desired show), office (we configure word processing and spreadsheet templates), and workplace (pilots program the flight management system and manufacturing shop-floor personnel “teach” desired operations to a computer-based machining station).

Human–system interaction includes a wide variety of systems and a wide variety of humans who interact with them. One useful way of organizing and sorting out the myriad of associated design issues is to consider types of users and types of systems with which the users interact. Design issues and characteristics are widely acknowledged to be context-dependent, that is, both task- and user-dependent. As most human factors engineering practitioners (Cardosi and Murphy, 1995) and human–computer interaction texts (Kantowitz and Sorkin, 1983; Preece, 1994) point out: identifying and understanding the in-

tended users and specifying the users' needs are *the* fundamental prerequisites to effective design.

This chapter concerns systems that are sufficiently complex that personnel who manage them must be trained in system operation or maintenance. As such, we characterize both the types of systems and design issues of interest, and circumscribe, to some extent, the generalizability of the discussion.

The term *operator* is used rather than the more general term *user* to indicate the types of systems and human–system interactions under discussion. This chapter concerns work-related systems, as opposed to, for example, home-entertainment systems. The operators of such systems are usually highly trained, professional personnel. They have a set of reasonably well-specified goals with respect to system performance and, likewise, a set of reasonably well-defined activities with respect to system operation and maintenance. Thus, operator refers to a person who is “well-trained and well-motivated” with respect to the operation of a particular system (Baron, 1984; Sheridan and Ferrell, 1974).

Users who are casual, intermittent, or naive do not characterize the types of human–system interactions addressed in this chapter. We hasten to note that such interactions are not unimportant: for some systems, each of us is a naive or casual user and Norman's (1988) *Psychology of Everyday Things* speaks poignantly to the importance of effective design for such systems. Design issues, however, are far different for operators who are trained in system operation than they are for casual users. Systems with operators are typically complex and the cost of error may be high, and occasionally catastrophic. Often such systems are dynamic; that is, the state of the system evolves over time without waiting for operator input. Error-free activity or designs that allow operators to recognize and recover from errors are important, since the costs of unrecovered errors may be unacceptable. Examples of such systems include aerospace, aviation, air-traffic control, process control, telecommunications, and manufacturing systems.

21.3 EMERGING TECHNOLOGY AND DESIGN

New computer-based information and control technologies require careful consideration of human factors and human–system interaction issues. New technologies succeed or fail based on a designer's ability to reduce incompatibilities between the characteristics of the system and the characteristics of the people who operate, maintain, and troubleshoot it (Casey, 1993). Moreover, well-designed human interfaces are essential to ensure reliable human performance and system safety (International Nuclear Safety Advisory Group, 1988; Moray and Huey, 1988; O'Hara, 1994; Woods et al., 1994). O'Hara (1994) suggests that safety depends, in part, on the extent to which a design reduces the chances of human error, enhances the chances of error recovery, or safeguards against unrecovered human errors, e.g., provides an *error-tolerant* system (Billings, 1997; Nagle, 1988; Rasmussen, 1987; Woods et al., 1994).

Experience in a wide variety of systems and applications suggests that the use of sophisticated computer technology in the human interface, such as computer displays, controls, and operator aids, raises serious issues related to the way humans operate, troubleshoot, and maintain these systems (Billings, 1997; Casey, 1993; Sheridan, 1992; Woods et al., 1994). Several recent studies highlight the importance of the “human factor” when incorporating computer technology in safety-critical systems. One study, conducted by a

subcommittee convened by the Federal Aviation Administration (FAA), found that interfaces between flight crews and modern flight decks to be critically important in achieving the administration's zero-accident goal (Abbott, 1996). The committee, however, noted a wide range of shortcomings in designs, design processes, and certification processes for current and proposed systems. Other examples in which the human factor is identified as a significant contributor abound. Two recent studies of incidents in nuclear power plants identified human factors as the primary concern. One study, which classified failures in nuclear plants that included digital subsystems, found that human factors issues, including human-machine interface errors, were a "significant" category (Ragheb, 1996). The other study noted that over the 13-year study period, the trend in most error categories was decreasing or flat, whereas events attributable to inappropriate human actions "showed a marked increase" (Lee, 1994). Woods and his colleagues (1994) report that a survey of anesthetic incidents in the operating room attribute between 70 and 75 percent of the incidents surveyed to the human element. Moreover, in a set of diverse applications, including medical, aerospace, process control, and aviation, Woods et al. (1994) identify characteristic and recognizable human-system interaction problems that manifest themselves repeatedly across a broad and varied range of systems. These systems, however, all have in common newly introduced "automation" or human interaction with advanced technology systems.

As a result there is a widely held belief that while overall reliability in complex and/or safety-critical systems is increasing, the number of problems attributed to human error is increasing or, at least, not decreasing. The FAA committee on human factors stated that human-related problems, as a percentage of the whole, are increasing (Abbott, 1996). Billings (1997) notes. . . "that it has long been an article of faith that from 65% to 80% of air transport accidents are attributable in whole or part to human error" (p. 4).

Human factors engineers and researchers are quick to note that many of these problems are *design* problems, not inherent deficiencies of the technology (Abbott, 1996; Sheridan, 1992; Wiener, 1989; Woods et al., 1994). Skillful design that uses emerging technology effectively can make a system safer, more efficient, and easier to operate (Abbott, 1996).

Many industries currently make extensive use of digital technology: Fossil-fuel plants, a wide range of process control industries (e.g., textile, steel, paper), manufacturing, aerospace, aviation, and air-traffic control systems increasingly use digital technology for operator displays and controls, aids, and control systems themselves. Despite many computer glitches, most industries perceive a substantial benefit to overall system safety and effectiveness by incorporating digital technology. The most striking example may be in aviation. Although there are many areas that require improvement, incorporation of digital technology in commercial aircraft is widely believed to have increased overall safety and system efficiency (Abbott, 1996). Reviews of cockpit automation such as those appearing in *Aviation Week and Space Technology* (1995a, b) in early 1995, note problems or glitches in the human interface, but none of the parties involved in the flight-deck dialog (e.g., pilots, airlines, air frame manufacturers, or regulatory bodies) suggests that these problems necessitate a return to conventional analog technology. The belief is that digital technology, despite many serious and persistent problems, is often beneficial, and, with more effective design and evaluation, will continue to improve the effectiveness of the overall system (Abbott, 1996; Parasuraman and Riley, 1997). Thus, the challenge is to *identify* important human-system interaction issues and to develop design methodologies to address these issues so that safe and effective systems are assured.

21.4 HUMAN–SYSTEM INTERACTION ISSUES

The proliferation of inexpensive computers and computer-interface technologies raises a seemingly uncountable set of design opportunities and challenges. Historically, design and implementation of new concepts or technologies were slow and change was incremental. For the past 10 to 20 years, however, there have been tremendous economic and social pressures to use rapidly emerging and relatively inexpensive digital technologies to enhance system performance and safety, while concurrently reducing human error. Human factors specialists suggest, however, that the research base supporting the design of human–system interaction in advanced technology systems, in comparison to knowledge available to previous generations of designers, has not kept pace with technology development (O'Hara, 1994). O'Hara and his colleagues suggest that there are many more *issues* than *answers* for how to design computer-based interactions with complex systems.

Some guidance does exist. In fact, some argue that too many guidelines exist, with the result that designers are often confused and even the best guidelines ignored (Lund, 1997; Woods, 1992). Lund (1997), in a recent review of human–computer interface guidelines, reported on recent guideline development efforts including the compilation of 3,700 guidelines for the Belgian government with an accompanying bibliography of 1,200 entries (Vanderdonckt, 1994). Available guidelines include the widely cited Smith and Mosier (1988) guidelines for “generic” human–computer interaction, Space Station Freedom Human-Computer Interface Guidelines [National Aeronautics and Space Administration (NASA), 1988], Human Factors in the Design and Evaluation of Air Traffic Control Systems (Cardosi and Murphy, 1995), and User Interface Guidelines for NASA Goddard Space Flight Center (NASA, 1996). The guidance contained in such documents is limited, however. Anthologies of guidelines primarily address low-level physical or generic human–computer interaction issues rather than higher level cognitive issues (Wadlow, 1996; Woods, 1992). Cognitive issues, such as mode error, cognitive workload, and situation awareness, are increasingly important in complex systems that incorporate high levels of computer control and automation (Parasuraman and Mouloua, 1996).

Guidance that extends beyond guidelines is often conceptual; Lund (1997) calls such guidelines *design maxims*. Most maxims are unarguably correct, but there is no consensus on how to implement them in an actual design. For example, Billings's (1997) *human-centered automation* is a timely concept that should permeate all human–system interaction. But knowledge or methods for how to create a design that is human centered, or to evaluate a design or prototype to ensure that it is human centered do not exist. Design maxims are sometimes formulated as features to avoid rather than characteristics that a design should possess. Wiener's (1989) notion of *clumsy automation* describes an undesirable design feature: automation that “helps” when workload is low but makes excessive demands when workload is high. No one would disagree with the maxim that a design should *not* be clumsy, but methods to avoid creating clumsy automation do not exist.

Such maxims might be thought of as *conceptual normative characteristics*. They provide concepts that a good design should possess or avoid, but are specified at such high levels that there is a great deal of difficulty or uncertainty in understanding how to apply effectively a particular maxim in a specific application.

Finally, because the science and engineering basis of human factors for computer-based systems is so new, little guidance is generally applicable. Most concepts must be carefully implemented as a prototype, and subsequently evaluated in the context of the application under consideration (Cardosi and Murphy, 1995; O'Hara, 1994).

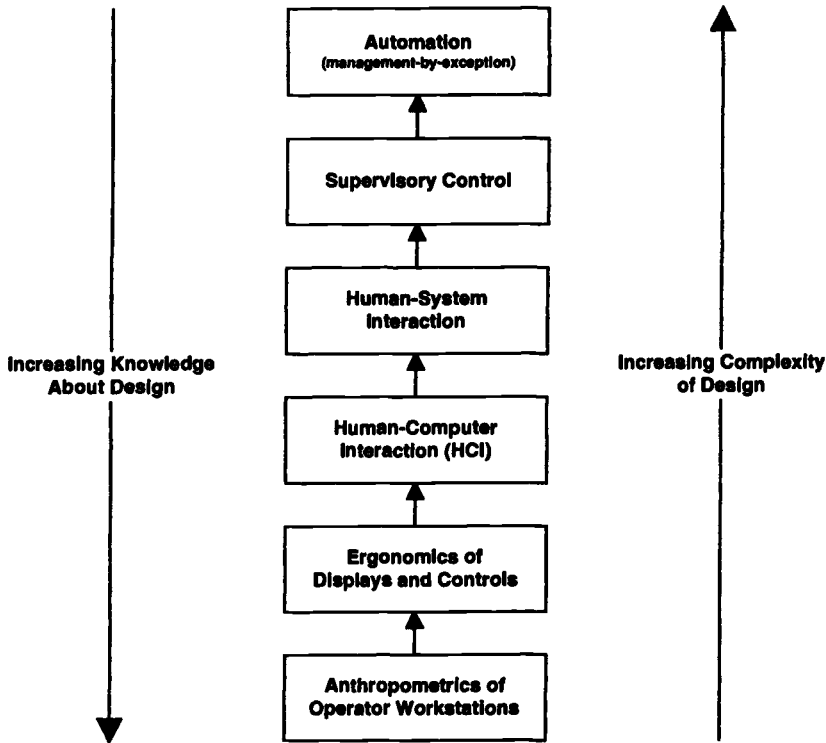


Figure 21.1 Design knowledge for human interaction with complex systems (HICS).

Figure 21.1 depicts a hierarchy of issues related to human factors in the use of advanced technologies in complex systems. The amount of existing knowledge is inversely related to the levels in the hierarchy. Thus, more generally accepted and widely applicable design knowledge is available for lower-level issues. Moving up the hierarchy, design knowledge is less detailed and more conceptual, and design experience is often not applicable across a range of applications (e.g., from office automation to control rooms).

21.4.1 Anthropometrics of Operator Workstations

At the lowest level, anthropometry—the science of establishing the proper size or height of equipment and space—there is a good deal of knowledge. Like guidelines for conventional displays and controls, the hardware associated with computer-based workstations is no longer subject to widespread debate. There are standards and recommendations for computer-based workstations that specify working levels, desk height, foot rests, document holders, task lighting, and viewing distances (Cakir et al., 1980; Cardosi and Murphy, 1995).

21.4.2 Ergonomics of Displays and Controls

At the next level are issues that specify the characteristics of computer-based displays and controls. Issues include font size, use of color, input devices, and types of displays (e.g.,

visual, audio). Knowledge here blends commonly accepted guidelines with emerging research results, which are often task-specific or user-dependent. For example, despite many disputes when first introduced in the late 1970s, a computer-input device, called a mouse, was empirically shown to produce performance superior to available alternatives for pointing tasks. Today, mice are routinely packaged with computer hardware. The number of buttons on a mouse, however, still varies from one to three. Recommendations for the best design vary depending on task and user requirements, as well as designer and user preferences.

The issue of the “ideal” or “best” number of buttons on a mouse illustrates the state of a great deal of human factors engineering knowledge. Often there is no single, best answer (Wadlow, 1996). In some cases, within some range, some characteristics may not make a discernible difference and users can readily adapt to a range of characteristics. In other situations, an acceptable solution is task- and/or user-dependent. Mock-ups or prototypes and empirical evaluations are needed to ensure that a proposed design meets user needs.

21.4.3 Human–Computer Interaction

Level-three, human–computer interaction (HCI) is the area to which the most study has been devoted. This area receives widespread academic and industry attention. Most current guidelines address this level of consideration. Issues include style of windows, window management, and dialog type. HCI guidelines address application-free, or generic, characteristics of the human–computer interface. From the point of view of design in a complex system, such issues might be thought of as the computer medium’s *interface syntax*. Most guidelines for HCI specify attributes that are *likely* to be effective that *may* be desirable, or that *must be evaluated* in the context of the application (see Cardosi and Murphy, 1995). Again, at this level, there is a wide range of acceptable designs and no indication that a single, best design is emerging. Following routine HCI guidelines, this level of human factors issues can be adequately, though not optimally, addressed.

21.4.4 Human–System Interaction

The transition to the fourth level of consideration, human-system interaction, marks the point where many serious issues concerning human capabilities and limitations and human attributes that affect safe and effective system operation emerge. This is also the level where there are many more questions than even marginally satisfactory answers. At this time, the majority of issues arises, and must be addressed, in the context of the application-specific tasks for which a computer interface will be used.

Early conceptual issues, still not adequately resolved, include “getting lost” and the “keyhole effect” (Woods, 1984), gulfs of evaluation and execution (Hutchins et al., 1986; Norman, 1988), and the inability of designers to aggregate and abstract the vast amount of data made available by computers into meaningful information (Mitchell and Miller, 1986; Mitchell and Sundström, 1997; Rasmussen and Goodstein, 1988). Essentially, issues at this level concern the *semantics* of the computer interface: how to design information displays and system controls that enhance human capabilities and compensate for human limitations.

“Getting lost” describes the phenomenon in which a user, or operator, becomes lost in a wide and deep forest of display pages (Woods, 1984). As computer displays proliferate, so too does the potential number of display pages available to operators. One recent de-

scription of a new digital nuclear power plant reported that operators have as many as 15,000 display pages from which to choose. Operators are often overwhelmed with choices, and empirical research shows that some operators use information suboptimally in order to reduce the number of transitions among display pages (Mitchell and Miller, 1986; Woods, 1984). When issues of across-display information processing are ignored, the computer screen becomes a serial data presentation medium with which the user has a *keyhole* through which data are observed. The limitations on short-term memory suggest that a keyhole view can severely limit information processing and increase cognitive workload, particularly in comparison to parallel displays used in analog control systems.

The gulfs of evaluation and execution describe attributes of a design that affect cognitive workload. They describe the conceptual distances between decisions and actions that an operator must undertake and features of the interface to support them. The greater either distance, the less desirable the interface (Hutchins, 1983; Hutchins et al., 1985; Norman, 1988). The gulf of evaluation characterizes the difficulty with a particular design as a user moves from perceiving data about the system to making a situation assessment or a forming an intention to make a system change. The gulf of execution characterizes the difficulty with a particular design as a user moves from forming the intention to make a change to executing the action or series of actions required to bring about the change. Display characteristics such as data displayed at too low a level or decisions that require the operator to access several display pages sequentially, mentally or manually extracting and integrating data along the way, are likely to create a large gulf of evaluation. Likewise, control procedures that are sequential, complex, or require a large amount of low-level input from the operator are likely to create a large gulf of execution.

Finally, and particularly true of control rooms in which literally thousands of data items are potentially available, defining information—that is, the useful part of data—is a serious concern. The keyhole effect and getting lost are due in part to the vast number of display pages that result when each sensed datum is presented on one or more display pages in a format close to the manner in which the data are recorded. Rasmussen (1986) characterizes such displays as representative of single-sensor–single-indicator (SSSI) design. Reminiscent of analog displays, and because displays may be used for many different purposes, data are often presented at the lowest level of detail possible, typically the sensor level. There is rarely an effort to analyze the information and control needs for specific operational activities and to use that knowledge to design operator displays and controls that support system control goals given current system state.

A good deal of conventional wisdom characterizing good human–system integration is available with the goal of reducing the cognitive workload associated with information extraction, decision making, and command execution in complex dynamic systems. As noted before, Woods (1984) cautions against designs that create a keyhole effect and proposes the concept of *visual momentum* to mitigate it. Visual momentum proposes that display-page design attempts to limit the number of transitions among pages for a single monitoring or control task and that pages be consistent for the task at hand; operators should not be required to dig around or aggregate low-level data to obtain needed information. Hutchins et al. (1986) propose the concept of *direct engagement* to bridge the gulfs of evaluation and execution, that is, the use of direct manipulation to implement displays and controls. Others propose the use of system or task models to organize, group, and integrate data items and sets of display pages (Kirlik, et al., 1994; Mitchell, 1996b; Rasmussen et al., 1994; Vicente and Rasmussen, 1992; Woods, 1995).

Such concepts are well understood and have broad agreement at high levels. This agree-

ment, however, does not translate into specific or widely accepted design methods or guidelines. For example, everyone agrees that a newly designed interface should not raise the level of required problem-solving behavior as defined by Rasmussen's (1983) SRK (skills–rules–knowledge) problem-solving paradigm, yet agreement for how to design displays that respect this principle do not exist.

Thus, at this time, design of operator workstations is often an art, depending on designer intuition and skills. As such, design prototypes, walk-throughs, and empirical evaluations are all necessary components to ensure effective design (Cardosi and Murphy, 1995). A prototype communicates some things directly from designers to users, such as the set of available displays, display formats, and alternative input options. User preferences, however, do not guarantee effective human–system interaction. Recent data show that user performance does not necessarily improve with designs based on user preferences. When implemented, preferred styles sometimes failed to enhance performance, and in some cases actually degrade it (Andre and Wickens, 1995). Thus, prototypes alone are insufficient. Prototypes and their implementation for carrying out operator control activities must be assessed. Walk-throughs with actual users performing realistic tasks help to ensure effective design for higher-level cognitive and system control issues. Even walk-throughs alone, however, are not sufficient. Karat et al. (1992) compared walk-throughs with experiments. The experiments yielded analyzable data of user and system performance. Subjects in both walk-throughs and experiments were actual users who carried out realistic tasks from the domain in which they were expert. The results showed that experimental evaluations identified new, and sometime very serious, problems that walk-throughs failed to identify. Thus, a proposed design must be rigorously and empirically evaluated to ensure that it enhances significantly operator and overall system effectiveness (Cardosi and Murphy, 1995).

21.4.5 Supervisory Control

Introduced by Sheridan in 1960, the term *supervisory control* characterizes the change in an operator's role from manual controller to monitor, supervising one or more computer-based control systems (Sheridan, 1976). The advent of supervisory control raises even more concerns about human performance. Changing the operator's role to that of a predominantly passive monitor carrying out occasional interventions is likely to tax human capabilities in an area where they are known to be quite weak (Bainbridge, 1983; Wickens, 1991). Specific issues include automation complacency, out-of-the-loop familiarity, and loss of situation awareness.

Concern about automation complacency is widespread in aviation where the ability of pilots to quickly detect and correct problems with computer-based navigation systems is essential for aircraft safety (Abbott, 1996; Billings, 1997; Wiener, 1989). As a result, demands for human-centered flight decks abound (Billings, 1997; Parasuraman and Mouloua, 1996). Yet, to date, there are no proven methods a designer can use to specify displays, aids, or warning systems that enable operators to monitor automation effectively and intervene when necessary. To the contrary, research has shown that it is difficult to design automation that an operator will override even when it is necessary (Roth et al., 1987; Smith et al., 1997).

As with the concepts of visual momentum and direct engagement, the set of conceptual normative characteristics dictate that keeping the operator in the loop and cognizant of computer-based operations are important goals (Bainbridge, 1983; Billings, 1997; Sheridan, 1992). There is, however, no consensus as to *how* to achieve these goals. Most

operational designs address the out-of-the-loop issue by periodically requiring the operator to acknowledge the correctness of the computer's proposed solution, despite research data to the contrary (Roth et al., 1987), or have the operator log system state variables at various times (Thurman and Mitchell, 1994). The latter feature is similar in principle to a software-based deadman's switch: it guarantees that the operator is alive, but not necessarily cognizant.

How to achieve a design that supports situation awareness is a recent but pervasive topic (Garland and Endsley, 1995). Pew (1995) notes that "Achieving situation awareness has become a design criterion supplementing more traditional performance measures" (p. 7). He and many other human factors practitioners and cognitive engineers, however, have yet to agree on a definition for situation awareness, measurement methods, or even if the concept of situation awareness is anything more than an abstraction (Garland and Endsley, 1995). If situation awareness is only an abstraction, Billings (1995) concludes that it can be neither measured nor quantified.

Moreover, as supervisory control has become the dominant paradigm, advanced information systems and relatively inexpensive computer technology has made possible a wide variety of aids to help operators, including intelligent displays, computer-based aids/assistants/associates, electronic or intelligent checklists, and error-tolerant systems. These concepts are all appealing. A display or aid that utilizes computer-based knowledge with stable, accessible, and up-to-date information about nominal and off-nominal operations has obvious advantages. Up-to-date information has the potential to enhance greatly operator decision making and problem solving. To date, however, research has not produced prototypes or design guidance that consistently lives up to the expected outcome.

In principle, smart displays can mitigate the SSSI problem and provide display pages that directly support specific operator activities given current system state. Research results are mixed, however. Some designs that sought to enhance performance with direct perception (Kirluk and Kossack, 1993) or direct engagement (Benson et al., 1992; Pawlowski, 1990) found that while such techniques helped during training, the design did not continue to enhance performance once the operator was no longer a novice. Some research has shown that some design methods enhance performance for trained operators (Mitchell and Saisi, 1987; Thurman and Mitchell, 1994), but these methods are in initial stages of testing and are not uniformly known or agreed upon.

As with displays, there are mixed results concerning the effectiveness of specific designs for computer-based aids. When empirically evaluated, some aids failed to enhance performance and in some cases actually degraded it (Kirluk and Kossack, 1993; Knaeuper and Morris, 1984; Resnick et al., 1987; Zinser and Henneman, 1988).

Electronic checklists or procedures are another popular concept. Such checklists or procedures are technically easy to implement and reduce the substantial overhead associated with maintaining up-to-date paper versions of procedures and checklists. The Boeing 777 flight deck includes electronic checklists, and several European nuclear power plants are evaluating them as well (Turinsky et al., 1991). Two empirical studies, however, demonstrate the mixed results that occur when electronic checklists are compared to traditional paper-based versions. In a full-motion flight simulator at NASA Ames Research Center, a study showed that pilots made *more* mistakes with computer-based and smart checklists than with the conventional paper versions (Degani and Wiener, 1993). Another study, in a simulated nuclear power plant control room with certified power plant operators as subjects, also had mixed results. The data showed that during accident scenarios, while computer-based procedures resulted in significantly fewer errors, time to initiate a response was significantly longer than with traditional paper-based procedures (Converse, 1995).

The design of error-tolerant systems is similar. An error-tolerant system is a system in which a computer-based system compensates for human error or helps operators recognize errors and recover before undesirable consequences occur (Morris and Rouse, 1985; Uhrig and Carter, 1993). This notion remains ill-defined. For example, researchers currently do not agree on whether once an error-tolerant system detects a potential error, it notifies the operator, who may take remedial action, or whether the computer-based system assumes control and corrects the problem without human intervention or possibly even notification. Yet error-tolerant systems are necessary to meet new and higher safety goals.

21.4.6 Automation

In the continuum from manual control to full automation, operators are increasingly removed from system control, and in-the-loop familiarity fades. In some systems, control will be fully automatic; anomalies will cause the system to fail safe; and personnel will be notified and eventually repair the automation and mitigate any problems with the controlled system. “Lights out” automation in manufacturing (Shaiken, 1985) and on-going projects in NASA (Brann and Mitchell, 1995) and Navy (Campbell, 1998) are current examples. The design of several recent models of Airbus aircraft is a step in this direction. Some Airbus aircraft such as the Airbus A-320 and A-300, have an electronic envelope that overrides some pilot inputs for given system states (*Aviation Week and Space Technology*, 1998).

There are numerous human performance issues associated with fully automatic systems in which the operator is no longer in the control loop. One issue is whether automation in which the operator is a periodic manager can ever be considered human centered. Another is whether *any* fully automated system can be designed so that an operator can swiftly assume control when the automation fails or engineers and software designers can quickly locate and correct software problems. If so, how should such automation and its interface be designed? What characteristics must the human–system interaction possess to support an operator engaged in fault management rather than control? What characteristics must the human–system interaction possess to enable an operator to assume control? What characteristics must the human–system interaction possess to support engineers and software designers to enable quick and accurate correction of the conditions in the automation software that that caused it to fail?

It is important to note that even with fully automated systems there will always be humans involved with the system. Fault-free software does not exist (Leveson, 1995). Thus, when automation reaches its inevitable limits or system conditions change beyond those for which the software was designed to cope, operators, engineers, and software designers will interact with the system to diagnosis and correct the problems. The design issue is how to specify the human–automation interface(s) so that required personnel can quickly assume system control, carry out fault management activities, and correct the automation software.

21.5 MODEL-BASED DESIGN

This chapter proposes a design methodology to address some of the issues discussed in the previous sections. The methodology, model-based design, proposes that designers *de-*

velop and use a comprehensive model of operator activities to specify *all* human-system interactions. That is, all design decisions about operator roles, workstation design, aids, and so on, should be linked to the descriptions and needs specified in the model. The use of an operator model may be one way of realizing a human-centered design concept. The process of creating such a model is almost certain to ensure that designers understand both broadly and in detail, the users and users' needs. Model-based design may decrease the use of "design by intuition" that occurs all too frequently, in both simple and complex systems.

Figure 21.2 depicts some of the issues model-based design can address. This section, as well as Figure 21.2, is generic in form. The sections that follow propose a set of model characteristics for a model intended to guide design, describe a model-based design methodology, and illustrate how to use the methodology to address the design issues in Figure 21.2. After this generic presentation, the next section describes a specific model-based design methodology, one based on the OFM (Mitchell, 1996a).

21.5.1 Background

Design based on a model of operator monitoring and control activities offers one way of addressing many of the unresolved issues in human-system interaction in complex work domains. Displays, for example, can use this type of model to identify and tailor information needed to support various operator activities, given the current system state. Aids can use such a model to structure a knowledge base, which in turn can advise, remind, or assist operators in contextually relevant ways.

New insights, theories, and models are needed, however, if one of the intended uses of a model is to guide design of human-system interaction in contemporary systems (Miller, 1985). As technology evolves and is introduced, systems change, and operator behaviors change from primarily manual actions to predominantly cognitive and perceptual actions. Models that represent operator activities in contemporary systems must include realistic system features and operator behaviors (Baron, 1984). Historically, human-machine sys-

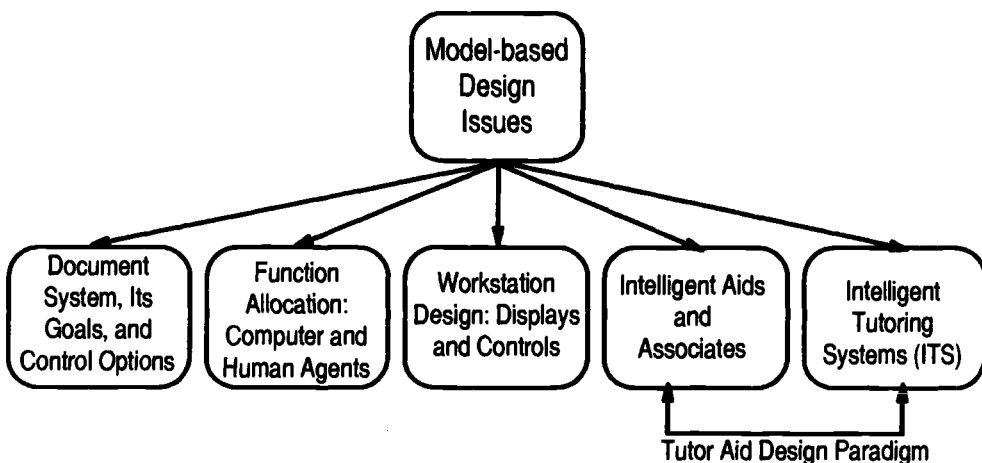


Figure 21.2 Model-based design issues.

tem models often lacked the needed structure to represent fully such systems, their constraints, and the range of operator activities, including discrete manual actions or non-manual actions such as cognitive and perceptual actions (Baron and Levison, 1980; Miller, 1985). Moreover, these models often do not use the increasingly powerful yet inexpensive computing technologies or the knowledge representation techniques developed by the artificial intelligence community over the past 30 years.

Below, a set of characteristics is proposed that an operator model intended to guide design of human–system interaction should possess. They are probably neither sufficient nor comprehensive. The intent is to define a *minimal* set of characteristics. Models for other purposes, for example, cognitive science models whose goal is to represent processes that specify how a human takes input and produces output, are likely to require very different characteristics and structures.

21.5.2 Model Characteristics

Models should reflect the *actual* work domain and make explicit the types of information and range of control activities operators use to interact with the system (Mitchell, 1996b; Morris and Rouse, 1985; Rasmussen, 1988). As Rasmussen cautioned the Human Factors and Ergonomics Society in his 1988 keynote address, context-free models or theories based on research using *simplistic* systems, *unrealistic* tasks, or *naive* subjects are likely to provide little design guidance for real-world applications (Rasmussen, 1988) and may in fact be misleading.

Models of the operator should provide an explicit and detailed representation of operator activities and task constraints. Models should reflect the dynamic nature of the system and environment as perceived by operators given the evolving system state and changing system goals. Representations of operator activities should identify and structure information about important system variables, important collections of states, and define variables that aggregate data or compute measures from low-level data that operators use to assess the current state and to anticipate its evolution. Models should represent *all* operator actions: cognitive and perceptual actions, such as situation assessment and monitoring, as well as the manual actions that traditional task analysis techniques represent (Kirwan and Ainsworth, 1992).

An operator model is likely to be *hierarchical* in form and to represent at least three elements of the overall system: the controlled system, the control system, and the operator or team of operators supervising both. Each level of the hierarchy has important relationships to the levels above and below it. For a given level of abstraction, the current level defines what activities the operator wishes to undertake. The level above indicates *why* the activities need to be performed, while the level below specifies *how* the required activities are to be carried out (Mitchell, 1996a; Rasmussen, 1985).

Operator models should represent the flexibility inherent in human interaction with most systems. As a result, such models are likely to be *heterarchic* at high levels, since most operators juggle several high-level activities simultaneously. At lower levels, flexibility might be represented by a feasible set of activities from which an operator can choose. In this case, the model specifies a set of *acceptable* activities given the current state and higher-level constraints, but does not designate a particular activity from within that set that the operator will choose.

Models of operators in complex systems can take a variety of forms—see Jones and Mitchell (1987) for a more detailed discussion. At one end of the continuum are *conceptual*

models; at the other are *analytic* or *computational* models. Concepts from model theory and philosophy of science help to distinguish among various model types (Mitchell and Miller, 1993). From a model-theoretic perspective, a conceptual model is really a *theory* (Sheridan and Ferrell, 1974); whereas, a model is a physical or formal *structure*, for example, a physical scale model, blueprints, or a mathematical or computational formalism, that implements the concepts contained in the theory (Mitchell and Miller, 1993). For example, a conceptual model of troubleshooting, which posits that an operator first uses symptomatic search, followed by topographic search if symptomatic search fails, is really a theory (Rasmussen, 1986; Rouse, 1978); whereas, computational representations of this problems-solving theory implemented in a work domain are models in the sense previously defined (Fath et al., 1990; Hunt and Rouse, 1984; Knaeuper and Rouse, 1985). For design, the complete spectrum of models from conceptual to computational is important.

21.5.3 A Model-based Design Methodology

Model-based design begins with the development of an operator model. Given a model, each design application and design decision within an application is linked to the model. Essentially the model provides a rationale for *all* design decisions.

21.5.3.1 Step 1. Develop an Operator Model. For the system of interest, develop a detailed operator model. Rouse (1980) suggests that one of the primary advantages of modeling is that the modeling process itself is very beneficial. For complex systems, mathematical or computational models help ensure that the complexity and breadth of the system has been taken into account. As Rouse (1980) notes

(O)nce the model is sufficiently detailed to allow the computer program to produce predictions of performance, then it is not unusual to find predictions to be ridiculous when compared to performance of the real system. . . . Then, the modeling process iterates and testing provides many insights with respect to the system being studied. (p. 2)

The model should possess the characteristics described earlier. It may well include additional features. The following are useful procedures for the construction of an operator model to guide human–system interaction design.

21.5.3.2 Procedures for Model Development

Step 1. Prepare a high-level written description of the system of interest. Specify system goals, exogenous system inputs, major system components, and high-level activities—both those carried out by the automation and those performed by operations personnel. Conduct extensive on-site observations of normal, that is, nominal and off-nominal,¹ operations. In addition, conduct interviews with various personnel and review system

¹Most complex systems define normal operations as nominal—everything goes as planned—or off-nominal—one or more unexpected events occur, but operators are trained to recognize and cope with them. Off-nominal conditions can be thought of as anticipated anomalies. For example, pilots practice procedures for coping with a failed engine: first stabilize the aircraft, then land with one engine.

documentation. Verify the information in the document with various personnel involved in the system, including system designers, management, and, most importantly, operators. Note that for a new system, which has no previous instantiation of any kind, such as a workstation for the International Space Station, the operator model must evolve with the other parts of design. Most systems, however, that involve introducing or upgrading information technology or computer-based control systems have some previous version, even if it significantly different in appearance or operation. The transitions from steam-gauge airplanes and analog nuclear control rooms to glass cockpits and digital nuclear control rooms typify such changes.

- Step 2.* Prepare an initial, high-level draft of the operator model, that is, identify the high-level activities that operators perform. Again, conduct extensive on-site observations of operations. Ask operators to describe *what* they do to attain system goals defined in Step 1. Interview a range of operators.² Review available documentation that describes the system, and most importantly operator functions or procedures. Periodically review the evolving model with operators and other relevant personnel.³
- Step 3.* Define the *heterarchy*. Identify and annotate when and how operators initiate, terminate, or transition among high-level activities. Such annotations relate previous system state and the results of operator actions to the current state or new expectations about operator activities.
- Step 4.* For each high-level activity, specify an associated hierarchy. For each decomposition and at every level of abstraction, define conditions that initiate or terminate activities. For existing systems, the model may represent operator actions as its lowest level. For a proposed design, the development of an operator model forces specification to the action level—a level often overlooked by system designers. Note that actions can be cognitive, perceptual, manual and so on. Avoid “oversequencing” activities. Many groups of activities do *not* have to be performed sequentially. Assuming a deterministic operator or fixed sequences of activities is a serious flaw in many operator models.
- Step 5.* *Validate* the model via direct observation of operators as they carry out control operations under various conditions, nominal and off-nominal. Map operator actions bottom-up into the model. That is, for each observed action, including cognitive or perceptual actions, ensure that the observed action links to the appropriate higher-level activity in the current context. On-line validation while operators carry out normal activities is always preferred. Occasionally, validation may have to be *post hoc*. Then modelers should use, as appropriate, videotapes, extensive notes, display screen snapshots, and event and command logs, to recreate operational contexts. They should then ask one or more operators *what* was done—the actions—and *why*. That is, how the

²Operators, like all other people, bring various perspectives to a problem or set of problems. It is important that the model synthesizes the perspectives of a variety of operators and, when necessary, resolves seeming contradictions.

³In complex systems, many people play important roles, including management, system designers, and operators of similar systems. The goal is to obtain an accurate and balanced model of operator activities. Typically operator verification is the final resolution of differing perspective among various groups involved in the design process. It is not unusual to find that management has vague or even an incorrect understanding about detailed operator activities and procedures (Bradley, 1997).

actions link to currently active activity decompositions or activities that should have been active but were not. The latter case is a model error! Avoid validation that depends on interviews. Interviews provide answers obtained out of context. In complex systems, particularly dynamic systems, context is of the utmost importance. In this process attempt to distinguish between model errors and operator error. An operator activity or action identified in this step, which is *not* represented in the model for the current context, may indicate an error of commission. An activity predicted by the model, which the operator fails to perform or to perform fully, may indicate an error of omission. It is important to note, however, that each of these situations can be due to *model*, as opposed to *operator*, error.

Step 6. *Refine the model iteratively.* Conduct multiple on-site observations of actual operations to periodically validate and refine the model. Review the model with operators, management, and designers. Continue to review new documentation as the system evolves, particularly new operations or operator training manuals. With each change or addition of design information, update the model accordingly and validate it both by on-site observation of actual operations and by asking domain experts to review the “new” model.

21.5.3.3 Step II. Use the Model to Guide Design of Specific Applications. Figure 21.2 depicts various design issues that an operator model can help address. The figure organizes the discussion below, showing how an operator model guides design in various applications.

Issue 1. Document the System, Its Goals, and Control Options. The development of a formal model is one way to ensure and make explicit the most widely touted of all design maxims: “know thy user, and YOU are not thy user” (Lund, 1997). A model of the operator-system interaction helps to ensure that designers have developed a detailed understanding of the overall system, its goals, the *controlled* system, the *control* system, and operator *activities* available to meet system goals. Prepare a short description of the system and its controls, linking goals, limitations, and characteristics to the operator model. This description is typically based on the description prepared in Step I of the model development process. Compare the model-based description with other available system documentation to ensure there are no serious mismatches.

Issue 2. Function Allocation: Human and Computer Agents. The role of operators in complex systems should be as well understood and specified as system hardware and software. It is surprising how often system designers consider operators and their roles at the very end of the design process. Hardware is purchased, software is specified, and only then are the roles of operators considered—and that role is often to fill the gaps between computer-based subsystems (Kantowitz and Sorkin, 1983). Designers normally think about the operator’s role at a very high level: ensure everything works!

One reason for this situation may be the lack of tools to specify the roles of operators. Hardware and software designers have much better tools than do human–system interaction designers, a situation that well-defined operator models may help to remediate. As noted earlier, currently, design guidance is often very general—maxims or rules-of-thumbs. Guidelines typically address *syntactic* HCI issues, which are far less important than *semantic* human–system interaction issues, for example, the content of

displays and the design of high-level operator controls. Furthermore, although there is wide agreement about semantic design principles at high levels, there are essentially no methods to help designers apply them in actual systems.

Increasingly, designers can choose how to allocate functions between human and computer agents. Sophisticated computer technology provides the capability to begin to automate activities previously performed by operators. These are typically activities for which no analytic model currently exists, so expert operators pass on their knowledge to novices by extensive on-the-job training.

Function allocation is not a straightforward decision. Human factors practitioners (Fitts, 1951) suggest that function allocation begin by considering the individual characteristics, strengths, and limitations of both humans and machines/computers, and that then functions should be allocated according to what each does best—a good design heuristic. The oft-repeated acronym MABA/MABA (men are better at/machines are better at) typifies this philosophy. As with most conceptual normative guidance, although this principle sounds good in principle, it is hard to apply and frequently ignored (Chapanis, 1970).

Typically, the limits of technology define operator activities. In many contemporary systems, the prevalent design philosophy dictates that everything that *can be* automated *should be* automated (Kantowitz and Sorkin, 1983). Operators are responsible for performing those activities that remain and, of course, ensuring that *all* activities, human and computer, are carried out effectively in the face of changing or unanticipated system and environmental conditions.

Allocation by default often results in “islands of automation,” with operators in such systems acting as bridges between automated subsystems (Rouse, 1985; Shaiken, 1985). Few studies investigate whether operators perform “bridge” functions well. Studies that have been conducted show that operators frequently perform low-level activities such as tending the computers or manually transferring computer-generated output from one computer to another (Brann et al., 1996; Roth et al., 1987). At best, operators performing such activities are underutilized; at worst, they are error-prone (Brann et al., 1996; Roth et al., 1987). Allocation of function by default may result in humans performing tasks for which they are poorly suited or in operator workload that is so low that humans, who are retained in the system to monitor the automation and compensate when it reaches its inevitable limits, are so removed from the system that situation awareness is not maintained and fault compensation is slow or inadequate (Bainbridge, 1983; Sheridan, 1992; Wickens, 1991).

Dynamic or *collaborative* function allocation is an alternative to static allocation or allocation defined by technology limitations (Chambers and Nagel, 1985; Hammer, 1984; Johannsen, 1992; Rouse, 1994). A human-computer collaborative strategy is one in which workload is balanced between operators and computers in response to changing task demands and system state (Jones and Jasek, 1997). Dynamic task allocation and effective human-computer collaboration potentially offer a means of realizing Fitts’s principle, with allocation tailored by environmental demands, dynamic control requirements, and an operator’s own style (Billings, 1997; Bushman et al., 1993; Jones and Jasek, 1997; Jones and Mitchell, 1995; Rouse, 1981, 1994; Rouse et al., 1990).

Dynamic function allocation, however, requires a detailed, and computationally tractable, representation of operator activities that can guide computer implementation of operator activities and specify the details of how to execute each. Moreover, if an operator model, which the operator understands, guides computer behavior, some of

the problems with human–automation communication and understanding may be reduced. An operator model can provide a common frame of reference.

Issue 3. Workstation Design: Displays and Controls. The impact of new technology is most obvious in recent generations of operator workstations, both displays and controls. Historically, control systems were analog (power plant control rooms, cockpits display, process control systems) and sometimes distributed (e.g., manufacturing systems, refineries) (Zuboff, 1991). With the introduction or extensive upgrade of inexpensive computer-based technology into control rooms, operators often work in centralized areas away from the plant and use sophisticated workstations to monitor and control various computer-based subsystems that make up the overall control system.

As discussed briefly earlier, the design of workstations for operators of complex systems must be considered at two levels: syntactic and semantic. *Syntactic* issues concern human–computer interaction. For displays, syntactic issues include formatting, dialog style, windowing, color, and consistency among windows and display pages. For controls, syntactic issues address the choice of input device(s) such as command line, mouse, joystick, or touch screen. *Semantic* issues concern human–system interaction and address issues concerning display content and control functionality. Designers, however, often confuse, or fail to distinguish between, syntactic and semantic issues, or, worse, they fail to realize that *two* sets of issues exist. As a result, semantic issues are often ignored.

In complex systems, however, semantic issues are critical. Research on the design of interfaces for complex systems repeatedly demonstrates that semantic issues, such as display content, level of abstraction, visual momentum across display pages, control functionality, and wide gulfs of execution or evaluation, dominate the effects of primarily syntactic aspects of the human–computer interface (Mitchell, 1996a; Woods, 1992). Most operators, within some envelope, compensate for and adapt to less-than-perfect design at the syntactic level, if they have at their disposal sufficient information and effective controls at the semantic level (Mitchell and Sundström, 1997; Woods et al., 1994).

The design challenge is to understand and represent system goals, structure, and behavior together with required operator activities in such a way that workstation semantics can be specified. Effective displays and controls should support operator activities for which they are used rather than being a potpourri of what is possible to sense or change. Too frequently display content and control commands reflect system hardware and software, sensed data, and hardware-level commands. For example, one system required operators to encode and input commands in hexadecimal representation.

An operator model can help designers bridge the gaps between high-level operator needs and low-level system data and hardware-oriented control commands. Effective displays should provide the correct information, at the correct time, at an appropriate level of detail. Likewise, operator controls should correspond to high-level operator control intentions rather than low-level, often arcane, commands defined by system hardware.

Displayed information should enable operators to develop and maintain a coherent and accurate representation of the current system. Moreover, such displays should allow operators to anticipate accurately how the system will evolve (Bainbridge, 1983) and to be aware of upcoming changes such as indirect (that is, programmed rather than human-activated) mode transitions (Woods et al., 1994). Displays should mitigate problems such as those Wiener (1989) identifies for many pilots with respect to the highly

automated flight management systems. Pilots often ask: “*What* is it doing now? *Why* is it doing that? *What* will it do next?” An operator model can help identify needed data, guide the abstraction and aggregation of low-level data, and suggest how to link one or more display pages to support each operator’s activities, particularly monitoring, thus, potentially mitigating or reducing the gulf of evaluation (Hutchins et al., 1985).

Controls should permit operators to modify the system quickly and accurately in response to evolving system conditions, failures, or changing system goals. Controls should not require operators to input long sequences of low-level commands. Operator commands should concatenate hardware-level commands into semantically meaningful high-level commands, thus, potentially mitigating or reducing the gulf of execution.

Below is a set of suggested procedures for developing an operator model to help design and validate operator workstations, including specification of the number of monitors that make up the workstation and the sets of operator displays and controls. Model-based design assumes that designers have developed a detailed operator model.

The model construction steps proposed below assume initially that designers have the ability to mock up each proposed display page in software as simple as PowerPoint. Before carrying out the third step (Step c), more sophisticated software is needed to permit dynamic display transitions. Various products such as Visual Studio provide this capability at very little expense. The last step (Step d) requires a simulator, though simulator fidelity need only be sufficiently high to replicate the look and feel and human–computer interaction of the actual operator workstation with the system, thus allowing operators to carry out realistic monitoring and control activities in real time.

21.5.3.4 Procedures for Developing and Validating a Workstation Prototype

Step a. Top-down Assessment of Display and Control Prototypes. For each operator activity contained in the model, identify the set of display pages designed to support it. Perform an analysis to ensure that the data are aggregated and presented in a way that meets operator decision-making needs for that activity without imposing excessive workload. Operators should not be forced to perform computer functions such as data gathering, adding, multiplying, and synthesizing low-level data in order to produce information needed for operator control assessments and decisions. Having a computer do the things it does best, that is, quickly and accurately collect and integrate data, is likely to reduce operator error due to workload mandated by real-time computations. This top-down approach ensures that operators are likely to have the right information, at the right time, and at the right level of detail for routinely executed operator activities.

Next, for each operator command—that is, commands designed to reflect high-level operator control intentions, repeat the preceding process: identify the sequence of interface actions required to carry out the command. Even for “conceptual commands” such as situation assessment, analysis should ensure that required display page requests or data synthesis are reasonable and manual operator input is sufficiently simple and meaningful so that operators can execute it in a manner that is likely to be rapid and error free.

Step b. Bottom-up Assessment of Display and Control Prototypes. Identify the collection of displays that designers expect operators to use frequently for routine monitoring and fault detection. This process is bottom-up: for each display page, designers should provide a rationale, with respect to the operator model, for *why* the page was designed,

why each datum is included on the page, and *why* the format in which the datum is displayed was chosen, for example, level of abstraction and/or aggregation.

Step b is not meant to exclude traditional or hardware-oriented displays. Unanticipated events may require access to low-level system data. Step b is intended to augment the set of displays available to operators and ensure that displays designed to support operator activities in fact do so.

Similarly, identify the collection of operator commands designed to support operator control activities. Again the process is bottom-up: for each command, designers should provide a rationale, with respect to the operator model, for *why* the command was created, *why* the format in which the command is implemented was chosen, for example, checkpoints or required operator input. Commands should be effective and relatively easy to use.

Again note, Step b is not intended to eliminate operator access to low-level commands. The intent is to reduce errors and operator workload by combining into a set of high-level control commands collections of low-level commands that operators frequently use together to achieve specific purposes or system goals.

Step c. Conduct Walk-throughs to Evaluate the Proposed Operator Workstation. This is the stage where the number of monitors composing the workstation should be specified. The number of monitors should be justified by the required activities specified in the operator model: the number of monitors should reflect the tradeoff between display page manipulation and the available space vs. the ability to display all or most of the needed information concurrently for each activity the operators performs.

Jointly, designers and users should evaluate the assignment of displays, or the default assignment of displays, to monitors. A workstation design that uses display real estate—that is, concurrent displays—to support operator activities helps operators maintain situation awareness, and supports visual momentum, thus, reducing the keyhole effect and cognitive workload and enhancing overall system effectiveness. A multimonitor workstation designed along the lines of *system* characteristics, as opposed to day-to-day operator needs, may result if one is not careful to consider workload. With the latter design, operators often have to dig through an increasingly large collection of display pages and synthesize low-level data to obtain required information. Such designs add unnecessary workload.

Next, given a specific workstation prototype, that is, the exact number of monitors and the full sets of displays and controls, conduct an operational walk-through. One or more operators should use the proposed workstation to carry out each activity specified in the operator model. This step begins to integrate the operator workstation with the dynamic system and ensure that the two work together effectively. This exercise, though sometimes labor-intensive, is much more likely to detect problems that increase operator workload and error, or reduce overall system effectiveness, than a traditional workstation design review.

Workstation design reviews are a common technique for obtaining user input. Designers present display pages individually, usually on overhead projectors, and operators are invited to comment or critique the proposed designs. The audiences for such presentations are typically very silent and passive. Thus, designers assume that the proposed workstation is acceptable and, as dictated by good human factors engineering practice, they have obtained user input.

Walk-throughs, on the other hand, actively involve users in the evaluation process.

Designers can observe how operators use the proposed workstation to carry out activities specified in the operator model. Flagrant flaws or omissions will become obvious. Walk-throughs help to ensure that transitions between display pages are few in number and that the overall design supports Woods's (1984) concept of visual momentum, that is, the operators can quickly and effectively obtain needed information and carry out necessary activities. At the conclusion of the walk-through, users and designers, in conjunction with the operator model, should assess the degree to which the overall workstation reduces the gulfs of evaluation and execution.

Demonstrations, walk-throughs, and all other evaluations should include *both* management and users, in addition to system designers. It is important that all the people responsible for the system—designers, managers, operators—share a common understanding of the overall set of operator activities, how each is carried out, and the demands placed on operators during nominal, off-nominal, and unanticipated situations.

Step d. Conduct an empirical evaluation. The design process concludes with a rigorous experiment in which numerous operators use the proposed workstation to carry out a range of routine operator activities, as specified in the operator model. As Karat et al. (1992) found, walk-throughs (Step c) sometimes are not sufficient. Rigorous experiments enable users and designers to detect new, and occasionally very serious, workstation limitations that walk-throughs do not reveal. Experimental data should be evaluated to ensure that the workstation as a whole provides both usability and utility. The data should demonstrate that the proposed design *significantly enhances* operator and system performance (utility).

Issue 4. Intelligent Aids and Associates. In the face of numerous reports of human error in operational settings, the urge to design intelligent systems to guide operator actions is almost irresistible. With hindsight, it is clear that if a current state or consequence of an action was known, the error could have been avoided.

Yet, given decades of research on expert systems, advisory systems, and operator aids, there are few generalizable operational success stories. Empirical studies show that advice-giving systems frequently fail to enhance overall system performance. Sometimes operators do not request proffered advice (Zinser and Henneman, 1988); sometimes operators do not like the "tone" of the advice-giving agent, and thus refuse to follow its advice (Knaeuper and Morris, 1984); and sometimes operators do not think the advice is worth seeking (Resnick et al., 1987). Other aids are effective for novices, but as operators acquire skill, the aid no longer has a discernable effect on performance (Benson et al., 1992; Kirlik et al., 1994).

Despite the lack of demonstrated effectiveness of such systems, advice-giving and warning systems proliferate in operational settings. It is therefore not surprising that operators often question their usefulness. For example, "Bellowing Bertha" on the flight deck commands "Pull-up! Pull-up!" The initial response of pilots may be to silence the alarm first and only subsequently attempt to identify the system state that triggered it. Such behavior was also observed during the accident at Three Mile Island.

Advice-giving and warning systems manifest prototypical limitations across a wide variety of systems. First, such aids rarely integrate multiple limit violations in plausible ways. Warnings, particularly audio warnings, are frequently linked to a single sensor or measurement—an auditory example of SSSI design. In operational settings, a single out-of-limits value for one variable may be acceptable in the context of the values of other variables. Second, control software often lags behind operational requirements.

Factors such as software errors or changes in system dynamics may necessitate a software upgrade that is frequently preceded by long delays. Delayed updates lead to alarms that are obsolete, meaningless, or expected. A visit to almost any control room shows alarms or warnings to which an operator responds “Oh, ignore that. It’s not important” or “That value (displayed in red) is now within limits.”

Design issues for computer-based aids, associates, and assistants are varied and many. Should the aid be stand-alone, that is, compute and recommend a solution? Or should it have the capability to interact with the operator, who can guide and refine its reasoning? How can operators identify when an aid gives bad advice or has reached the limits of its capabilities? How should knowledge and related advice be structured? Can knowledge be monolithic, that is, compiled, or should the aid help the operator understand the *process* by which it arrived at its recommendation? Should the aid recommend a solution that the operator must accept or reject? Or should the aid help the operator evaluate solutions that the operator rather than the computer proposes?

Billings (1997) suggests that the ability to engage in mutual-intent inferencing by collaborative human and computer agents is a necessary prerequisite of human-centered automation. An aid whose knowledge is structured by an operator model can quickly interpret what the operator is doing and why, and notify the operator if discrepancies between model expectations and operator actions are detected. Moreover, representing the knowledge and rationale of an advisory or aiding systems by means of an operator model is likely to facilitate rapid operator understanding of what the aid is doing and why.

For pilot aiding in particular, and operator aiding in general, intent inferencing provides a first step in defining an intelligent computer assistant: task knowledge and a definition of a common frame of discourse (Callantine et al., 1998; Geddes, 1989; Rouse et al., 1987; Rubin et al., 1988). Models of operator activities, goals, and plans have served as the basis for numerous successful strategies to infer operator intent (Callantine et al., 1998; Geddes and Hammer, 1991; Rubin et al., 1988). Moreover, the model, in computational form, is one way to provide the “intelligence” required by an aiding or associate system.

Extending intent inferencing to aiding or collaboration, an operator model can help designers specify different levels of assistance. The model-based system can structure advice or offer assistance at multiple levels of abstraction, dynamically aggregate and abstract data into useful information, and control its timing using a computational implementation of the operator model that represents the operator’s current state as well as system state. Bushman et al. (1993), Funk and Lind (1992), Geddes (1989), Geddes and Hammer (1991), Jones and Mitchell (1995) have all successfully used this paradigm: model-based intent inferencing and an aiding system that uses the model of operator intent as the basis for advising operators or providing assistance.

Issue 5. Training Systems. In contemporary systems, more frequent, formalized, and efficient training to accommodate rapidly changing systems is a necessity. Historically, change in complex systems was slow, and both system and operational knowledge was embedded in organizations by virtue of staff with very low turnover. On-the-job training, with one generation of operators training the next, was common. Today’s rapidly changing systems, however, require more frequent training.

The widespread use of digital technology exacerbates this need. Digital systems are almost always characterized by rapid change; and with rapidly changing technologies

come rapid changes in how systems are controlled and even configured. Each digital upgrade or added function may result in a changed system.

Further increasing training problems, workforces in modern systems change more rapidly than those of previous generations. Workforce mobility is increasing and a system, which regularly had operators with 5, 10, or 20 years of experience, may now have an operational staff that changes completely on an annual basis. Thus, relying upon on-the-job training is no longer sufficient.

As such, training has become an increasingly important and challenging issue. In lieu of an institutional memory whose repository is personnel with extensive and varied operational experience, an organization must explicitly define and document operational procedures, fault management techniques, and make explicit the implicit knowledge that was traditionally informal, heuristic, and passed on to new generations of operators implicitly. Moreover, such information must be communicated to new employees, or to existing employees being trained on new and possibly much more sophisticated systems, in timely and cost-efficient ways.

Computer-based training systems and intelligent tutors, that is, computer-based training systems that tailor instruction to individual student needs, have great potential to partially fulfill growing training needs. Inexpensive computer technology offers a range of tools to support knowledge consolidation (Mitchell, 1997b; Mitchell et al., 1996; Morris and Mitchell, 1997; Ockerman and Mitchell, 1998) that can also be used as the basis of knowledge to construct an operator model and use it to design and implement training systems that in turn teach this knowledge to operations personnel (Chappell et al., 1997; Chu et al., 1995; Govindaraj et al., 1996; Mitchell, 1997a; Vasandani and Govindaraj, 1994).

An intelligent tutoring system (ITS) typically has three components: an expert model, a student model, and a pedagogical component. For systems that train operators to control a dynamic system, a fourth component is often added: a simulated environment to emulate, from the user's point of view, the actual work environment (Fig. 21.3). Chu

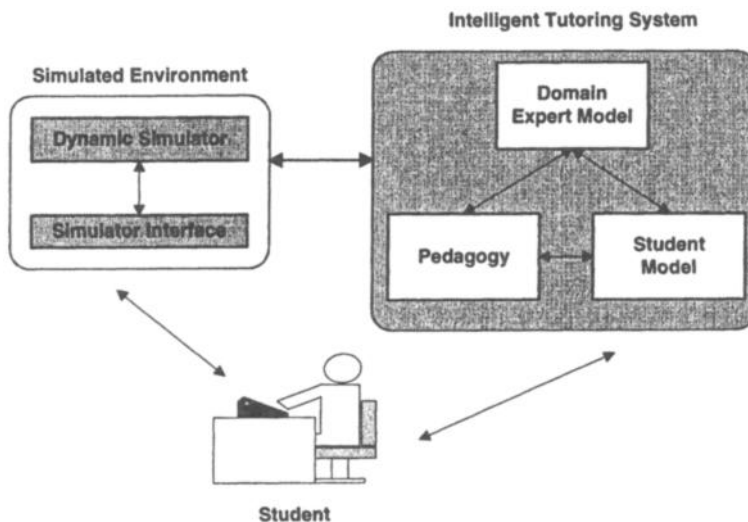


Figure 21.3 An ITS for operators of complex dynamic system.

et al. (1995) suggest that an intelligent tutoring system for operators of a dynamic system consists of three knowledge components: declarative knowledge about the system; procedural knowledge about how to control it; and operational skill that integrates declarative and procedural knowledge into real-time control of the system. The expert model encodes expertise that may include declarative and procedural knowledge as well as contextualized operational skill. The student model represents a student's current level and range of understanding and skill. The pedagogical component structures and controls instruction. When the system includes a simulator, the pedagogical component, in conjunction with expert and student models, use the simulator to construct and control "interesting" scenarios and provide contextually appropriate instruction or coaching. Moreover, for dynamic systems, a simulator, running in conjunction with a computer-based training system, provides a *protected* environment that allows a system to train operators and provides an opportunity for operators to practice operational skills for nonroutine or unsafe events that occur rarely in the actual domain.

The effectiveness of an intelligent tutor depends extensively on the quality of two models: the expert model and the student model. An operator model is a natural representation for the procedural knowledge and operational skill portions of the expert model. A large portion of the expertise embedded in an operator model is precisely the knowledge an intelligent tutoring system attempts to teach the student—the expert model. Student models can take many forms. Overlay models, however, have been widely and successfully used in intelligent tutoring systems (Carr and Goldstein, 1997). An overlay student model has the same structure as the expert model and, over time, the ITS annotates the student model to indicate that the student has acquired another piece of knowledge or skill as represented in the expert model. Comparison of these models indicates the degree to which the student model matches the expert model, that is, the extent to which the student has learned the material encoded in the expert model or areas in which additional instruction is needed. Thus, an ITS using a model of operator activities as the basis of both its expert and student models addresses two important ITS design issues.

Issue 6. The Tutor-aid Paradigm. Figure 21.2 depicts a relationship between training and aiding. Chu et al. (1995) call this concept the tutor-aid paradigm. The expert model in a tutor and the knowledge base embedded in an aid contain essentially the same information. The development of structurally similar models and model-based architectures for both tutors and aids holds out the promise of combining into a single software system the knowledge and resources currently required to build and maintain two separate systems. If successful, such integration may greatly reduce costs while increasing the reliability and sophistication of the resulting training–aiding system. As important from an operator's point of view, a tutor that smoothly evolves to an aid has the potential of becoming a well-understood and trusted assistant.

21.5.4 Conclusions

As the above discussion suggests, operator models for complex systems potentially provide designers with well-defined knowledge for systematically specifying and verifying many components of human interaction with complex systems. Given this background, the first, and probably most important, issue is the selection or construction of a suitable model.

The next section describes such a model—the operator function model (OFM)—and illustrates how OFM-based design is carried out.

21.6 MODEL-BASED DESIGN USING THE OPERATOR FUNCTION MODEL

The operator function model (OFM) was developed at Georgia Tech's Center for Human-Machine Systems Research (Mitchell, 1987, 1996a). The model itself has changed and matured over the years in response to the needs of a range of domains and applications to which it has been applied. It is certainly not the only modeling representation that can be used for model-based design, but it is one that has a demonstrated track record of real-world applications together with empirical data that support the effectiveness of the resulting applications.

The subsections that follow provide a background and description of the OFM and OFMspert, the OFM's computational implementation. The section concludes with a summary of lessons learned using an OFM-based design methodology. Most lessons were encouraging, but some applications were somewhat less than successful; in particular, they illustrated a *gap* between model-based design and implementation.

21.6.1 Background

The operator function model grew out of Miller's discrete control model (1978, 1985). Miller suggests that traditional control-theoretic representations of the human operator in complex systems limit both the types of systems and the types of models that can be constructed. In particular, traditional models can only represent tasks that are little more than simple tracking tasks. As an alternative, he proposes that models of operators in realistic applications consist of cognitive symbol and knowledge-manipulation tasks, with each task characterized by a control context, goal structure, and set of feasible actions.

Miller (1978, 1985) initially proposed the discrete control model as a metatheory, a formal characterization of a system in which the operator fills the roles of adaptive controller, coordinator, and supervisor. This metatheory specifies the types of objects and relations that a theory must contain to extend traditional manual-control models to account for actual operator behaviors in current and future systems. Miller (1985, pp. 180–192) first instantiated his metatheory of discrete control as an abstract system-theoretic structure, specifically a hierarchic/heterarchic network of finite-state systems. Finite-state systems are similar to finite-state automata,⁴ which are defined by a collection of nodes and arcs, which cause transitions from one node to another (Bobrow and Arbib, 1974).

The first application of the discrete control model was to analyze data from a simulated

⁴Like finite-state automata, finite-state systems (nodes) in a discrete control model may be deterministic, stochastic, nondeterministic, or some combination thereof. In a deterministic system, the state transition function maps the current state into a *single* next state. In a stochastic system, the state transition function maps the current state into a set of *feasible* next states, with the set of current-state–next-state pairs described by a *probability function*. In a nondeterministic system, the state transition function maps the current state into a *feasible set* of next states, but does not specify the element from the feasible set.

antiaircraft artillery (AAA) system (Miller, 1979, 1985). The task was to model the behavior of a three-person team controlling the AAA system. The AAA system involved some continuous tasks, such as manual gun tracking. Due to the level of automation, however, lower-level behaviors consisted primarily of *discrete* actions, such as switch settings, and higher-level behaviors consisted of *cognitive* tasks, such as selection of overall strategy, mode choice, and crew coordination.

It is interesting to note that there were at least two other attempts to model the AAA system. Both used or extended Baron et al. (1970a,b) optimal control model (OCM)—a formulation using traditional control-theoretic mathematics to represent both the system and human operator. Phatak and Kessler (1977) modeled only the gunner and the associated gun-tracking task, essentially a compensatory tracking task. Zacharias et al. (1981) modeled a two-person AAA system consisting of a gunner and commander. They modeled the gunner's continuous tracking task with the OCM and used the procedure-oriented crew model (PROCRU) to model remaining crew activities. PROCURU, proposed by Baron (1984) to extend the OCM, models tasks and systems that include discrete actions and more complex tasks than tracking together with structures to represent multiple operators, procedures, and somewhat unconstrained control and monitoring behaviors. Thus, these two models implicitly or explicitly acknowledge the validity of Miller's concerns: exclusive use of control-theoretic representations limits the types of possible engineering theories and the types of tasks that can be modeled with such representations.

Miller (1985) characterizes discrete control models as an attempt to capture in a mathematical or computational form the ways in which team members and operators might decompose a complex system into simpler parts and how they then manage to coordinate their activities and system configuration so acceptable overall performance is achieved. The model attempts to address questions of knowledge representation, information flow, communication, and decision making in complex systems. Miller suggests that the result might be thought of as an operator's internal model of the system plus a control structure that specifies how the model is used to solve the decision problems that make up the control tasks.

At the highest levels, the discrete control model identifies system goals, exogenous inputs to the system, and possible strategies. The goal in the AAA application is to maximize time on target. Exogenous inputs include type of target and target arrival. The commander specifies crew strategy. At the lowest level, the discrete control model of the AAA system accounts for all system outputs; these are typically switch settings that in turn correspond to individual manual crew actions. Levels in between represent the structural coordination of subsystems by upper level systems. A heterarchical control structure shifts the focus of control to the proper subsystem at the proper time (Miller, 1985).

The discrete control model of the AAA system is *descriptive* in nature. It was used to guide analysis of data obtained from teams controlling the AAA simulation. The AAA discrete control model clearly illustrates the decision-making strategies employed by individual teams. Using the model, situations of high confusion, errors, and misunderstandings are readily detectable. Miller (1985) summarizes the results as follows:

As research progressed it became apparent that the issues of problem organization and complexity management on the part of the team members were much more interesting than the problems of quantifying the mode-switching behavior. In this light, the discrete control network, rather than details of individual nodes, provided much insight. Through the *network*

which was developed, it became clear that much of the *complexity* in the problem was *superficial*. Once things were *structured*, it became apparent that the system was controlled by a very small number of key decisions. (p. 195; emphasis added)

Miller proposes a general strategy for developing discrete control models. He notes that there are at least two types of discrete control models: descriptive and prescriptive. This in turn leads to two slightly different modeling strategies. Both begin by identifying system goals and exogenous inputs. If the model, like Miller's discrete control model of the AAA system, is intended to support data analysis, that is, it is a *descriptive* model, available data guide model development and it typically proceeds bottom up. If the intent is to use the model for design, that is, it is a *prescriptive* model, model development proceeds top down, with specification of the higher-level components or activities first. High-level nodes are decomposed until the lowest level nodes represent observable data. Miller notes that prescriptive models tend to be more operator-centered. Both strategies are likely, however, to require iterative refinement. A descriptive model may include top-down analysis; similarly, a prescriptive model may require bottom-up analysis.

Mitchell and Miller (1983, 1986) extended the discrete control model to a discrete control model of operator function for the control of dynamic systems. The purpose was to explore the use of the discrete control model to identify high-level operator control functions and to specify the design of computer-based displays required to carry out the functions. As a proof-of-concept, the discrete control model of operator function was used to identify operator functions in a simulated test-repair routing system. Since Mitchell and Miller's model was intended for design, the model was prescriptive and constructed primarily in a top-down manner. The hierarchic/heterarchic constraint network was used to identify possible high-level operator activities together with needed information, that is, display pages, to carry out each.

21.6.2 The Operator Function Model

The OFM is an *operator*, as opposed to a *system*, description (Mitchell, 1987). The OFM evolved from the discrete control model of operator function, with changes due to the need to represent systems that were much more complex than Miller's or Mitchell's initial applications.

The first use of the OFM was to represent a NASA satellite ground control system. At the lowest level of this system, there were often thousands of possible operator actions, including displays requests, system commands, and both cognitive and perceptual actions. Furthermore, operator actions did not correspond directly to system settings or outputs. As a result, the discrete control model of operator function was modified so that nodes represent operator activities at various levels of abstraction and the system is represented only to the extent that system events or outputs initiate, terminate, or sequence operator activities.

Like the discrete control model, the operator function model is a hierarchic/heterarchic network of finite-state systems. Figure 21.4 depicts a generic operator function model together with some of its essential properties. Table 21.1 lists and defines many OFM features.

Network *nodes* represent operator *activities*. The OFM hierarchy decomposes each high-level operator activity into lower-level activities. Activities are defined at multiple levels of abstraction: from high-level planning sequences and operator functions to low-

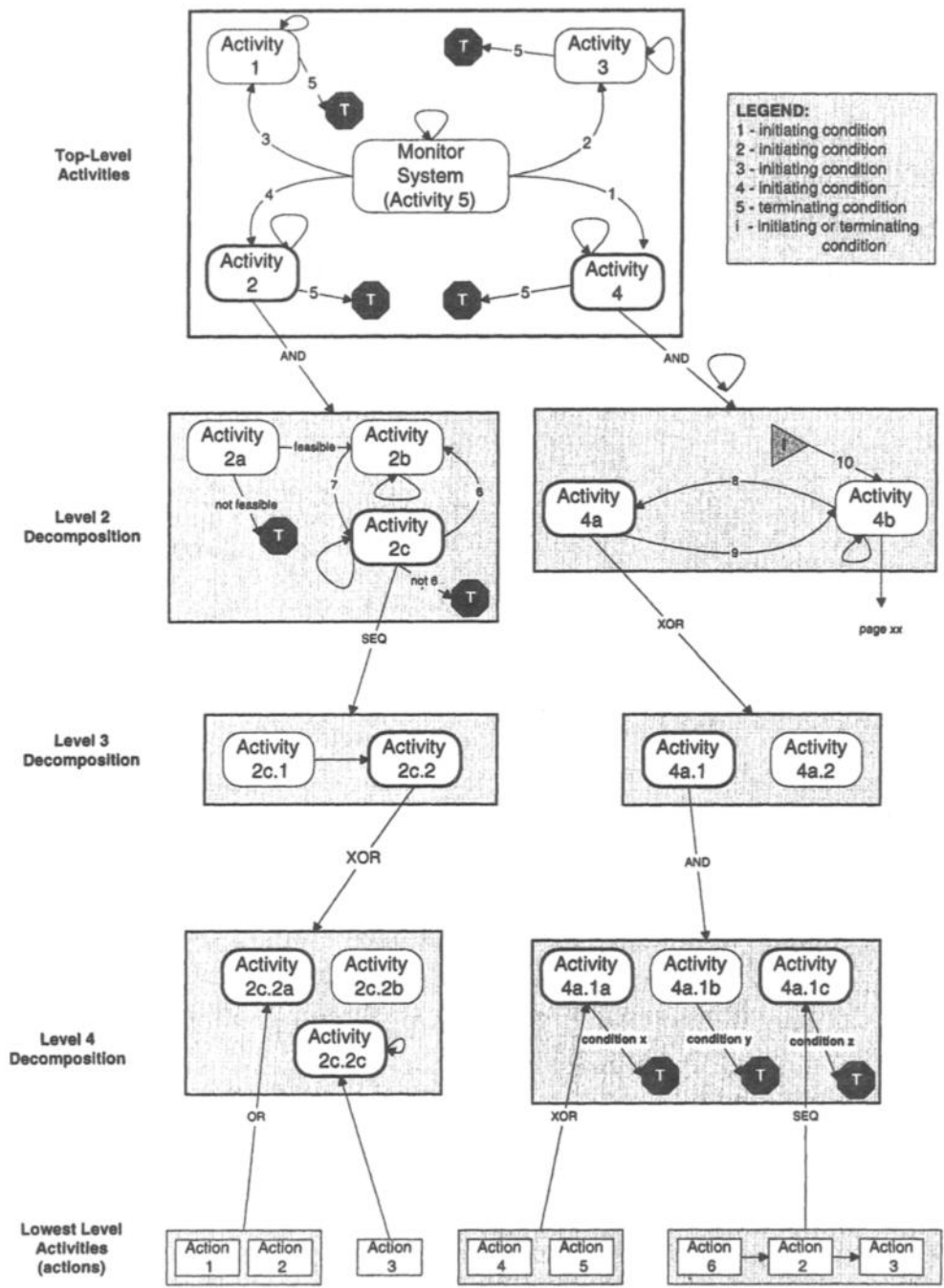


TABLE 21.1 Features that Characterize the Operator Function Model

Feature	Description
Activity/Node	The nodes in the OFM network correspond to activities; as such they are the basic unit of description. An activity occurs at various levels of abstraction; it may be a high-level function or a low-level action
Hierarchy	Activities are decomposed hierarchically. Operators of complex systems organize activities into hierarchies to relate and coordinate them. Operators aggregate low-level activities to high-level activities as a means of reducing cognitive workload. Furthermore, most operator activities are described, documented (manuals and checklists), and taught (training) by means of hierarchies
Heterarchy	The heterarchical structure of the OFM reflects the concurrency of activities for which operators are often responsible. Heterarchical activities are typically defined at the same level of abstraction
Finite-state Systems	In discrete mathematics, a finite-state automation is defined as a finite collection of nodes (states), arcs between nodes, and conditions that cause transitions from one node to another
Initiating and Terminating Conditions	System events, for example, a new air-traffic control clearance, or the results of an operator action, can initiate or terminate activities. They often define the transitions to the finite-state automaton
Nondeterminism	Models may be static (nonchanging), deterministic (for each input there exists exactly one output), stochastic (for each input, there is a set of possible outputs together with a probability distribution that links input–output pairs), or nondeterministic. A nondeterministic model is similar to a stochastic model in that for a given input there is a set of feasible outputs; the difference between a stochastic and nondeterministic model is that the latter does not have a probability distribution associated with input–output pairs
Goals	New Collegiate Dictionary defines a goal as a state to be achieved. All functions, and indeed most activities, are linked either explicitly or implicitly to one or more goals. Pilots may navigate in such a way as to minimize fuel and passenger comfort. Goals are not, however, activities; nor are they represented explicitly in the OFM

Table continued on following page

TABLE 21.1 (Continued)

Feature	Description
"Functions, Subfunction, Task" Vocabulary	Operators often use terms such as function, subfunction, task, or procedure to describe their activities. For example, pilot functions are frequently described as <i>aviate</i> , <i>navigate</i> , and <i>communicate</i> . Functions, subfunctions, tasks, etc., however, are closely linked to goals
Actions: Manual, Cognitive, Perceptual	In the OFM hierarchy, actions are typically the lowest level activity. In addition to manual actions, the OFM also represents cognitive actions, for example, situation assessment, and perceptual actions, ensure that no warning indicators are illuminated
Required Information and Actions	Some interfaces require multiple actions, for example, requests for multiple display pages, to obtain desired information. In some OFM applications, information nodes are added to action nodes
Operator Flexibility	The heterarchic structure, together with nondeterminism, allow the OFM to reflect operator flexibility. Typically, heterarchy represents choice, such as a change in the focus of control, for high-level activities. Nondeterminism depicts choice at the same level abstraction and is one type of state transitions in which operators can choose from a set of feasible next actions, given the current state
Operator Error	The OFM models operator error by predicting operator activities starting at the highest level activities down to the lowest level actions. As operators perform actions, the OFM attempts to link new actions to one or more active activity trees for which the new action is included in the feasible set of next actions. A potential error of commission occurs if an action cannot be linked to <i>any</i> active activity tree, that is, the action is not contained in any of the feasible set of next actions. A potential error of omission can also be identified when all required lower-level activities in the decomposition are not performed.

level actions. In between, collections of nodes can represent procedures, checklists, tasks, or other sets of activities typically carried out together. The *heterarchy* represents activities that the operator may perform concurrently or shifts in the operator's focus of control from one high-level activity to another. Arcs between nodes depict conditions that may initiate, terminate, or sequence activities. At high levels, initiating and terminating conditions add or delete an activity to the current set of activities for which the operator is responsible. For example, in Figure 21.4, arc 1, in the top-level representation, initiates Activity 4, adding it to the set of high-level activities for which the operator is currently responsible. A terminating condition indicates that some activity, or a portion of some activity, is completed and actions to support that activity are no longer expected. In this example, arc

5 terminates Activity 4. In many dynamic systems, operators engage in multiple, concurrent activities, with an overall monitoring and control activity that is always active. For example, in Figure 21.4 Activity 5, at the highest level of specification, is expected to be active at all times. Some arcs, particularly at the lower levels, merely sequence activities. Since OFM nodes are *finite-state systems*, a node can be in one of a finite number of states. State transitions are typically *deterministic* or *nondeterministic*. An OFM used for design rarely has enough data to specify a stochastic transition function.

Both the discrete control model and the OFM *implicitly* acknowledge that operator activities are undertaken to achieve one or more goals. Neither model, however, *explicitly* represents goals. Goals in most control systems are implicit—states to be achieved. Thus the goal of each activity, particularly a high-level activity, is to carry out the activity in a timely and correct manner. Moreover, the OFM deliberately uses terms such as “functions, subfunctions, tasks” and more generally, “activity,” rather than “goals, operations, and so on,” that the GOMS model and methodology use (Card et al., 1983). The reason is simple: the function–subfunction–task vocabulary is the vocabulary that operators themselves use to describe and document how they view and control the system.

In the OFM, actions can take forms other than manual. In particular, various OFM applications include *cognitive* actions, such as situation assessment, or *perceptual* actions, such as scanning for alarms. Some applications of the OFM have linked information requests to actions. This is sometimes necessary when multiple display pages are needed to carry out an action. More frequently with modern graphical user interfaces, multiple actions are alternative *syntactic* actions for carrying out the same *semantic* action. As a simple example, in Microsoft Word 6, users can save their current file (a semantic action) in one of several ways: (1) select the File menu, then the Save option; (2) press ctrl-S on a (PC) keyboard; or (3) click the diskette icon on the tool bar.

Flexibility in the OFM is represented by two features: heterarchy and nondeterminism. Operators are rarely deterministic, though procedure manuals often suggest that they are. Operators can, and do, make choices, selecting which of the set of active high-level activities to pursue next. At higher levels, heterarchy represents the choices operators have, within some time frame, to alternate among control activities. For lower-level activities, nondeterminism represents choices operators have about which activity from a feasible set of activities to perform next. Moreover, high-level activities are not necessarily performed to completion; rather operators interleave actions to meet the requirements of the set of high-level activities for which they are currently responsible. For example, in considering a series of operator actions, the first two actions may support Activity A, the next action Activity B, and the fourth action Activity A, which in turn terminates Activity A. Explicit representation of acceptable actions via heterarchy and nondeterminism allows the OFM to identify potentially erroneous actions. If the operator performs an action that does not support any currently active activity or if an action is not contained in any feasible set of actions, the action may be an error.

Figure 21.5 depicts a number of heterarchic and nondeterministic relationships that the OFM uses. It is not comprehensive, but illustrates major relationships together with the most recent OFM modeling syntax. Heterarchy (a) depicts an (AND) relationship in which an activity decomposes into a set of activities, all of which must be carried out concurrently. Heterarchy (b) depicts a set of activities that must be carried out concurrently only if the associated initiating condition occurs. If the initiating condition does not occur, the operator is only responsible for Activity A. Moreover, since Activity B also has a terminating condition, part of the time there may be two concurrent activities, A and B, and part of

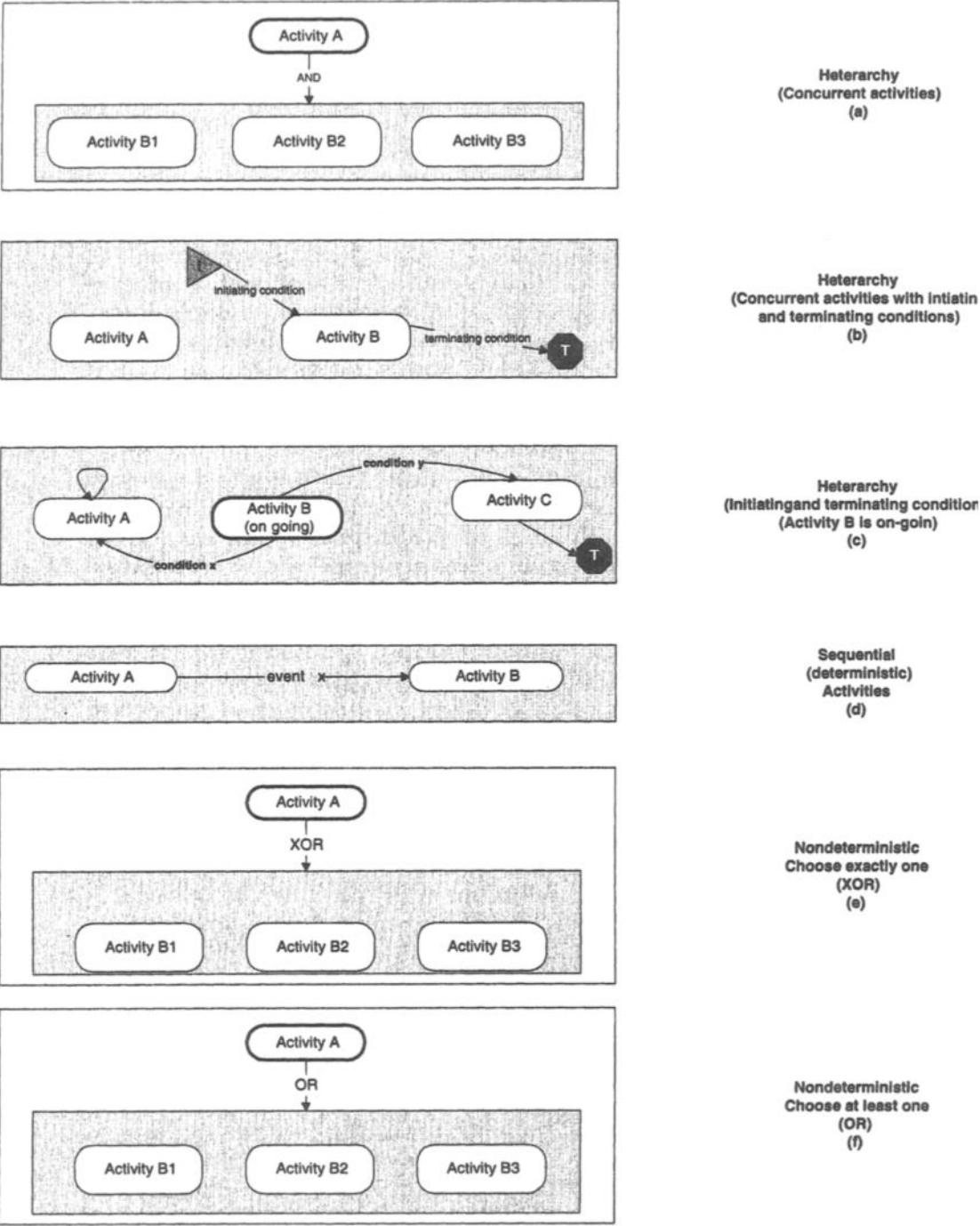


Figure 21.5 Various types of heterarchic and nondeterministic OFM relationships.

the time only one activity, Activity A. Heterarchy (c) depicts a high-level heterarchy that is common in control rooms in which operators are responsible for extensive monitoring as well as other activities. Activity B is on-going, that is, the operator is always responsible for Activity B, whereas the operator is responsible for other activities only when the associated initiating condition occurs. As with heterarchy (b), when a terminating condition occurs, the associated activity is no longer active. The relationship depicted in Figure 21.5d is the standard deterministic state transition function typical of most finite-state automata. The (XOR) nondeterministic relationship depicted in Figure 21.5e is one in which the operator must choose exactly one activity from the feasible set of activities: B1, B2, and B3; this activity might be a mode choice. The (OR) nondeterministic relationship depicted in Figure 21.5f is one in which the operator may choose one or more activities from the feasible set of activities: B1, B2, and B3. Relationship (f) could represent the “save file” example. Typically, a user would select one of the three activities; it would not be an error—or cause harm—if two or more of the actions were performed.

21.6.3 OFMspert: The Operator Function Model Expert System

The operator function model expert system (OFMspert) is a computational implementation of the OFM. The original intent was to use the OFM to predict operator activities, and as an operator performed actions in real time, attempt to interpret those actions by linking them to the upper level activities they were intended to support. As such, it could be thought of as a “living task analysis” (Palmer, 1987).

OFMspert is a form of intent inferencing (Rubin et al., 1988) or activity tracking (Callantine et al., 1998). Originating in the days when great things were expected from artificial intelligence, the theory underlying intent inferencing was that understanding takes place when actions are identified as part of a sequence of actions undertaken to achieve one or more goals (Card et al., 1983; Schank and Abelson, 1977). Callantine et al. (1998) are a bit more restrained. They define activity tracking as a computer capability that seeks answers to the following questions: *What* is the operator doing? *Why* is the operator doing that? *What* will the operator do next? Answers to such questions are possible because an OFM represents an operator acting in a task domain where goals are shared and personnel are trained. Readers might note that the questions just posed are the complement of Wiener’s (1989) characterization of questions that pilots often have about automation. Moreover, a human–machine system’s ability to respond accurately to both sets of questions is one way to implement Billings’ (1997) principle of mutual intent inferencing between the operator and the automation in a human-centered system design.

21.6.3.1 ACTIN: OFMspert’s Actions Interpreter. OFMspert uses a blackboard model of problem solving (Nii, 1986b, c, 1994) to construct expectations about current operator activities and to interpret in real-time actions that the operator performs. The blackboard model consists of several components. The blackboard data structure is a global database where a set of partial and full solutions, known as the *solution space*, are kept. The blackboard data structure is both heterarchic and hierarchic. Heterarchies represent alternative or concurrent solutions. The hierarchies are decompositions that correspond to various levels of abstraction in the problem analysis. *Knowledge sources* (KS) are functions that link or transform information on one level of the blackboard into information on another level. As knowledge sources add, delete, and modify blackboard entities, the solution space changes and at any given time it provides the current best hypothesis about the problem.

The original HASP blackboard was designed to interpret sonar signals in real time in order to characterize both the number and type of ships in the area (Nii, 1986a). Thus, the blackboard model of problem solving possesses many of the properties required to implement the OFM in computational form.

Figure 21.6a depicts ACTIN (actions interpreter), OFMspert's blackboard model of problem solving. Implementing a computational form of the operator function model as a blackboard seemed a natural way to represent the OFMspert's most salient properties: representation of dynamically changing state and environment data, hierarchic and heterarchic levels of the OFM, and interpretation of input data (i.e., operator actions) in real time. The heterarchical hypotheses that the HASP blackboard contained matched well the OFM requirement to represent concurrent operator activities. Since the blackboard data structure is hierarchical in form, it allows OFMspert to represent hierarchically decomposed expected activities as well. Operator actions are represented as the lowest level nodes on ACTIN's blackboard. Knowledge sources implement ACTIN's intent inferencing functions, attempting to interpret detected actions as supporting one or more activities posted on the blackboard.

Modeled after the HASP blackboard model of problem solving (Nii, 1986a, b, c), ACTIN contains a blackboard data structure, knowledge sources, and a set of lists to coordinate activities. ACTIN's blackboard data structure is the current best hypothesis about what an operator or team is doing and why. The ACTIN blackboard data structure (Fig. 21.6b) contains activity trees, each a hierarchy of nodes that represent high-level operator activities and their decompositions as specified in the OFM. Operator actions are typically the lowest level(s) of the blackboard, though recent applications depict both *expected* and *detected* actions as the two lowest levels, respectively. Activity trees posted on the ACTIN blackboard are usually *model-derived*, specified by a combination of the operator function model and current system state. Thus, some event, perhaps a change in system state or an operator action, triggers an ACTIN cycle that posts one or more activity trees on the ACTIN blackboard data structure.

Actions that the operator performs are always *data-derived*. Knowledge sources attempt to link each action to all possible upper level activities that the action may support. Rubin et al. (1988) call this property *maximal connectivity*. The assumption is that it is better to overinfer than to guess and be wrong. As shown in Figure 21.6b, sometimes a detected action is posted on the blackboard but has no links. That is, given the current state of OFMspert, knowledge sources are unable to interpret the action at the present time as supporting one or more activities posted on the blackboard. Subsequent processing may allow ACTIN to disambiguate multiple-linked actions or provide links for an action that has none. Occasionally, there are data-derived activity trees. ACTIN may examine a sequence of actions that was not previously understood and infer a purpose. Subsequently this data-derived, or inferred, tree is posted on the blackboard.

Like HASP, ACTIN has three lists (Fig. 21.6a). The future-events list is a time-ordered list where blackboard activities such as periodic assessments or efforts to disambiguate or interpret actions are scheduled for future execution. Processing events on the future-events lists may add events to the events list. The events list contains activities that need to be executed during the current blackboard cycle, that is, before the blackboard processes any new input. The problems list is where anomalies, such as a failed attempt to interpret a detected action, are stored. Like HASP, during every blackboard cycle, ACTIN processes all the lists in the following order: future events, events, and problems.

ACTIN knowledge sources do many things: post activity trees and actions, attempt to

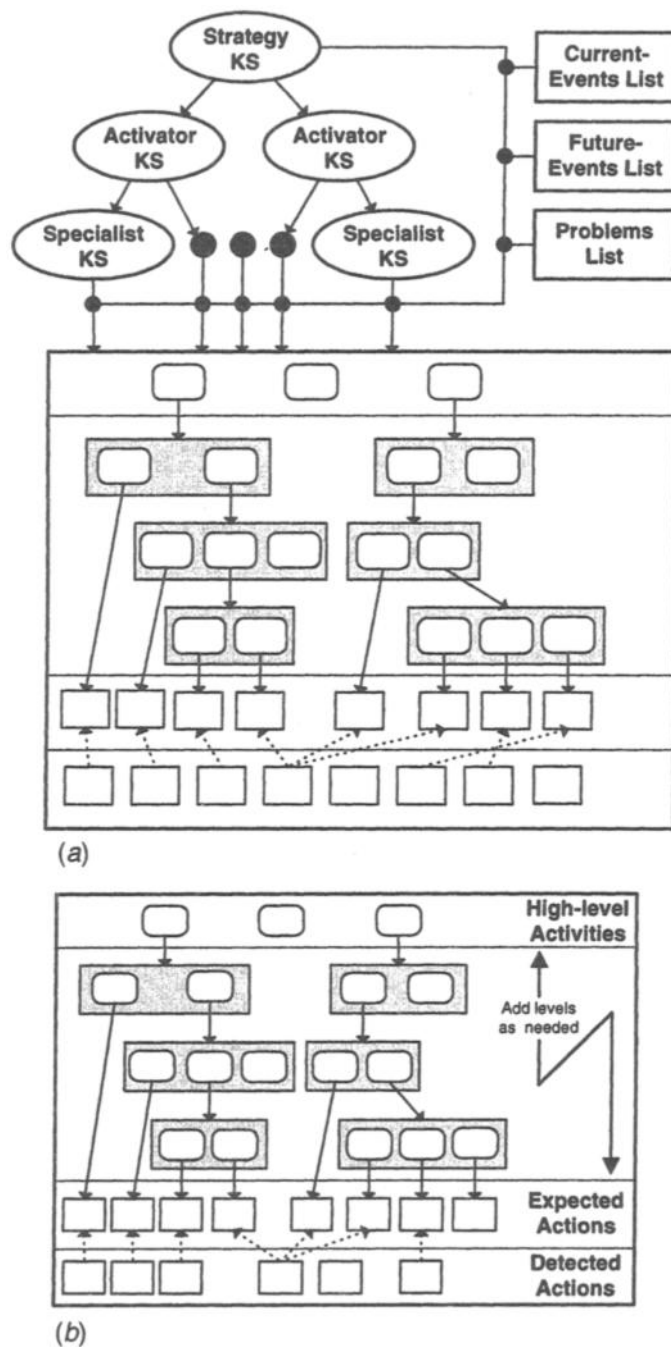


Figure 21.6 (a) ACTIN: OFMspert's blackboard. (b) ACTIN's blackboard data structure.

link an action to one or more activity trees that it might support, terminate an activity and all of its children, and assess the extent to which an activity has been correctly carried out. Other knowledge sources attempt to disambiguate actions with multiple links. Assessment knowledge sources are often used after a terminating event to assess the degree to which the operator carried out expected activities. Moreover, if OFMspert is used as the basis of an aid or tutor, assessment knowledge sources may check intermittently the progress of the various activities and possibly remind the operator or student that some activities remain incomplete.

The original HASP blackboard had a hierarchy of knowledge sources: strategy, activator, and specialist (Figure 21.6a). In the OFMspert application, *strategy* knowledge sources process information from the various lists, that is, future-events, current events, and problems lists. After determining the list from which the next event will come, the corresponding strategy knowledge sources triggers one or more *activator* knowledge sources to manage various types of events or blackboard activities. For example, if a new action arrives, the strategy knowledge source for the events list activates the action-management activator knowledge source. In turn, this knowledge source initiates at least two specialist knowledge sources: post action and interpret action. The first knowledge source posts the new action on the lowest level of the blackboard; the second attempts to link the action to any activity tree currently on the blackboard that the action can support. If the interpret-action knowledge source is unable to link the action, it will place a copy of the action together with a snapshot of the current system context on the problems list and schedule on the future-events list a subsequent attempt to interpret the action. Future operator actions may help other specialist knowledge sources infer a purpose for a set of actions that was not previously understood.

21.6.3.2 OFMspert Architecture. Figure 21.7 shows the complete OFMspert architecture. In addition to ACTIN, OFMspert has two other major components: the current problem space and the operations model. The current problem space stores information about the system, including state variables, derived variables, and a semantic description of the operator workstation. State variables contain sensor-level information about the system, while derived variables represent operator computations in which they collect and aggregate lower-level data into meaningful information or make a qualitative assessment of one or more state variables. Derived variables are usually identified during the development of the OFM and often have Boolean values. For example, in satellite ground control, a derived value may be data quality: data quality is good if 90 percent or more of the expected data are recorded. In aviation, a derived variable might be “heading out of limits.” This Boolean variable is true if the cleared/desired heading differs from the actual heading by more than several degrees. The representation of the operator workstation, that is, the display pages and information that the operator has in view while performing some activity, is also represented in the current problem space. The representation is defined at a semantic level—a description of the information or data currently available to the operator without performing any additional display page or information requests.

The operations model, initially called the enhanced normative model (Rubin et al., 1988), contains static information about the system. Its primary component is the OFM activity model. The operations model may also contain operational heuristics. Operational heuristics are rules of thumb or background knowledge, which are not represented in the operator function model. Satellite controllers, for example, know that a “Wallops pass” requires a certain network configuration. Similarly, pilots have expectations about aircraft

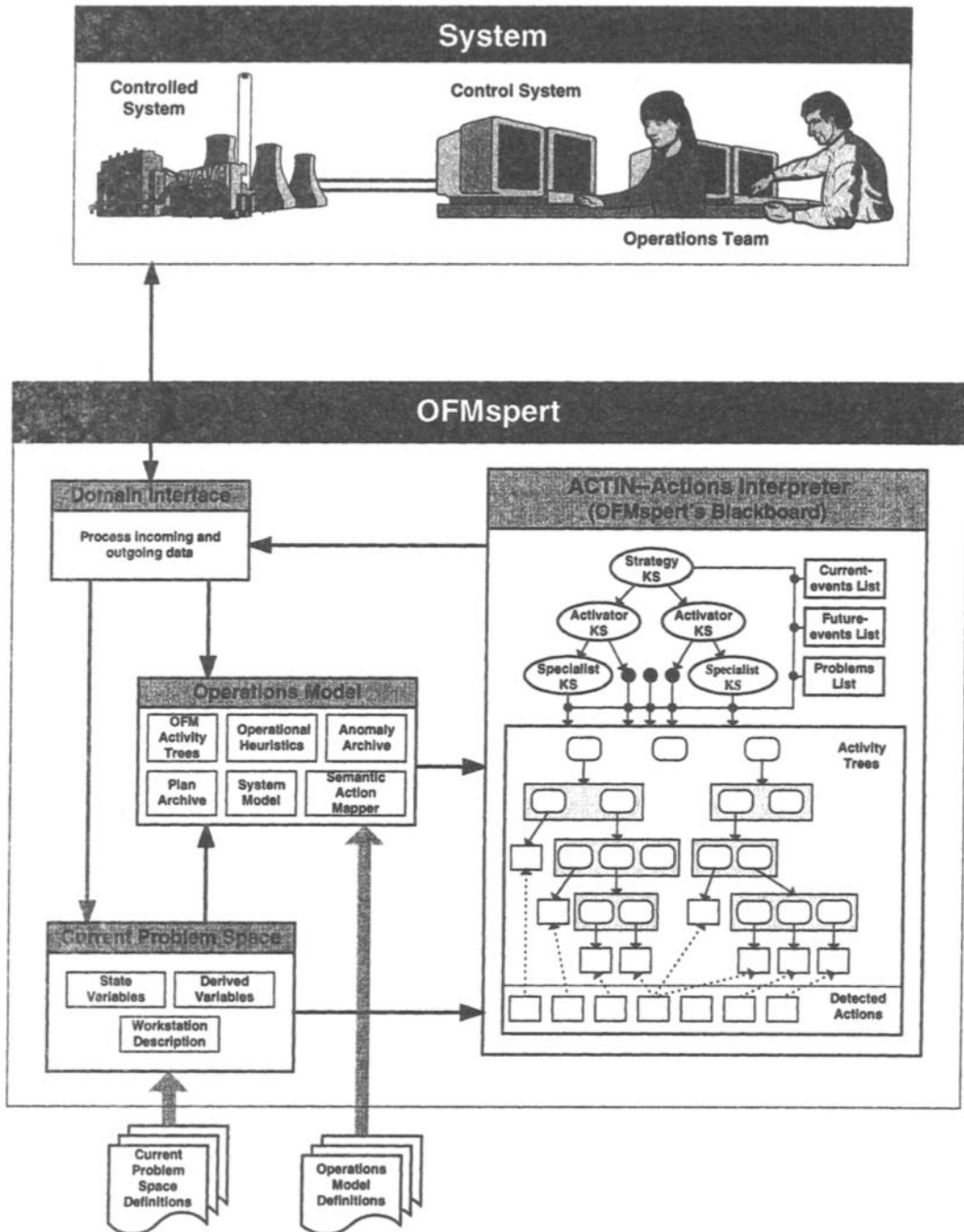


Figure 21.7 OFMspert architecture.

performance based on aircraft type. The semantic action mapper in the operations model is the structure that converts one or more syntactic actions into a single semantic action. Particularly in workstations using modern interface technology, operators often have multiple ways of accomplishing the same objective. The operations model may include an anomaly archive to support on-line troubleshooting and fault management of previously seen and solved problems. Another component of the operations model may be a plan archive. This is a domain-specific database that stores prototypical plans. For example, a plan archive might contain pass-plan templates for satellite control or initial flight plans for a specific company, aircraft, and route. Finally, the operations model may contain a state or system model; such a representation may be important for OFMspert applications such as tutors or aids in which the “intelligent” system explains its recommendations to an operator or contextualizes information for a student. Rubin et al. (1988) provide a detailed description, at the implementation level, of the components that made up the initial OFMspert.

21.6.4 Constructing an Operator Function Model

The process of constructing an OFM, like most modeling and design activities, is characterized as a process of initial conceptualization, iterative evaluation and refinement, and final specification (Jones et al., 1995). Generally, the steps described in the generic methodology for model construction and validation apply equally well to the construction of an OFM.⁵ The generic steps are summarized as follows:

1. Prepare a high-level *written* description of the system of interest, specifying system goals, exogenous system inputs, major system components, and high-level activities—both those carried out by the automation and those performed by operations personnel
2. Prepare an initial, high-level draft of the operator model—that is, identify the high-level activities that operators perform
3. Define the *heterarchy*. Identify and annotate when and how operators initiate, terminate, or transition among high-level activities
4. For each high-level activity, specify an associated *hierarchy*. For each decomposition and at every level of abstraction, define *conditions* that initiate or terminate activities
5. *Validate* the model via direct observation. Map operator actions bottom up into the model, including cognitive and perceptual actions, ensure that the observed action links to the appropriate higher-level activity in the current context. Identify each discrepancy between the model and observed activities; resolve whether it is due to a model or operator error
6. *Refine* the model iteratively as the proposed system evolves

Exceptions or additions include the following. In Step 2, make extensive use of the *visual form* of the operator function model (e.g., Fig. 21.4). The pictorial form allows

⁵*Caveat:* All these steps assume extensive on-site observation of operations and periodic interaction with the various groups involved: managers, designers, and operators. Operators are always the final arbiters if there is some disagreement in how operations are carried out.

domain experts to easily and iteratively inspect, correct, and refine the evolving model. In Step 3, construction of an OFM hierarchy, make use of OFM hierarchic properties such as those displayed in Figures 21.4 and 21.5 and described in Table 21.1. Step 4 specifies the hierarchic decomposition for each high-level activity. This involves defining the relationship among a set of activities resulting from a decomposition. It is important to define conditions that initiate (make active) and terminate (make inactive) activities defined at the same level of abstraction. Since mathematically the OFM is a hierarchic/heterarchic network of finite-state systems, it is important to specify the state transition functions. Identify the conditions that *initiate* or *terminate* a node, define a *sequence* between nodes, and *enable* transitions among the nodes that decompose the activity above them. As shown in Figure 21.5, activities may decompose into subactivities where the state transition is nondeterministic, but mutually exclusive (XOR), that is, a feasible set of activities from which exactly one activity can be selected. Others may have a nondeterministic next state that consists of a feasible set of activities where one or more (OR) activities may be chosen. Some sets of activities must all be done (AND), but the activities may be carried out in any order. The number of levels in the OFM hierarchy depends entirely on the application of interest. Although the original OFM application had four levels of activities (e.g., functions, subfunctions, tasks, and actions), subsequent OFMs have had many different numbers of levels, including phase, subphase, subtask, mode, and procedure. Recall that actions can be cognitive, perceptual, and manual. In the current OFMspert, the two lowest levels of decomposition consist of *expected* actions (model generated) and *detected* actions (operator performed). Step 5, model validation, is critical. The visual form, the use of operator language, and level of detail in the model facilitates verification and validation. Direct observation of operations or *post hoc* analysis of videotaped operations is essential. Validation includes bottom-up mapping of observed operator actions to OFM activities that are currently active. This is straightforward, as the OFM decomposition supports this type of validation. The last step, iterative refinement, is as important as initial model definition and validation. In work domains, particularly those that are changing due to the introduction of new computer-based or information technology, systems change quickly during design and implementation. As a result, particularly at lower levels, the model changes. The OFM drawings should be updated to reflect all changes and the visual form used to validate the modeler's new understanding by domain experts. Direct observation, however, remains the most reliable verification and validation method.

The generic model-based design methodology ends with Step 6, iterative refinement. An OFM-based design methodology may also include development of OFMspert, the computational form of the operator function model. Some types of intelligent displays, aids, and tutors need a software implementation of the objects and relations specified by the operator function model in order to define the intelligence of the artifact. Implementing OFMspert, given a well-defined operator function model, is conceptually straightforward, though potentially, like all large software development efforts, quite time-consuming.

21.7 OPERATOR-FUNCTION-MODEL-BASED DESIGN: ILLUSTRATIVE APPLICATIONS

As described in the generic section on model-based design, there are numerous applications for which a model-based design methodology proves quite effective. The following sec-

tions illustrate a number of such applications. Each was conceived, implemented, and evaluated by members of Georgia Tech's Center for Human-Machine Systems Research.

Although the operator function model and OFMspert have been used in many applications, (e.g., aviation, air-traffic control, manufacturing), the applications discussed in this section are all in the context of NASA Goddard's Satellite Ground Control System. These examples are not necessarily the newest or most sophisticated, but they illustrate OFM-based design without requiring the reader to understand in detail numerous complex domains. References to applications in other domains are also provided.

21.7.1 Operator-function-model-based Design Methodology

Phase I. Develop an Operator Function Model

Prepare a High-level Written Description of the Domain of Interest (Step 1). The main function of NASA Goddard Space Flight Center (GSFC) is to design, launch, and control near-earth satellites. These satellites are unmanned spacecraft that gather science data about the weather, atmosphere, sun, and the Earth itself. The satellites periodically transmit data to an earth ground station that in turn forwards the data in real time to NASA Goddard control rooms. During these short periods of time, called *passes*—as they occur as the satellite passes over a ground station—spacecraft-specific controllers command the spacecraft, initiate transmission of science data to the ground, and ensure that the current state of the spacecraft is within nominal limits. The goal is to collect as much good-quality data as possible.

The Multisatellite Operations Control Center (MSOCC) is a specific NASA Goddard system that coordinates the use and monitors the effectiveness of communication and computer systems shared by a number of satellites. Since contact with a given satellite is brief, satellite-specific controllers share this equipment. The MSOCC system is responsible for scheduling, configuring, and deconfiguring the shared equipment. During real-time contacts, MSOCC operators are responsible for ensuring the continuous functioning of computers and communications networks as well as the integrity of the data flowing through them. Exogenous inputs to the MSOCC system are satellite pass and equipment schedules and the status of MSOCC hardware.

MSOCC is a good candidate for increased use of automation. The current system is labor-intensive—four operators per shift, three shifts a day, seven days a week. Most operator functions are straightforward and many manual activities can be automated by means of simple rules. When this research began in 1984, NASA was in the process of upgrading the system with state-of-the-art automation (Computer Sciences Corporation, 1983).

The most significant information obtained during Step 1 was that the schedule for the new MSOCC system would always be at least 12 hours old. This was due to constraints of other systems with which MSOCC needed to interact and to budget limitations for the upgrade to the MSOCC system itself. All scheduled configurations and deconfigurations are carried out automatically. Operators, however, had to perform manually any activity that was not included in the predefined MSOCC schedule.

Prepare an Initial High-level Draft of the OFM (Step 2). Despite the fact that three operators would staff the system 24 hours a day, seven days a week, management and

system designers initially said that operators would not have *any* functions except monitoring the automation, as the system would be highly automated. Demonstrating the importance of developing a well-defined model that reflects multiple perspectives, it turned out that even with the addition of automation, MSOCC operators would have at least five major functions (top-level activities of Fig. 21.8). MSOCC operators are expected to monitor continuously any current spacecraft-to-ground transmission and, when necessary, engage in fault detection and management whenever the automation reaches its limits, which for this system is any time a change to a 12-hour-old schedule is made.

Define MSOCC's Heterarchy for the Top-level Activities (Step 3). Determine the relationships among the top-level, heterarchic activities, on-going or ones with initiating and terminating conditions. As Figure 21.8 shows, there is one on-going activity:⁶ monitoring currently active spacecraft. The other activities all have both initiating and terminating conditions. Recall, at high levels, heterarchy and the combination of initiating and terminating conditions model operator flexibility and refocus of attention. Initiating or terminating conditions typically add or delete high-level activities to the set of activities for which the operator is currently responsible.

Decompose Each High-level Activity and Define Conditions that Initiate, Terminate, or Sequence Activities at the Same Level of Abstraction (Step 4). The remaining portion of Figure 21.8 shows the decomposition of one high-level activity, "configure to meet support requests." The first level of decomposition is a sequence of activities (SEQ). If activity one or three in this sequence is not feasible, then the upper level activity terminates, together with all its subactivities, and the operator reports that the request cannot be met.

The lower portions of Figure 21.8 show the remaining decomposition for activities associated with "configure to meet support requests" down to the actions level. The activity terminates (normally) if the operator is able to configure the requested network; otherwise, the operator informs controllers that the addition of the requested pass violates a system constraint or that there is insufficient hardware at the time requested.

Validate the Model via Direction Observations of Operations and Refine the Model Iteratively over Time (Steps 5 and 6). Extensive on-site observation of operations allowed modelers to link observed actions to upper level activities that they were hypothesized to support, and both validate or update the model. Figure 21.8 and the remaining seven pages that made up the MSOCC OFM were intermittently validated over a period of 12 to 18 months as MSOCC evolved; see Mitchell (1987) for the complete MSOCC OFM. The visual form of the OFM greatly facilitated this process.

Develop an MSOCC OFMspert (Step 7). Although development of a computer implementation of the model is not a required step in the proposed methodology, several of types of applications require a computer implementation of the model. OFMspert is one such computer implementation and it formally guides the transition from the knowledge and structure contained in the OFM to a rigorous software specification. OFMspert for MSOCC served both as a proof-of-concept for the OFMspert architecture and as the

⁶MSOCC operators are *always* responsible for monitoring any network, from control room to spacecraft, that is currently transmitting data. Other activities are always performed in addition to the "control of current missions" activity.

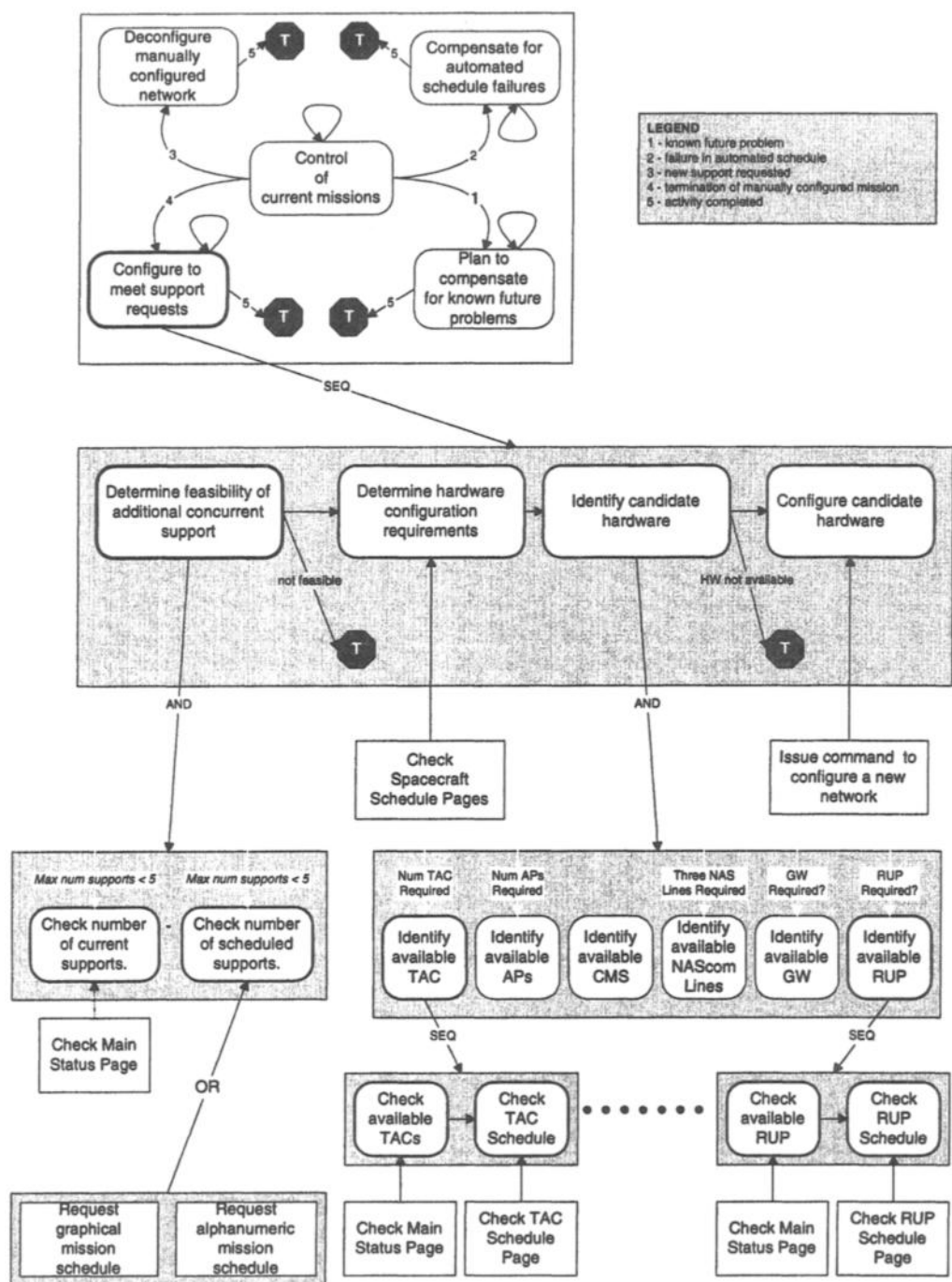


Figure 21.8 Representation of MSOCC top-level activities and decomposition of "configure to meet support requests."

basis of several studies to ensure that it performed adequately and could be used as the intelligence for computer-based artifacts such as assistants and tutors.

Phase II. Use the Model to Guide the Design of Specific Applications

The MSOCC operator function model and its computational implementation—OFMs-pert—have served as the basis of a variety of applications. Examples for each design issue depicted in Figure 21.2 are given. References to more detailed publications for the applications are also provided.

Prepare a High-level Written Description of the Domain of Interest (Step 1). During the initial interaction with MSOCC managers and designers, it became clear that operators were vital to the success of the system, though neither managers nor designers initially believed this. A number of “limits to automation” reduced the extent of the automation that was finally implemented and required operators to weave together various automated subsystems into a coherent whole. MSOCC operators were responsible for ensuring that the system worked and quickly correcting any faults. The goal is to collect as much “good” science data as time limits for a pass allow.

Function Allocation: Human and Computer Agents (Step 2). Although it is a mantra of all human factors engineering that each function should be allocated to the agent (human or computer) that can best carry it out (Chapanis, 1970), function allocation for the new MSOCC system proceeded typically: all activities that could be automated—within budget constraints—were automated, leaving operators to perform all other activities. Moreover, operators were expected to fill the gaps between automated subsystems and compensate when the limits of automation were reached. The evolving MSOCC operator function model clearly showed the high-level activities that operators were likely to perform, in addition to monitoring the system.

MSOCC designers and NASA management could have used the MSOCC operator function model as an engineering tool to evaluate staffing needs, project operator workload, specify workstation hardware, and design workstation software. Instead, and somewhat arbitrarily, NASA planned to staff the new MSOCC system as they had staffed the old system: around the clock with at least three operators on console, that is, each operator sitting at a MSOCC workstation, and a supervisor. A thorough review of system documentation never showed any rationale or analysis to justify the planned staffing levels for the new, highly automated system.

Workstation Design: Displays and Controls (Step 3). This subsection describes two workstations: the baseline workstation proposed by NASA and a workstation designed using OFM-based design methodology. We begin with a description of the base-line workstation.

*The baseline MSOCC operator workstation.*⁷ Most systems specify workstation hardware and software long before details on specific operator activities are considered. This was also true for the new MSOCC system. Given staffing information, designers planned to have four MSOCC workstations—one for each of

⁷The baseline workstation used in this study extensively replicated, from the operator’s point of view, the proposed MSOCC workstation (Computer Sciences Corporation, 1983). The baseline workstation is very similar to the workstation that MSOCC eventually implemented around 1986, and, with some modification, is still in use today.

the three operators and the supervisor. Each MSOCC workstation would consist of several monitors and a keyboard for command-line input. Since the designers were unaware of any operator functions other than monitoring the automation, they had little rationale for configuring the workstation or its associated display pages, except along physical and functional system characteristics.

There are three obvious MSOCC functional features: (1) information about schedules; (2) information about hardware status and active networks; and (3) data and data-quality counts for each piece of equipment composing the overall system. Given these three functions, designers specified a three-monitor MSOCC workstation. The left monitor contained all schedule-related information. There were hundreds of schedule pages corresponding to the overall system, individual spacecraft, equipment classes, and specific pieces of equipment. The second (middle) monitor consisted of one graphics display that depicted the current hardware status for every piece of MSOCC equipment and the networks that were currently supporting spacecraft contacts. This page was completely filled by the specified information. No other MSOCC equipment, using the proposed design, could be added; no consideration was given to the inevitable addition of hardware or how its status would then be displayed. Finally, as the system's primary responsibility was to ensure the transmission of good-quality data between the spacecraft and the ground system, the third monitor contained dozens of alphanumeric pages displaying the amount and quality of data received, both incoming and outgoing, at every piece of MSOCC equipment, whether the equipment was operating or not.

These displays, organized by system, rather than operator function, required operators to do a tremendous amount of work. For example, it was not unusual for the operator to request ten or more display pages, copy low-level data onto paper, and perform extensive paper-and-pencil calculations to ensure that data were flowing through each active network properly and were acceptable. Thus, monitoring and troubleshooting, if a problem was suspected, or carrying out most of the other major operator functions, were difficult and error-prone tasks.

Essentially, this workstation violated every step of the proposed workstation design construction and validation process. There was no operator model; thus, most model-development steps could not be carried out at all. Traditional design reviews were the only methods of assessment.

An OFM-based MSOCC operator workstation. In the first application of the operator function model for workstation design, Mitchell and Saisi (1987) used the OFM to specify workstation hardware requirements and sets of display pages and information requests as specified by the MSOCC OFM. The descriptions below summarize corresponding features of the baseline vs. the OFM-based workstation. The OFM-based workstation design methods were used to assess the baseline workstation and to specify, and partially validate, the utility of, the OFM.

The MSOCC OFM-based workstation had two very different types of displays. Each type corresponded to one of two major operator functions: monitoring and fault detection vs. system reconfiguration. As Figure 21.8 shows "control of current mission" is an on-going monitoring activity. Its decomposition shows that this activity consists of both routine monitoring of each network and more detailed monitoring or troubleshooting when a fault is suspected. Each currently active

network was represented by a window containing two icons: a spigot icon and an hourglass timer. The spigot icon integrated the features of the network important to the MSOCC operator: Are the data flowing? Are they flowing at the correct rate for the given network configuration? Is the error count reasonable? The width of the spigot icon depicted the *expected* data flow at the end point of the system—the spacecraft-specific control room. The width of the fluid flowing from the spigot depicted the *actual* flow rate at the same point. Significant differences in widths indicated a possible problem. The fluid representing data rate flowed from the spigot icon into a bucket. Error accumulation was depicted by red dots at the bottom of the bucket. The second icon, the hourglass timer, depicted the time remaining in the pass. By examining remaining time and the current number of errors, MSOCC operators could estimate whether the error count was reasonable or not. These metaphors defined a high-level qualitative monitoring display: there was one spigot and one hourglass timer for each active network. If a fault was suspected, the operator could request additional information about the suspect network, which was displayed as a network of spigot icons depicting expected and detected flow rates and error counts at each component in the network. The second window was a more detailed, though still a qualitative display, and permitted the operator to troubleshoot quickly and accurately a suspected network. These two windows shared a single monitor, supporting monitoring and fault-detection.

The second type of activity, the conditional activities depicted at the top level of Figure 21.8, all relate to system reconfiguration and require a different type of display than the monitoring and troubleshooting representations. The OFM was used to design a series of help requests that corresponded to each high-level activity except “control of current missions.” By requesting help, the operator obtained required information to either carry out the system configuration or decide that it was not possible. This information was a set of alphanumeric windows displayed on the same page. The contents of each window were tailored by the needs specified by the OFM and current system state. For example, for the activity decomposition in Figure 21.8, “configure to meet support requests,” the operator would receive a request from a spacecraft, say ERBS, to see if they could have a pass for the next five minutes. The operator input the help command: help configure ERBS 5. This command brought up a set of windows corresponding to the decomposition of that activity in the OFM. A window determined the feasibility of “configure to meet support requests” by checking to ensure that adding another network for the next five minutes would not violate the five-network constraint. If this constraint would be violated, another window provided this information to the operator, who then communicated to the controller that the support was not possible. If the support was possible, the help request automatically displayed a spacecraft-specific network template together with a window for each equipment type. Equipment windows showed individual components that were available for the entire time in question. Given this information, the operator could easily configure an ERBS network with the correct equipment types and individual components that were free for use. This type of interface was easy to use and resulted in few operator errors due to memory constraints or input error.

Thus, the OFM workstation consisted of two monitors: one support monitoring and fault detection with the qualitative graphical icons depicted at two levels of

abstraction, and another to provide contextually relevant information, displayed on the same page, to enable the operator to evaluate the feasibility of a manual activity.

Results. The OFM-based workstation was designed to meet the activities specified by the operator function model: in number of monitors, types of displays, display contents, and operator commands. Essentially the OFM guided software developed to provide contextually relevant information in response to help requests and presents the iconic panels to support on-going monitoring and, as needed, troubleshooting.

Many walk-throughs of the OFM-based workstation were conducted. Each walkthrough led to improvements until the designers/researchers felt that the workstation was ready for experimental evaluation. The project concluded with a controlled experiment. A between-subject experiment compared performance using the two workstations on a variety of measures. Not surprisingly, the evaluation demonstrated that the OFM-based workstation, on almost all measures, significantly enhanced operator and system performance. Where performance was not significantly better, performance with the OFM-based workstation was *always* qualitatively better (Mitchell and Saisi, 1987).

A subsequent project in the area of model-based design for operator workstations, extended the OFM to include information flow and intentionally designed human-computer interaction into the workstation (Thurman, 1995, 1997; Thurman and Mitchell, 1994, 1995). The latter feature—an interactive operator workstation—was an attempt to mitigate the inevitable boredom associated with extensive monitoring in increasingly automated systems (Bainbridge, 1983). The enhanced OFM and the proposed design methodology were implemented in proof-of-concept form for another NASA satellite control system, the spacecraft-specific control room. The empirical evaluation improved on the Mitchell and Saisi results. The OFM-based workstation consistently and significantly improved operator performance on *all* measures. Moreover, a number of its features have become part of the standard NASA suite of software available to individual spacecraft control systems.

A similar result was obtained in a manufacturing application in which an operator function model was used to specify information for displays used by a human supervisory controller of a manufacturing system. Again, the OFM-based workstation resulted in significantly better system performance (Dunkler et al., 1988).

Intelligent Aids and Associates (Step 4). Knowledge acquisition and knowledge representation are the most difficult parts of creating an intelligent, advice-giving system (Buchanan and Wilkins, 1993; Forsythe and Buchanan, 1993). Obtaining accurate and complete operational knowledge, by designers often lacking domain and operational knowledge, is very difficult. Knowledge acquisition is typically performed by identifying a single-domain expert, and perhaps supplementing the expert's views with less rigorous interviews with other personnel, such as other operators, management, and designers, and by reviewing system documentation. These methods have obvious limitations. When a knowledge base depends primarily on a single-domain expert, the broad range of experience, alternative techniques, and so on, of the set of operators is not represented. Interviews and other off-line information-gathering techniques are obtained out of context. Context, however, is very important in dynamic systems. It is

often necessary in order to understand the conditions under which something occurred: the what, how, and why.

The process of developing an OFM is an alternative to typical knowledge engineering. The extensive on-site observations required to develop and, subsequently, verify and validate the OFM ensures that extensive operational control and system knowledge is gathered and represented formally in the OFM's hierarchic/heterarchic network.

The OFM and OFMspert both facilitate knowledge representation that can be inspected, refined, or repaired. The visual form of the OFM helps operators and other domain experts ensure that the model is complete and correct. As implemented in OFMspert, the ACTIN blackboard provides a means for representing dynamic information visually. Operators and other domain experts can validate OFMspert by observing ACTIN, as the system dynamically posts expectations for operator activities and attempts to interpret detected operator actions. Furthermore, the ACTIN display and corresponding information assessments provide a straightforward mechanism that operators and designers can use to further inspect, correct, or verify the system. Moreover, as both the OFM and OFMspert use a representation that closely emulates the *manner* in which operators carry out required activities, a mutually intelligible framework for discourse is defined. As noted earlier, mutual intent inferencing between operators and automation is one of Billings' (1997) primary principles of human-centered automation.

Intent inferencing. The first activity in creating an artifact with computer-based intelligence is to specify the intelligence itself. For control of a dynamic system, the OFM provides a prescriptive specification of operator activities to meet various system goals over a range of nominal and off-nominal operating conditions. Thus, the OFM, or a similar prescriptive model of operator control, provides an initial definition of intelligence.

The definition of OFMspert as derived from an OFM is intended to specify, in a principled and formal manner, a mechanism to encode the intelligence embedded in an operator function model so that it can be used computationally by a range of applications. As noted earlier, in the Mitchell and Saisi model-based workstation, the MSOCC OFM guided the design and implementation of workstation hardware and software, but it did so in an *informal* manner.

The original OFMspert was implemented in proof-of-concept form for the MSOCC system (Rubin et al., 1988). Its current problem space included state variables to represent spacecraft networks, equipment status, and MSOCC schedules, and data and error block counts at each piece of equipment. The operations model, originally called the *enhanced normative model* (Rubin et al., 1988), contained both activity trees derived from the OFM and triggers that OFMspert used to post trees on the ACTIN blackboard. The operations model also contained triggers that allowed ACTIN to post operator actions on the blackboard and attempt to link them to current activities. Finally, the operations models for MSOCC contained operator heuristics, not included in the OFM, that operators often used to troubleshoot suspected networks.

The initial proof-of-concept implementation showed that OFMspert could indeed represent dynamically changing state data for the MSOCC system, correctly post corresponding activity trees, and typically link actions accurately to posted activities. The primary performance measure in this first step, however, was to

show that the OFMspert concept could be implemented, run in real time, predict reasonably correct operator activities, and interpret in a reasonably correct manner actions that supported them (Rubin et al., 1988). The next step was to demonstrate empirically that OFMspert could perform dependable-intent inferencing or operator activity tracking. Jones et al. (1990) undertook an extensive evaluation of OFMspert as implemented for MSOCC. The experiment involved comparing previously collected MSOCC data from the Saisi and Mitchell experiment as well as verbal protocols from two additional subjects performing the complete MSOCC experiment in the control condition. The data were paired: for each action, there was an interpretation from either a domain expert (original MSOCC data) or the subjects and an interpretation from OFMspert. A nonparametric sign test showed significantly good matches, that is, matches that could not have occurred by chance, for all system commands and most display requests. *Post hoc* analysis suggested that when there were poorer matches, that is, not significant matches, they occurred for activities that were less structured, for example, planning or browsing. The latter was an unanticipated result, but one that has consistently reappeared in similar applications (Callantine et al., 1998). Operators like to interact with the system, testing it, confirming their understanding, and so forth, particularly when no other active operational activities require their attention.

Another implementation of OFMspert, tailored to navigation for the Boeing B-757 flight deck, showed similar results (Callantine et al., 1997, 1998). While "flying" a B-757 desktop simulator, ten certified B-757 line pilots working for the same airline performed 2,089 actions. OFMspert correctly interpreted 92 percent of them. The *post hoc* analysis showed that the remaining 8 percent consisted primarily of browsing actions.

One OFMspert implementation demonstrated at least one OFMspert limitation. OFMspert was implemented for navigation of a Boeing B-727 aircraft (Smith, 1991; Smith et al., 1990). The B-727 is a mixed-mode aircraft in that pilots can choose to hand fly it using continuous control actions with throttle, stick, and rudders, or they can use discrete actions on the mode control panel to set a desired heading or altitude. The autopilot then controls the plane to fly to the specified heading or altitude, and hold it until it receives further pilot commands. OFMspert for this application accurately interpreted the discrete actions involving inputs to the mode control panel, but had a great deal of difficulty interpreting continuous actions associated with manually controlled flight.

Aiding and assistance. The ultimate goal of systems such as OFMspert is to use its dynamic understanding of the operator or operations team to either provide advice or act as an electronic assistant. Bushman et al. (1993) first used the MSOCC OFMspert to create Ally. Ally used OFMspert for its intelligence about current system state and to hypothesize operator activities. In addition, Ally augmented OFMspert with control capabilities through which the operator could delegate a control task or portions of a control task at various levels of abstraction and aggregation. The OFM defined the various choices Ally offered to the operator; for each high-level OFM activity, Ally offered advice or assistance with options corresponding to activity decomposition. An experiment compared two teams controlling the MSOCC system: one team consisted of two human opera-

tors and the other of a human operator and Ally. The analyses of variance of operator actions were within subject designs and the second operator in the two-operator team was a confederate. The data showed that there were no significant differences between the two teams for any measure of performance. Not surprisingly, however, *post hoc* analysis revealed that operator strategies were quite different depending on whether a team had a human or computer-based assistant.

In another NASA satellite ground control system application, Jones and Mitchell (1995) again used OFMspert as the basis for the design of an operator assistant. The goal was to define principles of human–computer collaboration for control of dynamic systems. Georgia Tech Mission Operations Cooperative Associate (GT-MOCA) blended the proposed principles of collaboration with OFMspert inferences about operator needs and current workload in order to specify an operator-controlled assistant. An experiment, similar to Bushman et al.’s, showed that, with NASA operators serving as subjects, two-person teams, and teams consisting of one NASA operator and GT-MOCA were indistinguishable on all measures.

Intelligent Tutoring Systems (Step 5). Models comprise two of the primary components of an intelligent tutoring system (ITS) (Figure 21.3): the expert model and the student model. OFMspert, with its embedded OFM, offers a potentially productive starting place for development of both models. A well-verified and extensively validated OFMspert can serve as an expert model. For use in an ITS, however, it requires extensions that support novices rather than experts. Moreover, if the student model is an overlay model—that is, a model with the same form as the expert model—OFMspert again provides a robust foundation. As instruction and student–system interaction progress, the ITS annotates the student model and tailors instruction based on comparisons between expert and student models.

Chu et al. (1995) proposed an architecture for an ITS for students learning to control dynamic systems. Both her expert and student models began with OFMspert; extensions were added to accommodate tutoring. Georgia Tech Visual and Inspectable Tutor and Aid (GT-VITA), a general architecture for tutoring operators of complex systems, included declarative knowledge, procedural knowledge, and operational skill. Declarative knowledge was added to entities in the OFMspert current problem space to explain purpose, function, and behavior. Procedural knowledge and operational skill were added to the OFMspert operations model. A separate pedagogical module was also added. Pedagogy defines teaching styles. GT-VITA demonstrates operational skill and coaches students who are practicing operational skill using various levels of assistance. For example, as student skills increase, tutor interventions decrease. The ACTIN blackboard was used to represent expected operator activities, given the current state. Knowledge sources provided information to the tutor to guide students through procedural and operational activities. GT-VITA was implemented in proof-of-concept form for the same NASA satellite control system used by Jones and Mitchell (1995).

Evaluating something that did not exist before, at least in a tractable form, is very difficult. Since current NASA training is almost exclusively informal, on-the-job training, extending over a three to six month period, it was impossible to conduct a controlled experiment to compare the tutor with the actual alternative. Thus, a study was conducted, similar to traditional classroom assessment, to determine the extent to which students learned the material that the tutor was designed to teach. Sixteen employees

of NASA Goddard who worked in the area of satellite ground control, but were novice operators—that is, they did not have previous training in satellite control—participated in this study. The overall results, both objective and subjective, showed that participants were able to describe correctly all declarative and most procedural knowledge for satellite ground control. For operational skill, participants performed well for *all* on-line tasks. Participants, however, had problems learning to perform correctly off-line tasks (e.g., bookkeeping), but neither the operator workstation nor the tutor concentrated on these activities. The highest compliment to this work is that the tutor, GT-VITA (Chu et al., 1995), is still in operational use today at NASA Goddard, almost seven years after its initial demonstration and evaluation.

Tutor-aid Paradigm (Step 6). In addition to making a contribution to intelligent tutoring, the two sets of research performed by Jones and Mitchell (1995) and Chu et al. (1995) proceeded well down the path toward demonstrating the viability of a *student-aid paradigm*, as depicted in Figure 21.2. Both used the same domain of application, and both were based on the same OFM/OFMspert models. In GT-MOCA, OFMspert was the expert model for the intent inferencer. In GT-VITA, OFMspert provided the basis for both the expert and student models. As noted before, a tutor that evolves into a well-known and well-understood assistant may alleviate some “automation trust” concerns and certainly reduces the expense and possible errors associated with maintaining as many as three different models for two different software systems.

21.7.2 Lessons Learned

The series of research projects that led to the model-based design methodology described in this chapter taught many lessons over the 20 years that the work spans. These include limitations of using microworlds, especially if your results are surprising, the importance of including software specification in the interface between a research theory and a proof-of-concept implementation, the importance of a correct model, and the possibility that model-based design may compensate in part for individual differences among operators of complex systems.

21.7.2.1 Microworlds. In some respects, failures, which are often not published, teach more than successes. The results of the experiment evaluating Mitchell and Miller’s (1983, 1986) model-based displays designed with the discrete control model of operator function were both unexpected and undesirable: neither set of model-based displays enhanced performance over a typical SSSI static display, and one model-based display type significantly decreased performance. Analysis of the data showed that the “unexpected” and “surprising” results could be attributed to any number of factors, and thus both the researchers and the research community were left with few conclusions or explanations for these results. Initially, we speculated that perhaps the model itself was incorrect, but then concluded that if this was true, *both* model-based display conditions should degrade performance, but, the data did not support this hypothesis (Mitchell and Miller, 1986). Another explanation was that the results were due to a “fatal flaw” in the experiment itself. Our experimental domain was a microworld—a contrived system. In specifying the system, it turned out, somewhat serendipitously, that the system required exactly one dedicated display. The microworld had so few state variables that all the required information fit nicely on a single display page. This display page, however, was intended as a surrogate for

analog control rooms containing hundreds or even thousands of displays. It recalls the new nuclear power plant control room with 15,000 display pages that operators may select. For this experiment, this single page, with state-variable values updating, dynamically served as the control condition against which two sets of displays were designed using a discrete control model of operator function. In retrospect, it seems that a poor choice of a surrogate for the dedicated display condition as well as the specification of the contrived system itself might explain the experimental results.

After the unhappy results of this experiment, we never again used a contrived system for research intended to produce operational relevance. Thus, for all future work, research projects always started by identifying a generalizable, but real, system and emulating, from the operator's point of view, the look and feel of the operator–system interaction and dynamics. This typically meant that we were novices with respect to the system, but that the system itself was real and any results applied at least minimally to the system under study; in addition, they might well generalize to a wider class of systems.

The MSOCC research projects were the first to reflect this philosophy. The projects began with construction of a simulator with which to evaluate experimental work. The Georgia Tech Multi-Satellite Operations Control Center (GT-MSOCC) simulator replicated to a high degree the proposed MSOCC system. Each display page, several hundred in all, for the proposed system was exactly replicated in GT-MSOCC. The simulator carefully emulated operator–system interaction and dynamics. For example, the actual control system updates data and error block counts every 20 seconds, as does GT-MSOCC.

GT-MSOCC and its interfaces were used as the baseline experimental condition for most of the studies previously described. From NASA's point of view, these experiments had high face validity and generalized to an entire class of satellite control systems. Moreover, since traditionally analog systems continue to convert to digital technologies, it is reasonable to conjecture that the experimental results, particularly those involving model-based workstation design, generalize more widely, as human–system interaction issues are often quite similar across real-time control systems [Woods et al. (1994)].

21.7.2.2 Structure to Guide Software Development. Prior to research projects using OFMspert, a tutor for MSOCC, ITSSO (intelligent tutoring system for satellite operator), was designed using the MSOCC OFM to guide software design (Mitchell and Govindaraj, 1991). Research has many failures as well as successes, and like the OFMspert developed for navigation on the B-727, this project was not a success. Similar to the Mitchell and Saisi workstation, the MSOCC OFM was used to guide software development for this tutor, but in an unstructured and informal manner. Since this was an MSOCC experiment for which there was a controllable real-time simulator, a control condition was possible and a transfer-of-training experiment conducted. The experiment compared performance of subjects after the normal MSOCC training and system introduction (the control condition) to performance of subjects who were trained with the ITSSO tutor. To everyone's surprise, ITSSO failed to enhance performance on all but one measure. Moreover, on another measure, performance was significantly better in the no ITSSO condition.

After the unexpected results with ITSSO—for the most part the tutor did not help (Mitchell and Govindaraj, 1991)—it seemed clear that exactly *how* the OFM is used to guide software development is also important. Although OFMspert began as an exploration for the design of a dynamic tasks analysis and as the first step in constructing an operator's assistant, over time, it became clear that four facets composed the OFMspert project. First, there is the *theory* of OFMspert. Next, there is the OFMspert *architecture*, that is, a well-

defined OFM specification intended for software design. Next, there is a *proof-of-concept implementation*, and fourth, an *evaluation*. From the perspective of the research community and potential operational systems, though it is a prerequisite that the proof-of-concept implementation perform as advertised, the more interesting OFMspert facets are its theory and architecture. The architecture is the tool that enables designers to use OFMspert in other domains.

There are many ways to move from a theory to an architecture that implements the theory. For example, the OFM theory states that a hierarchic/heterarchic network of finite-state systems is a useful way to represent how operators control complex systems. Moving from this theory statement to OFMspert required much thought, experimentation, and evaluation, and resulted in a variety of mistakes.

OFMspert continues to evolve. The OFMspert architecture specifies the manner in which to construct a current best hypothesis about operator activities and to interpret incoming operator actions. Moreover, though at this time only manual actions can be validated, it is important to represent cognitive and perceptual actions as well; in fact, representation of cognitive and perceptual actions is essential for a tutoring system.

The HASP blackboard data structure helped initially to define the current problem space, but other requirements manifested themselves over time and individual applications. For example, OFMspert's current problem space needs information about the current system state (state variables) and how operators interpret various sets of state variables to form intentions or validate anticipated expectations (derived variables). Moreover, to understand whether the operator is accessing the needed information to make a well-informed decision, the current problem space needs a structure to represent the display pages an operator currently has in view or that have been in view recently enough that OFMspert can reasonably assume that the operator remembers the information contained on them. This requirement dictated the inclusion in the current problem space of a semantic description of the operator workstation.

In a similar way, OFMspert's operations model grew. It began with a static representation of the OFM and triggers to enable ACTIN to post activity tress at the appropriate time. It also included information about all possible operator actions—at the semantic, not the keystroke, level—and a table defining how to link each action to all possible upper level activities that the action could support. Other research projects added knowledge structures to the operations model. Weiner and Mitchell (1995) and Weiner et al. (1996) enhanced the operator workstation with a model-based anomaly archive. This component includes a case base of previously seen and resolved spacecraft or ground system problems together with models that organize the cases and help operators recognize an applicable case at the appropriate time—in real time. Moreover, both Jones and Mitchell (1995) and Callantine et al. (1998) realized that their applications, as opposed to the original MSOCC application, which had very little variation in the overall set of operator functions, required inclusion of a plan that guided operator activities, and thus OFMspert, over time. For Jones and Mitchell the plan was the pass plan that NASA operators follow carefully for each pass. For Callantine et al. it was the flight plan, which can change dynamically over the course of a flight. Without planning knowledge, it was impossible to predict or understand what operators were doing at particular times. Thus, for many applications, the operations model includes a plan archive that instantiates a plan for each mission—for example, flight or pass—for which OFMspert is implemented. The need for a semantic action mapper in the operations model grew as technology changed. Even in the original baseline MSOCC operator workstation there are multiple actions that operators can use to accom-

plish the same thing, typically different means of obtaining the same information. The fact that different *syntactic* actions are the *same semantic* action became even more obvious with the proliferation of graphical user interfaces in control rooms. Command-line input was not eliminated, but it was augmented by point-and-click and drag-and-drop options. This trend is likely to continue. Thus, OFMspert's semantic action mapper will be an increasingly vital component of its operations model.

21.7.2.3 Operator vs. Model Error. Another application highlights an important lesson in model-based design. With expert operators, and less-than-expert designers with respect to the domain, it is important to be very careful about assuming *operator*, as opposed to *model*, error. For Georgia Tech Crew Activity Tracking System (GT-CATS), Callantine (1996) and Callantine et al. (1998) substituted the ACTIN blackboard and its notion of *maximal connectivity* of operator actions with an operator function model and an associated software implementation that dynamically specified a preferred course of activities. The preferred or expected course of activities extended from high-level activities, such as predicted selection of a flight navigation mode, to low-level operator actions. Callantine et al. included a revision process to interpret valid, alternative choices that were available, but not preferred. Callantine et al.'s data showed that, even with certified Boeing-757 pilots from the same airline, flying under near-perfect conditions, almost 50 percent of pilot actions triggered the revision process, that is, pilots chose some other path than the path GT-CATS predicted as preferred. This is an example of model error, or at least a model limitation. When queried, pilots always had a reasonable explanation for why they chose a particular series of actions, but the explanation was not included in the model. Reasons included personal style, weather, or traffic in the area.

21.7.2.4 Subject-to-Subject Variability and Model-Based Design. Another interesting result has occurred several times as researchers began to use model-based design methodologies, realistic simulations of actual human-system interaction, and subjects who are experts in the domain of study. The result involves individual differences and the propensity of model-based designs to influence how individual operators make decisions: the data have begun to suggest that model-based design may help *reduce* individual differences.

The first case occurred in Miller's (1985) reexamination and analysis of Mitchell and Miller's data (1986). Although Mitchell and Miller's displays did not produce the expected outcome, Miller concludes: "From the models, it is clear that the (types of) displays are not the same in terms of their propensity to encourage good decision-making strategy (p. 244)."

Although Georgia Tech engineering students were used as subjects, the Mitchell and Saisi (1987) research on OFM-based workstation design analyses of variance showed that subjects were not significant factors affecting the measures of performance. This result occurs rarely in between subject designs.

Two more recent studies confirmed these results with real operators as subjects. Thurman et al.'s *interactive* operator workstation methodology was implemented for another NASA ground control system (Thurman, 1997; Thurman and Mitchell, 1995). The data show that the OFM-based workstation significantly enhanced subject performance on all measure of performance, whereas the subject was not statistically significant for several measures. In another proof-of concept implementation (Weiner et al., 1996), evaluation of the operator's workstation showed significant improvement in all performance measures with subjects not significant for more than half of the measures.

21.7.2.5 Summary. As the OFM and OFMspert are used for new applications and in the context of new domains, there will be additional lessons learned. Probably the most important contribution of these two models, in addition to their successful use as the foundation of a model-based design methodology, is the continuity of this research. Rather than traditional, one-shot laboratory experiments, the methodology, models, and applications have been implemented, evaluated, and refined over a long period of time and in a variety of real-world situations.

21.8 BASIC RESEARCH AND OPERATIONAL RELEVANCE TO REAL-WORLD DESIGN

The research base to guide effective design of computer applications is very weak and, as a result, has little, or no, impact on the design of real-world systems (Campbell, 1998; Carroll and Campbell, 1989; Mackie, 1984; O'Hara, 1994). It is vital, however, to conduct meaningful research and to provide guidance to real-world designers. Computers are ubiquitous and enormous strides are being made in technology. As a result, computer applications proliferate, affecting almost every aspect of modern life. This is true in everyday applications, safety-critical applications, and applications in which system failure is extremely costly, for example, telecommunications, power, and electrical networks. Ineffective design produces applications that are hard to use, expensive to train on, or possibly dangerous. Effective and usable design guidance is mandatory, and a primary source should be the enormous amount of research conducted in academic and research-and-development laboratories.

The failure to transfer laboratory results to real-world applications is often attributed to the methods employed by the research community, particularly academic researchers (Carroll and Campbell, 1989; Mackie, 1984; Wolfe, 1966). Academic research related to human factors is often guided by the paradigm and methods of experimental psychology. Experimental research has traditionally favored the study of narrow, artificial tasks, selected not because they illuminate anything typical or significant, but because they are tractable for study in the laboratory. Mackie's (1984) paper on research relevance and the information glut describes the problem succinctly:

Experimental psychologists are busily engaged in the production of prodigious amounts of information which, whatever its scientific merits, is having little impact except on small groups of other experimental psychologists. It is an unfortunate fact that very little of this large, high-quality scientific output offers useful direction for solving problems of the kind on which we claim to be working. . . . This is certainly not because the amount of ongoing scientific effort is lacking in either quality or quantity. Rather, it is because so much of our research is the wrong research. . . . In the interests of scientific control, psychologists have been taught to employ simplistic, highly constrained experimental tasks that are often *invented* by the experimenter to suit the convenience of hypothesis testing. . . . Guided by their hypotheses and the particular characteristics of conveniently available experimental apparatuses, they invent perceptual games, decision-making games, learning games, and 'stress' conditions that may admirably serve their experimental objectives but will also ensure that great difficulty will be encountered in operationalizing their results, whether the attempt is made by others or even by the behavioral scientists themselves. (p. 1)

As a result, it is not surprising that laboratory-bound research has produced little if any, impact on the design of computer systems or applications (see Carroll, 1991). Mackie's

critique of traditional research methods and why there is so much research and so little information is particularly applicable to the problem of transferring to real-world applications research on the design of human interaction with complex systems conducted in laboratories.

This section proposes a metatheory for conducting basic research, particularly research in university and research-and-development laboratories, that is more likely to produce believable and usable results for real-world designers. The metatheory attempts to address the underlying causes of the widespread belief that laboratory research has little, if any, impact on real-world design. First, some history is provided to explain in part why traditional methods were employed and the subsequent inability to transfer these results to realistic applications. Second, the processes that compose the metatheory are presented, both for field-study research and artifact development. Finally, generalization of results from a particular application to a wider class of systems is discussed.

21.8.1 Historical Origins

One frequently given reason for the lack of relevance of university and laboratory-based research is the research methods employed, including toy tasks and naive subjects (Carroll and Campbell, 1989; Mackie, 1984). These methods, however, were used in part because of technology constraints.

Before the advent of inexpensive and portable workstations, researchers were confined to laboratories that housed immobile computers and restricted to available subject populations, often university students with little or no domain experience. Physically, it was difficult or even impossible to conduct controlled experiments in realistic conditions with subjects who were actual users or operators in the domain of application. Thus, research employed contrived tasks with uncertain relevance to real-life applications. Experimental subjects were students who may have had 30 minutes, or at best 6 to 10 hours, of training—significantly different from real-world training programs that span weeks or months. Given these constraints, it is difficult to convince engineers and designers in the real world that such results are valid or generalize beyond university or laboratory walls. Historically, moreover, hardware and software incompatibilities were such that even if a concept or artifact was highly successful in the laboratory and potentially useful in the real world, transfer to an actual work environment was too expensive, too hard, too slow, or all three.

The emergence of portable, inexpensive computers challenges the need to continue the traditional academic research paradigm. Advanced technologies allow researchers interested in understanding and aiding human interaction with complex systems to exchange laboratories, microworlds, and naive subjects for the challenges and opportunities of actual work domains. Workstation technology, together with standardized hardware and software platforms, permit realistic demonstrations and empirical evaluations to be conducted on-site with actual users. Research based on actual tasks and evaluated by real users has high face validity within the domain of application, and is much more likely to generalize to comparable systems.

In addition to new challenges and opportunities, new technology permits new, and perhaps demands, more effective research methodologies. Rasmussen (1988), in his keynote address to the Human Factors and Ergonomics Society, summarizes the challenge as follows:

In a way, the effect of computerization is to activate rather suddenly the problems that already existed but were latent. . . . Automation creates higher-level tasks of diagnosis and disturbance control; centralization makes the consequences of errors less acceptable; and the flexibility of information technology requires that designers know how to match the interface to a user's resources and preferences. . . . (I)t creates a need for methods to analyze complex work settings, not well-defined laboratory experiments. (p. 2–3)

Thus, the tasks must be realistic, evaluations must reflect both the tasks and the context in which the tasks are performed, and, finally, participants in evaluations must be actual users. For those long-standing practitioners of traditional academic research, field-oriented research will be significantly different and very challenging. The end, however, operational relevance, may be well worth the effort.

The following section proposes a process for conducting academic research that may be more likely to produce results that address real-world problems. It encompasses two major approaches: field studies and artifact construction and evaluation.

21.8.2 A Metatheory for Conducting Academic Research

Figure 21.9 depicts a metatheory for how to conduct research that mediates between the historical problems of field studies and laboratory-based research and the concerns about validity and generalizability of such research in real-world applications. Whether the research goal is generalization of insights from one or more field studies or the construction and evaluation of an artifact, the first phase of the process requires extensive observation of one or more real-world systems.

In Phase 1, researchers first select an *interesting* system. Such a system is one that is available for observation by researchers, and one in which research results are likely to generalize beyond the specific system being studied. The second step in the observation phase dictates extensive field study to identify relevant issues and understand salient characteristics of the work domain (Mackie, 1984; Meister, 1988; Rasmussen and Goodstein, 1988).

The second phase, hypothesize, is one that is often implicit. Research, as opposed to technology development, however, requires formulation of a theory to characterize either insights that describe the work domain and its operations personnel, or system features and limitations that a research artifact can potentially ameliorate. Theory forms the basis of the research hypothesis or thesis. If the research approach is artifact or tool development (i.e., engineering), a second step in the hypothesize phase is required: specification of a domain-independent architecture to address the issues raised in the theory. For example, the widely cited Pilot's Associate project began with the premise that fighter pilots need assistance. The hypothesis was that a computer-based assistant could be built and would be useful. The domain-independent architecture was independent in the sense of being independent of a particular aircraft type or pilot. It is important to note that the architecture is platform-, hardware-, and software-independent. Essentially, the architecture is a concept with more definition than a theory and less definition than a computer implementation.

Figure 21.9 depicts the process of field study research in the lower half of the figure. Much of the insight that guides design at high levels results from this type of research. Essentially, concepts, such as human-centered automation (Billings, 1997), clumsy automation (Wiener, 1989), the keyhole effect and visual momentum (Woods, 1984), auto-

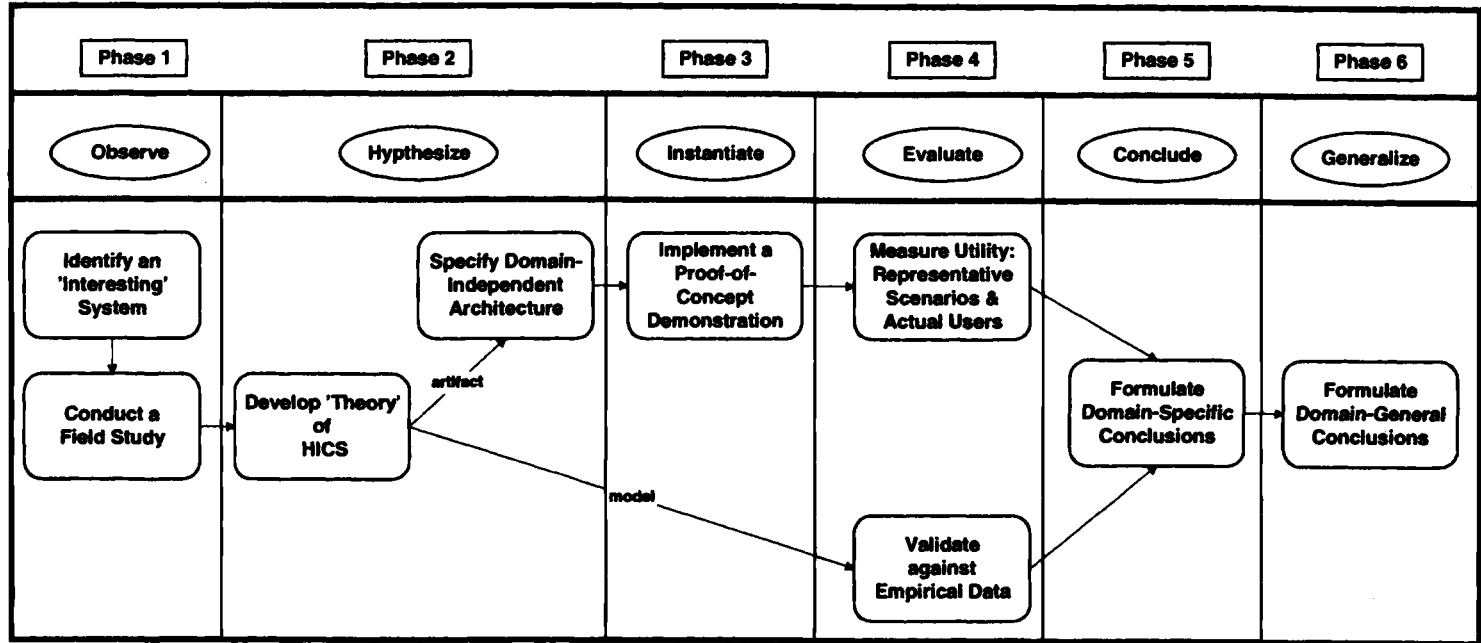


Figure 21.9 A metatheory for conducting research with operational significance in the design of HICS.

mation complacency (Sheridan, 1992), and the need for design that supports maintenance of accurate operator models (Sarter and Woods, 1995a, b), are derived from extensive field studies both within a domain and across domains (Woods et al., 1994). Given a theory such as the keyhole effect or visual momentum, researchers typically validate the theory by conducting additional field studies either in the same or a related domain. Next, domain-related conclusions or recommendations are proposed. Finally, since the goal of all research is to provide results that generalize, the last phase of the proposed process requires researchers to carefully specify under what conditions the theory generalizes to other systems.

The top portion of Figure 21.9 depicts a research process that has many points of correspondence with the traditional academic research paradigm. Where it differs, however, the differences are significant. As noted earlier, the first phase is to identify and study an actual system. Contrived tasks or microworlds are not adequate substitutes. This process requires a detailed understanding of a real domain, its goals, limitations, problems, and complexity. For the design of computer-based artifacts, it is imperative to state explicitly a theory that your research will prove or disprove. Specification of a theory provides the basis for subsequent generalization. The next step is specification of an architecture that embodies the characteristics specified in the theory. Phase three is a modeling or implementation phase in which the architecture is instantiated and typically linked to the system under study or a high-fidelity simulation of the system.⁸ The link, either the real system or a high-fidelity simulation, allows controlled evaluation. Evaluation should address both usability and utility. Usability and associated testing has become the watchword of human-computer interaction design. Usability studies typically seek to determine the ease with which a system can be learned or used. Designers of technology that affects operators responsible for complex systems must also ensure *utility*. Evaluation of utility seeks to measure the added value derived from use of the proposed artifact. Experimental evaluation allows the determination of whether the artifact significantly enhances performance. Moreover, and perhaps most importantly, portable and inexpensive workstations permit evaluations to be conducted on-site with actual users. Like field studies, the next phase is to draw conclusions, most of which are domain-specific. The last phase discusses how to generalize research results more widely.

21.8.3 Generalizability

Given the proposed research process, two levels of conclusions can be drawn: the first with respect to the reference system that motivated the research; the second to systems that share important features with the reference system. The latter are conjectures intended to generalize to a class of applications. By specifying salient characteristics of the domain of application, researchers can characterize other domains to which the research results are likely to apply. Clancey (1989) calls such conclusions *domain general*, as opposed to *domain independent*. He suggests that for realistic applications there are likely to be few domain-independent results and, thus, researchers should carefully construct research programs and specify concrete characteristics of the system that can produce conclusions that

⁸As noted before, if a simulator is used, its fidelity should closely resemble both the look-and-feel and human-system dynamics. A test for simulator adequacy might be whether or not operators of the original system, including experimental participants, believe that results obtained via the simulator apply to the real system.

generalize to a set of related systems. Research has as its goal the evolution of sets of design methods and principles appropriate to large classes of application—not just a single system, for example, the SAMPEX NASA spacecraft, nor a class of related systems, for example, the set of NASA near-earth satellite. The proposed metatheory for conducting research with operational significance attempts to address this issue: results have high face validity in the original domain of application, and, given careful specification of the salient characteristics of the domain from the “human-interaction-with-complex-systems” point of view, may well generalize more broadly to a larger class of systems that possess the specified characteristics held in common with the original application.

21.9 CONCLUSION

This chapter attempts to lay out important design issues for effective human interaction with complex systems in a world in which technology allows, or even encourages, systems to change quickly and often to include automation that replaces activities in complex systems that historically have been carried out by humans. First, design guidance and levels of human–system interaction were discussed, with limitations in design knowledge pointed out. Next, a set of important issues in human interaction with complex systems was proposed. The heart of the chapter is the description of a design methodology—model-based design. A generic formulation of the methodology was initially presented. A specific implementation using the OFM as the specific representation. To illustrate the use of the methodology, one or more applications using OFM-based designs for each issue were presented. A lessons-learned section discussed issues raised by various OFM model-based design applications. The chapter concludes with a challenge to human factors practitioners and others concerned with operational relevance of human–system interaction research, to use methods that better reflect real-world characteristics and actual users. Inexpensive and powerful computer technology creates many opportunities to conduct research with much more operational significance. A metatheory for a process with which to conduct academic research that is an alternative to the traditional experimental research paradigm is proposed. The author’s and her colleagues’ work, as well as much of the work cited in the chapter, follows successfully some variation of this paradigm and produces academic results that real-world systems find applicable and useful.

ACKNOWLEDGMENTS

Many organizations, colleagues, and friends made the research underpinning this chapter possible. At an organizational level they include Georgia Tech’s School of Industrial & Systems Engineering, NASA Goddard, NASA Ames, National Science Foundation, Delta Airlines, Federal Express, NCR, and Chrysler. I wish to thank my technical monitors over the years for their support, both financial and moral: Karen L. Moe (NASA Goddard), Walt Truszkowski (NASA Goddard), Bill Stoffel (NASA Goddard), Everett A. Palmer III (NASA Ames), and the many “operators” who familiarized ivory-tower novices with the real-world of operations in complex systems. The operators taught us the importance of people in ensuring overall system safety, reliability, and efficiency, along with the challenges in the changing roles of operators as, inevitably, new information and control technology are introduced into these systems. We interacted closely with many NASA Goddard satellite

controllers, airline pilots, airline dispatchers, airline instructors, and manufacturing shop-floor personnel. As the citations indicate, the majority of the research that allowed this methodology to manifest itself was conducted in collaboration with my colleagues. Individual sections are often based on papers that summarize thesis research, including that of Donna Saisi, Janet Fath, Serena Smith, Julie Chronister, Olaf Dunkler, Jim Bushman, Patty Jones, Rose Chu, Tom Pawlowski, Jim Williams, Andy Weiner, Todd Callantine, Dave Thurman, and Alan Chappell. Although Kenny Rubin was *just* a Georgia Tech undergraduate looking for a “fun” senior design project and Al Miller was my Ph.D. advisor, as the chapter indicates, each played a major role in the research and my own professional development. As always, I gratefully acknowledge the continued support and insight of Dr. Govindaraj, who at various times participated in the research and at other times provided suggestions that facilitated the projects when we seemed to be wondering. Finally, a “thank you” to the editors, Bill Rouse and Andy Sage. When asked, I always say “yes,” swear at them and myself for the length of time it takes me to write and repeatedly rewrite the promised material, and conclude by being grateful for the opportunity to integrate related but distributed ideas and projects.

REFERENCES

- Abbott, K. (1996). *The Interfaces Between Flightcrews and Modern Flight Deck Systems*. Washington, DC: Federal Aviation Administration.
- Andre, A. D., and Wickens, C. D. (1995). When users want what's **not** best for them. *Ergon. Des.*, 12(1), pp. 10–14.
- Aviation Week and Space Technology. (1995a). Automated cockpits: Who's in charge? Part 1. *Aviat. Week Space Technol.*, January 30, pp. 52–53.
- Aviation Week and Space Technology. (1995b). Automated cockpits: Who's in charge? Part 2. *Aviat. Week Space Technol.*, February 6, pp. 52–53.
- Aviation Week and Space Technology. (1998). NTSB warns of A300 display reset problem. *Aviat. Week Space Technol.*, February 9, pp. 76–77.
- Bainbridge, L. (1983). Ironies of automation. *Automatica* 19(6), 775–779.
- Baron, S. (1984). A control theoretic approach to modelling human supervisory control of dynamic systems. In *Advances in Man-Machine Systems Research* (W. B. Rouse, ed.) Vol. 1, pp. 1–47. Greenwich, CT: JAI Press.
- Baron, S., and Levison, W. H. (1980). The optimal control model: Status and future directions. *Proc. IEEE Conf. Cybernet. Soc.*, Chicago, pp. 90–100.
- Baron, S., Kleinman, D. L., and Levison, W. H. (1970a). An optimal control model of human response. Part I: Theory and validation. *Automatica* 6, 357–369.
- Baron, S., Kleinman, D. L., and Levison, W. H. (1970b). An optimal control model of human response. Part II: Prediction of human performance in a complex task. *Automatica* 6, 371–383.
- Benson, C. R., Govindaraj, T., Mitchell, C. M., and Krosner, S. M. (1992). Effectiveness of direct manipulation interaction in the supervisory control of flexible manufacturing systems. *Inf. Decis. Technol.* 18, 33–53.
- Billings, C. E. (1995). Situation awareness measurement and analysis: A commentary. *Proc. Int. Conf. Exp. Anal. Meas. Situation Awareness*, Daytona Beach, FL, pp. 1–6.
- Billings, C. E. (1997). *Aviation Automation: The Search for a Human-centered Approach*. Hillsdale, NJ: Erlbaum.
- Bobrow, L. S., and Arbib, M. A. (1974). *Discrete Mathematics*. Philadelphia: Saunders.
- Bradley, D. (1997). Operations in control rooms. Personal communications.
- Brann, D. M., Thurman, D. A., and Mitchell, C. M. (1995). Case-based reasoning as a methodology

- for accumulating human expertise for discrete system control. *Proc. IEEE Int. Conf. Syst., Man, Cybernet.*, Vancouver, BC, pp. 4219–4223.
- Brann, D. M., Thurman, D. A., and Mitchell, C. M. (1996). Human interaction with lights-out automation: A field study. *Proc. Sympo. Hum. Interact. Complex Syst.*, Dayton, OH, pp. 276–283.
- Buchanan, B. G., and Wilkins, D. C. (1993). Overview of knowledge acquisition and learning. In *Readings in Knowledge Acquisition and Learning: Automating the Construction and Improvement of Expert Systems* (B. G. Buchanan and D. C. Wilkins, eds.), pp. 1–4. San Mateo, CA: Morgan Kaufmann.
- Bushman, J. B., Mitchell, C. M., Jones, P. M., and Rubin, K. S. (1993). ALLY: An operator's associate for cooperative supervisory control systems. *IEEE Trans. Syst., Man, Cybernet.* **23**(1), 111–128.
- Kakir, A., Hart, D. J., and Stewart, T. F. M. (1980). *Visual Display Terminals: A Manual Covering Ergonomics, Workplace Design, Health and Safety, Task Organization* New York: Wiley.
- Callantine, T. J., (1996). Tracking operator activities in complex systems. Ph.D. Thesis, Georgia Institute of Technology, School of Industrial and Systems Engineering, Atlanta.
- Callantine, T. J., Mitchell, C. M., and Palmer, E. A. (1997). *GT-CATS: Tracking Operator Activities in Complex Systems* (CR-A-1997a). Moffett Field, CA: NASA Ames Research Center.
- Callantine, T. J., Mitchell, C. M., and Palmer, E. A. (1998). GT-CATS as a model of pilot behavior in the 'glass cockpit': An application and empirical evaluation. *Submitted for publication*.
- Campbell, G. (1998). When opportunity knocks. *Cogn. Times* **3** (1), 2–3.
- Card, S., Moran, T., and Newell, A. (1983). *The Psychology of Human-computer Interaction*. Hillsdale, NJ: Erlbaum.
- Cardosi, K. M., and Murphy, E. D. (eds.). (1995). *Human Factors in the Design and Evaluation of Air Traffic Control Systems*. Springfield, VA: National Technical Information Service.
- Carr, B., and Goldstein, I. (1997). *Overlay's: A Theory of Modeling for Computer-aided Instruction* (AI Lab Memo 406; Logo Memo 40). Cambridge, MA: MIT University Press.
- Carroll, J. M. (1991). Introduction: The Kittle House manifesto. In *Designing Interaction* (J. M. Carroll, ed.) pp. 1–16. Cambridge, UK: Cambridge University Press.
- Carroll, J. M., and Campbell, R. L. (1989). Artifacts as psychological theories: The case of human-computer interaction. *Behavi. Inf. Technol.* **8**(4), 247–256.
- Casey, S. (1993). *Set Phasers on Stun and Other True Tales of Design, Technology, and Human Error*. Santa Barbara, CA: Aegean Publishing.
- Chambers, A. B., and Nagel, D. C. (1985). Pilots of the future: Human or computer? *Commun. ACM* **28**(11), 1187–1199.
- Chapanis, A. (1970). Human factors in systems engineering. In *Systems Psychology* (K. B. DeGreene, ed.), pp. 28–38. New York: McGraw-Hill.
- Chappell, A. R., Crowther, E. G., Mitchell, C. M., and Govindaraj, T. (1997). The VNAV Tutor: Addressing a mode awareness difficulty for pilots of glass cockpit aircraft. *IEEE Trans. Syst., Man, Cybernet.* **27**(3), 327–385.
- Chu, R. W., Mitchell, C. M., and Jones, P. M. (1995). Using the operator function model and OFMspert as the basis for an intelligent tutoring system: Towards a tutor/aid paradigm for operators of supervisory control systems. *IEEE Trans. Syst., Man, Cybernet.* **25**(7), 1054–1075.
- Clancey, W. J. (1989). Viewing knowledge bases as qualitative models. *IEEE Expert*, pp. 9–23.
- Computer Sciences Corporation. (1983). *Multisatellite Operations Control Center (MSOCC) Data Operations Control System (DOCS) Operator Interface Design Review* (Des. Rev. NAS 5-27555). Greenbelt, MD: NASA Goddard Space Flight Center.

- Converse, S. A. (1995). *Evaluation of the Computerized Procedures Manual II (COMPMA II)*(NUREG/CR-6398). Washington, DC: U.S. Nuclear Regulatory Commission.
- Degani, A., and Wiener, E. L. (1993). Cockpit checklists: Concepts, design, and use. *Hum. Factors* 35(2), 345–359.
- Dunkler, O., Mitchell, C. M., Govindaraj, T., and Ammons, J. C. (1988). The effectiveness of supervisory control strategies in scheduling flexible manufacturing systems. *IEEE Trans. Syst., Man, Cybernet.* 18(2), 223–237.
- Fath, J. L., Mitchell, C. M., and Govindaraj, T. (1990). An ICAI architecture for troubleshooting in complex dynamic systems. *IEEE Trans. Syst., Man, Cybernet.* 20(3), 537–558.
- Fitts, P. M. (1951). Engineering psychology and equipment design. In *Handbook of Experimental Psychology* (S. S. Stevens, ed.), Vol. 65, pp. 423–432. New York: Wiley.
- Forsythe, D. E., and Buchanan, B. G. (1993). Knowledge acquisition for expert systems: Some pitfalls and suggestions. In *Readings in Knowledge Acquisition and Learning: Automating the Construction and Improvement of Expert Systems* (B. G. Buchanan and D. C. Wilkins, eds.), pp. 117–125. San Mateo, CA: Morgan Kaufmann.
- Funk, I., K. H., and Lind, J. H. (1992). Agent-based pilot-vehicle interfaces: Concept and prototype. *IEEE Trans. Syst., Man, Cybernet.* 22(6), 1309–1322.
- Garland, D. J., and Endsley, M. R. (eds.). (1995). *Proceedings of an International Conference on Experimental Analysis and Measurement of Situation Awareness*. Daytona Beach, FL: Embry-Riddle Aeronautical University Press.
- Geddes, N. D. (1989). Understanding human operator intentions in complex systems. Ph.D. Thesis, Georgia Institute of Technology, Center for Human-Machine Systems Research, School of Industrial and Systems Engineering, Atlanta.
- Geddes, N. D., and Hammer, J. M. (1991). Automatic display management using dynamic plans and events. *Proc. 6th. Int. Sympo. Aviat. Psycho.*, Columbus, OH, pp. 83–87.
- Govindaraj, T., Su, Y. L., Vasandani, V., and Recker, M. M. (1996). Training for diagnostic problem solving in complex engineered systems: Modeling, simulation, and intelligent tutors. In *Human/Technology Interaction in Complex Systems* (W. B. Rouse, ed.), Vol. 8, pp. 1–66. Greenwich, CT: JAI Press.
- Hammer, J. M. (1984). An intelligent flight-management aid for procedure execution. *IEEE Trans. Syst., Man, Cybernet.* 14(6), 885–888.
- Hunt, R. M., and Rouse, W. B. (1984). A fuzzy rule-based model of human problem solving. *IEEE Trans. Syst., Man, Cybernet.* 14(1), 112–121.
- Hutchins, E. (1983). Understanding Micronesian navigation. In *Mental Models* (D. Gentner and A. L. Stevens, eds.), pp. 191–225. Hillsdale, NJ: Erlbaum.
- Hutchins, E. L., Hollan, J. D., and Norman, D. A. (1985). Direct manipulation interfaces. *Hum. Comput. Interact.* 1, 311–338.
- Hutchins, E. L., Hollan, J. D., and Norman, D. A. (1986). Direct manipulation interfaces. In *User Centered System Design* (D. A. Norman and S. W. Draper, eds.), pp. 87–124. Hillsdale, NJ: Erlbaum.
- International Nuclear Safety Advisory Group. (1988). *Basic Safety Principles for Nuclear Power Plants* (Saf. Ser. No. 75-INSAG-3). Vienna: IAEA.
- Johannsen, G. (1992). Towards a new quality of automation in complex man-machine systems. *Automatica* 28(2), 355–373.
- Jones, P. M., and Jasek, C. A. (1997). Intelligent support for activity management (ISAM): An architecture to support distributed supervisory control. *IEEE Trans. Syst., Man, Cybernet.* 27(3), 274–288.
- Jones, P. M., and Mitchell, C. M. (1987). Operator modeling: Conceptual and methodological distinctions. *Proc. 31st Annu. Meet. Hum. Factors Soc.*, New York, pp. 31–36.

- Jones, P. M., and Mitchell, C. M. (1995). Human-computer cooperative problem solving: Theory, design, and evaluation of an intelligent associate system. *IEEE Trans. Syst., Man, Cybernet.* 25(7), 1039–1053.
- Jones, P. M., Mitchell, C. M., and Rubin, K. S. (1990). Validation of intent inferencing by a model-based operator's associate. *Int. J. Man-Mach. Stud.* 33, 177–202.
- Jones, P. M., Chu, R. W., and Mitchell, C. M. (1995). A methodology for human-machine systems research: Knowledge engineering, modeling, and simulation. *IEEE Trans. Syst., Man, Cybernet.* 25(7), 1025–1038.
- Kantowitz, B. H., and Sorkin, R. D. (1983). *Human Factors: Understanding People-system Relationships*. New York: Wiley.
- Karat, C.-M., Campbell, R., and Fiegel, T. (1992). Comparison of empirical testing and walkthrough methods in user interface evaluation. *Proc. CHI '92*, pp. 397–404.
- Kirlik, A., and Kossack, M. F. (1993). Perceptual augmentation to support skilled interaction. *Proc. IEEE Int. Conf. Syst., Man, Cybernet.*, Le Touquet, France, pp. 896–901.
- Kirlik, A., Kossack, M. F., and Shively, R. J. (1994). *Ecological Task Analysis: Supporting Skill Acquisition in Dynamic Interaction*. Atlanta: Georgia Institute of Technology, Center for Human-Machine Systems Research, School of Industrial and Systems Engineering.
- Kirwan, and Ainsworth (eds.). (1992). *A Guide to Task Analysis*. Washington, DC: Taylor & Francis.
- Knaeuper, A., and Morris, N. M. (1984). A model-based approach for online aiding and training in process control. *Proc. IEEE Int. Conf. Syst., Man, Cybernet.*, pp. 173–177.
- Knaeuper, A., and Rouse, W. B. (1985). A rule-based model of human problem-solving behavior in dynamic environments. *IEEE Trans. Syst., Man, Cybernet.* 15(6), 708–719.
- Lee, E. J. (1994). *Computer-based Digital System Failures* (Tech. Rev. Rep. AEOD/T94-03). Washington, DC: U.S. Nuclear Regulatory Commission.
- Leveson, N. G. (1995). *Safeware: System Safety and Computers*. New York: Addison-Wesley.
- Lund, A. M. (1997). Expert ratings of usability maxims. *Ergon. Des.* 5(3), 15–20.
- Mackie, R. R. (1984). Research relevance and the information glut. *Hum. Factors Rev.* pp. 1–11.
- Meister, D. (1988). References to reality in the practice of human factors. *Hum. Factors Soc. Bull.* 31(10) 1–3.
- Miller, R. A. (1978). *A Formal Model of the Adaptive and Discrete Control Behaviors of Human Operators* (RF 760373/784382). Columbus: Industrial & Systems Engineering, Ohio State University.
- Miller, R. A. (1979). *Identification of Finite State Models of Human Operators* (OSU-RP-784556). Columbus: Ohio State University, Industrial & Systems Engineering.
- Miller, R. A. (1985). A system approach to modeling discrete control performance. In *Advances in Man-machine Systems* (W. B. Rouse, ed.), Vol. 2, pp. 177–248. Greenwich, CT: JAI Press.
- Mitchell, C. M. (1987). GT-MSOCC: A domain for research on human-computer interaction and decision aiding in supervisory control systems. *IEEE Trans. Syst., Man, Cybernet.* 17(4), 553–572.
- Mitchell, C. M. (1996a). GT-MSOCC: Operator models, model-based displays, and intelligent aiding. In *Human/Technology Interaction in Complex Systems* (W. B. Rouse, ed.), Vol. 8, pp. 67–172. Greenwich, CT: JAI Press.
- Mitchell, C. M. (1996b). Models for the design of human interaction with complex dynamic systems. *Proc. Cogn. Eng. Process Control Workshop*, Kyoto, Japan, pp. 230–237.
- Mitchell, C. M. (1999). Horizons in pilot training: Desktop tutoring systems. In *Cognitive Engineering in the Aviation Domain* (N. Sarter and R. Almerberti, eds.). Mahwah, NJ: Erlbaum (in press).

- Mitchell, C. M. (1997b). *Technology and Education: Issues, Applications, and Horizons in the Workplace*. Washington, DC: National Research Council, Board on Assessment and Testing.
- Mitchell, C. M., and Govindaraj, T. (1991). Design and effectiveness of intelligent tutors for operators of complex dynamic systems: A tutor implementation for satellite system operators. *Interact. Learn. Environ.* 1(3), 193–229.
- Mitchell, C. M., and Miller, R. A. (1983). Design strategies for computer-based information displays in real-time control systems. *Hum. Factors* 25(4), 353–369.
- Mitchell, C. M., and Miller, R. A. (1986). A discrete control model of operator function: A methodology for information display design. *IEEE Trans. Syst., Man, Cybernet.* 16(3), 343–357.
- Mitchell, C. M., and Miller, R. A. (1993). A framework for model validity with applications to models of human-machine systems. In *Advances in Man-machine Systems Research* (W. B. Rouse, ed.), Vol. 6, pp. 233–293. Greenwich, CT: JAI Press.
- Mitchell, C. M., and Saisi, D. L. (1987). Use of model-based qualitative icons and adaptive windows in workstations for supervisory control systems. *IEEE Trans. Syst., Man, Cybernet.* 17(4), 573–593.
- Mitchell, C. M., and Sundström, G. A. (1997). Human interaction with complex systems: Design issues and research approaches. *IEEE Trans. Syst., Man, Cybernet.* 27(3), 265–273.
- Mitchell, C. M., Morris, J. G., Ockerman, J. J., and Potter, W. J. (1996). Recognition primed decision making as a technique to support reuse in software design. In *Naturalistic Decision Making* (C. E. Zsombok and G. A. Klein, eds.), pp. 305–318. New York: Erlbaum.
- Moray, N., and Huey, B. (1988). *Human Factors and Nuclear Safety*. Washington, DC: National Research Council, National Academy of Sciences.
- Morris, J. G., and Mitchell, C. M. (1997). A case-based support system to facilitate software reuse. *Proc. IEEE Int. Conf. Syst., Man, Cybernet.*, Orlando, FL, pp 232–237.
- Morris, N. M., and Rouse, W. B. (1985). The effects of type of knowledge upon human problem solving in a process control task. *IEEE Trans. Syst., Man, Cybernet.* 15(6), 698–707.
- Nagle, D. C. (1988). Human error in aviation operations. In *Human Factors in Aviation* (E. L. Wiener and D. C. Nagle, eds.), pp. 263–303. Orlando, FL: Academic Press.
- National Aeronautics and Space Administration (NASA) (1988). *Space Station Freedom Human-computer Interface Guidelines* (NASA USE-100). Washington, DC: NASA, Johnson Space Flight Center.
- National Aeronautics and Space Administration (NASA). (1996). *User-interface Guidelines* (NASA DSTL-95-033). Greenbelt, MD: NASA Goddard Space Flight Center.
- Nii, H. P. (1986a). *Blackboard Systems* (KSL 86-18) Stanford, CA: Stanford University, Knowledge Systems Laboratory.
- Nii, H. P. (1986b). Blackboard systems: Part I. *AI Mag.* 7(2), 38–53.
- Nii, H. P. (1986c). Blackboard systems: Part II. *AI Mag.* 7(3), 82–110.
- Nii, H. P. (1994). Blackboard systems at the architecture level. *Expert Syst. Appl.* 7, 43–54.
- Norman, D. A. (1988). *The Psychology of Everyday Things*. New York: Basic Books.
- Ockerman, J. J., and Mitchell, C. M. (1999). Case-based design browser to support software reuse. *Intern. J. Hum. Comp. Stud.* (accepted for publication).
- O'Hara, J. M. (1994). *Advanced Human-system Interface Design Review Guideline*, Vols. 1 and 2 (NUREG/CR-5908). Washington, DC: U.S. Nuclear Regulatory Commission.
- Palmer, E. A. (1987). Using the operator function model as a 'living' task analysis. Personal communications.
- Parasuraman, R., and Mouloua, M. (eds.). (1996). *Automation and Human Performance*. Mahwah, NJ: Erlbaum.

- Parasuraman, R., and Riley, V. (1997). Humans and automation: Use, misuse, disuse, and abuse. *Hum. Factors* **39**(2), 230–253.
- Pawlowski, T. J. (1990). Design of operator interfaces to support effective supervisory control and to facilitate intent inferencing by a computer-based operator's associate. Ph.D. Thesis, Georgia Institute of Technology, Center for Human-Machine Systems Research, School of Industrial and Systems Engineering. Atlanta.
- Pew, R. W. (1995). The state of situation awareness measurement: Circa 1995. *Proc. Int. Conf. Exp. Anal. Meas. Situation Awareness*, Daytona Beach, FL, pp. 7–15.
- Phatak, A. V., and Kessler, K. M. (1977). Modeling the human gunner in an anti-aircraft artillery (AAA) tracking task. *Hum. Factors* **19**(5), 477–494.
- Preece, J. (1994). *Human-computer Interaction*. New York: Addison-Wesley.
- Ragheb, H. (1996). Operating and maintenance experience with computer-based systems in nuclear power plants. *Int. Workshop Tech. Support Licens. Comput. Based Syst. Important Saf.*, Munich, Germany, pp. 143–150.
- Rasmussen, J. (1983). Skills, rules, and knowledge: Signals, signs, and symbols, and other distinctions in human performance models. *IEEE Trans. Syst., Man, Cybernet.* **SMC-13** (3), 257–267.
- Rasmussen, J. (1985). The role of hierarchical knowledge representation in decisionmaking and system management. *IEEE Trans. Syst., Man, Cybernet.* **15**, 234–243.
- Rasmussen, J. (1986). *Information Processing and Human-machine Interaction: An Approach to Cognitive Engineering*. New York: North-Holland.
- Rasmussen, J. (1987). Reasons, causes, and human error. In *New Technology and Human Error* (J. Rasmussen, K. Duncan, and J. Leplat, eds.), pp. 293–303. New York: Wiley.
- Rasmussen, J. (1988). Information technology: A challenge to the Human Factors Society? *Hum. Factors Soc. Bull.* **31**(7), 1–3.
- Rasmussen, J., and Goodstein, L. P. (1988). Information technology and work. In *Handbook of Human-computer Interaction* (M. Helander, ed.), pp. 175–202. New York: North-Holland.
- Rasmussen, J., Pejtersen, A. M., and Goodstein, L. P. (1994). *Cognitive Systems Engineering*. New York: Wiley.
- Resnick, D. E., Mitchell, C. M., and Govindaraj, T. (1987). An embedded computer-based training system for rhino robot operators. *IEEE Control Syst. Mag.* **7**(3), 3–8.
- Roth, E. M., Bennett, K. B., and Woods, D. D. (1987). Human interaction with an “intelligent” machine. *Int. J. Man-Mach. Stud.* **27**, 479–526.
- Rouse, W. B. (1978). A model of human decisionmaking in a fault diagnosis task. *IEEE Trans. Syst., Man, Cybernet.* **8**(5), 357–361.
- Rouse, W. B. (1980). *System Engineering Models of Human-machine Interaction*. New York: North-Holland.
- Rouse, W. B. (1981). Human-computer interaction in the control of dynamic systems. *Comput. Surv.* **13**(1), 71–99.
- Rouse, W. B. (1985). Man-machine systems. In *Productions Handbook* (J. A. White, ed.), pp. 52–66. New York: Wiley.
- Rouse, W. B. (1994). Twenty years of adaptive aiding: Origins of the concepts and lessons learned. In *Human Performance in Automated Systems: Current Research and Trends* (M. Mouloula and R. Parasuraman, eds.), pp. 28–33. Hillsdale, NJ: Erlbaum.
- Rouse, W. B., Geddes, N. D., and Curry, R. E. (1987). An architecture for intelligent interfaces: Outline for an approach supporting operators of complex systems. *Hum.-Comput. Interact.* **3**(2), 87–122.
- Rouse, W. B., Geddes, N. D., and Hammer, J. M. (1990). Computer aided fighter pilots. *IEEE Spectrum*, **27**(3), 38–41.

- Rubin, K. S., Jones, P. M., and Mitchell, C. M. (1988). OFMspert: Inference of operator intentions in supervisory control using a blackboard architecture. *IEEE Trans. Syst., Man, Cybernet.* **18**(4), 618–637.
- Sarter, N. B., and Woods, D. D. (1995a). How in the world did we ever get in that mode? Mode error and awareness in supervisory control. *Hum. Factors* **37**(1), 5–19.
- Sarter, N. B., and Woods, D. D. (1995b). *Strong, Silent, and Out-of-the-Loop': Properties of Advanced (cockpit) Automation and Their Impact on Human-automation Interaction* (CSEL Rep. 95-TR-01). Columbus: Ohio State University, Cognitive Systems Engineering Laboratory.
- Schank, R. C., and Abelson, R. P. (1977). *Scripts, Plans, Goals, and Understanding*. Hillsdale, NJ: Erlbaum.
- Shaiken, H. (1985). The automated factory: The view from the shop floor. *Technol. Rev.*, January, pp. 17–25.
- Sheridan, T. B. (1976). Toward a general model of supervisory control. In *Monitoring Behavior and Supervisory Control* (T. B. Sheridan and G. Johanssen, eds.), pp. 271–281. New York: Plenum.
- Sheridan, T. B. (1992). *Telerobotics, Automation, and Human Supervisory Control*. Cambridge, MA: MIT Press.
- Sheridan, T. B., and Ferrell, W. R. (1974). *Man-machine Systems: Information, Control, and Decision Models of Human Performance*. Cambridge, MA: MIT Press.
- Smith, P. J., Layton, C., and McCoy, C. E. (1997). Brittleness in the design of cooperative problem-solving systems: The effects on user performance. *IEEE Trans. Syst., Man, Cybernet.* **27**(3), pp. 360–372.
- Smith, S., and Mosier, J. (1988). *Guidelines for Designing User Interface Software* (ESD-TR-86-278). Washington, DC: U.S. Department of Defense, Office of Management and Budget.
- Smith, S. C. (1991). Modeling the pilots and constructing an intent inferencer for a Boeing 727 cockpit. M.S. Thesis, Georgia Institute of Technology, Center for Human-Machine Systems Research, School of Industrial and Systems Engineering, Atlanta.
- Smith, S. C., Govindaraj, T., and Mitchell, C. M. (1990). Operator modeling in civil aviation. *Proc. IEEE Int. Conf. Syst., Man, Cybernet.*, Los Angeles, pp. 512–514.
- Thurman, D. A. (1995). Improving operator effectiveness in monitoring complex systems: A methodology for the design of interactive monitoring and control interfaces. M.S. Thesis, Georgia Institute of Technology, Center for Human-Machine Systems Research, School of Industrial and Systems Engineering, Atlanta.
- Thurman, D. A. (1997). The interactive monitoring and control (IMaC) design methodology: Application and empirical results. *Proc. 41st Annu. Meet. Hum. Factors Ergon. Soc.* Albuquerque, NM, pp. 289–293.
- Thurman, D. A., and Mitchell, C. M. (1994). A methodology for the design of interactive monitoring interfaces. *Proc. IEEE Int. Confe. Syst., Man, Cybernet.*, San Antonio, TX, pp. 1738–1744.
- Thurman, D. A., and Mitchell, C. M. (1995). A design methodology for operator displays of highly automated supervisory control systems. *Proc. 6th IFAC/IFIP/IFOR/SEA Symp. Anal., Des., and Eval. Man-Mach. Syst.*, Cambridge, MA, pp. 821–825.
- Turinsky, P. J., Baron, S., Burch, W. D., Corradini, M. L., Lucas, G. E., Matthews, R. B., and Uhrig, R. E. (1991). *Western European Nuclear Power Generation Research and Development* (FASAC Tech. Assess. Rep.). McLean, VA: Science Applications International Corporation.
- Uhrig, R. E., and Carter, R. J. (1993). *Instrumentation, Control, and Safety Systems of Canadian Nuclear Facilities*, JTEC/WTEC Monogr. Baltimore, MD: Loyola College, World Technology Center.
- Vanderdonckt, J. (1994). *Tools for Working with Guidelines (Bibliography)*. Belgium: University of Namur.

- Vasandani, V., and Govindaraj, T. (1994). Integration of interactive interfaces with intelligent tutoring systems: An implementation. *Mach.-Media. Learn.* 4(4), 295–333.
- Vicente, K. J., and Rasmussen, J. (1992). Ecological interface design: Theoretical foundations. *IEEE Trans. Syst., Man, Cybernet.* 22(4), 589–606.
- Wadlow, M. G. (1996). Visual interaction design. *SIGCHI Bull.* 28(1), 32–33.
- Weiner, A. J., and Mitchell, C. M. (1995). FIXIT: A case-based architecture for computationally encoding fault management experience. *Proc. 6th IFAC/IFIP/IFOR/SEA Symp. Anal., Des., Eval. Man Mach. Syst.*, Boston, pp. 196–200.
- Weiner, A. J., Thurman, D. A., and Mitchell, C. M. (1996). FIXIT: An architecture to support recognition-primed decision making in complex system fault management. *Proc. 40th Annu. Meet. Hum. Factors Ergon. Soc.*, Philadelphia, pp. 209–213.
- Wickens, C. D. (1991). *Engineering Psychology and Human Performance*. New York: Harper-Collins.
- Wiener, E. L. (1989). *Human Factors of Advanced Technology ("Glass Cockpit") Transport Aircraft* (Tech. Rep. 117528). Moffett Field, CA: NASA Ames Research Center.
- Wolfe, D. (1966). Social problems and social science. *Science* 151, 1160–1178.
- Woods, D. D. (1984). Visual momentum: A concept to improve the cognitive coupling of person and computer. *Int. J. Man-Mach. Stud.* 21, 229–244.
- Woods, D. D. (1992). Are guidelines on human-computer interaction a Faustian bargain? *Comput. Syst. Tech. Group Bull.* 19 (1), 2.
- Woods, D. D. (1995). Towards a theoretical base for representation design in the computer medium: Ecological perception and aiding human cognition. In *Global Perspectives on the Ecology of Human-machine Systems* (J. Flach, P. Hancock, J. Caird, and K. Vicente, eds.), Vol. 1, pp. 157–188. Hillsdale, NJ: Erlbaum.
- Woods, D. D., Johannesen, L. J., Cook, R. I., and Sarter, N. B. (1994). *Behind Human Error: Cognitive Systems, Computers, and Hindsight*. Wright-Patterson AFB, OH: Crew Systems Ergonomics Information Analysis Center (SOAR/CERIAC).
- Zacharias, G. L., Baron, S., and Muralidharan, R. (1981). *A Supervisory Control Model of the AAA Crew* (October 4802). Boston: Bolt, Beranek, & Newman.
- Zinser, K., and Henneman, R. L. (1988). Development and evaluation of a model of human performance in a large-scale system. *IEEE Trans. Syst., Man, Cybernet.* 8 (3), 367–375.
- Zuboff, S. (1991). *In the Age of the Smart Machine: The Future of Work and Power*. New York: Basic Books.