Designing for Flexible Interaction Between Humans and Automation: Delegation Interfaces for Supervisory Control

Christopher A. Miller, Smart Information Flow Technologies, Minneapolis, Minnesota, and **Raja Parasuraman,** George Mason University, Fairfax, Virginia

Objective: To develop a method enabling human-like, flexible supervisory control via delegation to automation. **Background:** Real-time supervisory relationships with automation are rarely as flexible as human task delegation to other humans. Flexibility in human-adaptable automation can provide important benefits, including improved situation awareness, more accurate automation usage, more balanced mental workload, increased user acceptance, and improved overall performance. **Method:** We review problems with static and adaptive (as opposed to "adaptable") automation; contrast these approaches with human-human task delegation, which can mitigate many of the problems; and revise the concept of a "level of automation" as a pattern of task-based roles and authorizations. We argue that delegation requires a shared hierarchical task model between supervisor and subordinates, used to delegate tasks at various levels, and offer instruction on performing them. A prototype implementation called Playbook® is described. **Results:** On the basis of these analyses, we propose methods for supporting human-machine delegation interactions that parallel human-human delegation in important respects. We develop an architecture for machine-based delegation systems based on the metaphor of a sports team's "playbook." Finally, we describe a prototype implementation of this architecture, with an accompanying user interface and usage scenario, for mission planning for uninhabited air vehicles. **Conclusion:** Delegation offers a viable method for flexible, multilevel human-automation interaction to enhance system performance while maintaining user workload at a manageable level. **Application:** Most applications of adaptive automation (aviation, air traffic control, robotics, process control, etc.) are potential avenues for the adaptable, delegation approach we advocate. We present an extended example for uninhabited air vehicle mission planning.

INTRODUCTION

As systems become more complex, there is often a need to control them via automation – that is, using a device, machine, or system that accomplishes, fully or partially, a function that was or could be performed by a human (Parasuraman, Sheridan, & Wickens, 2000). Examples of this trend are rife in domains ranging from aviation to health care (Billings, 1997; Degani, 2004; Parasuraman & Mouloua, 1996; Sheridan, 2002).

Automation provides clear benefits, and many complex human-machine systems cannot be operated successfully without it. Billings (1997) documented payoffs of aviation automation in four key areas: safety, reliability, economy, and comfort. But automation also poses novel problems for operators. Poorly designed automation can increase workload and training requirements, decrease situation awareness, and, in extreme circumstances, lead to accidents (e.g., Billings, 1997; Casey, 1993; Degani, 2004; Parasuraman & Riley, 1997; Reason, 1997; Sarter, Woods, & Billings, 1997; Vicente, 2003).

This "double-edged sword" of automation use (Bainbridge, 1983) has motivated repeated questions about what to automate to what level or degree for optimal control, performance, and

Address correspondence to Christopher A. Miller, Smart Information Flow Technologies, LLC, 1272 Raymond Ave., St. Paul, MN 55108; cmiller@sift.info. ***HUMAN FACTORS,*** Vol. 49, No. 1, February 2007, pp. 57–75. Copyright © 2007, Human Factors and Ergonomics Society. All rights reserved.

safety. Technologists tend to push to automate tasks as fully as possible – the "technological imperative" (Sheridan, 1987). Human factors engineers and others concerned with safety and the human role in advanced systems have tended to highlight the risks of increased automation (e.g., Perrow, 1986; Reason, 1997; Woods, 1996) and to argue against the use of higher levels of automation.

An approach to human-automation relationships that retains the benefits of automation while minimizing its costs and hazards is needed. For reasons to be discussed, we believe that such an approach requires that neither human nor automation be exclusively in charge of most tasks but, rather, uses intermediate levels of automation (LOAs) and flexibility in the role of automation during system operations and places control of that flexibility firmly in the human operator's hands. Operators need to be able to *delegate* tasks to automation, and to receive performance feedback, in much the same way that delegation is performed in successful human-human organizations: at various levels of detail and granularity and with various constraints, stipulations, contingencies, and alternatives (see Milewski & Lewis, 1999, for a review).

The use of intermediate LOAs as a way of improving human-automation performance has been proposed before (Endsley & Kiris, 1995; Wickens, Mavor, Parasuraman, & McGee, 1998). However, these previous proposals saw the choice of an intermediate LOA as a design option (for an exception, see Moray, Inagaki, & Itoh, 2000). Once an LOA was designed into a system, it could not be changed during operation. In contrast, we propose that the LOA should be adjustable *during* system operations. This is consistent with the adaptive automation concept (Inagaki, 2003; Parasuraman, Bahri, Deaton, Morrison, & Barnes, 1992; Rouse, 1988; Scerbo, 1996). However, many users may be unwilling to accept system-driven adaptation, and any such adaptive changes may increase the system's unpredictability to the user (Billings & Woods, 1994). Previous research has demonstrated the effectiveness of adaptive automation, but in most of this work the automation decides how to adapt its own behavior (Banks & Lizza, 1991; Colucci, 1995; Dornheim, 1999; Duley & Parasuraman, 1999; Hancock, Chignell & Lowenthal, 1985; Kaber & Riley, 1999; Maybury & Wahlster, 1998; Miller & Hannen, 1999; Parasuraman, 1993; Parasuraman, Mouloua, & Hilburn, 1999; Prinzel, Freeman, Scerbo, Mikulka, & Pope, 2003; Rouse, 1988; Scallen, Hancock, & Duley, 1995; Scerbo, 2001). In contrast, we propose that the human should remain in charge, deciding how much automation to use. This approach has been characterized as *adaptable* automation (Opperman, 1994; Scerbo, 2001). We believe that adaptable automation can lead to benefits similar to those of adaptive automation while avoiding many of its pitfalls.

In this paper, we propose and provide supporting evidence for a delegation approach to adaptable human-automation interaction. We first examine traditional LOA characterizations and discuss why a delegation approach demands they be extended. We then present one specific implementation of a delegation approach, based on a sports team "playbook" metaphor. (The term Playbook® as a human-automation interaction application is a registered trademark of Smart Information Flow Technologies, LLC.) We describe the major characteristics of this theoretical framework for supervisory control and outline the benefits for human and system performance.

INTERMEDIATE LOAS

Costs of Automation Extremes

The promised, and frequently realized, benefits of automation are generally sufficient to drive developers away from the lowest LOA: purely manual performance. The economic benefit of automation is a strong, but not the only, motivator. Automation also offers substantial benefits in human workload reduction, improved performance, and safety when it is properly designed and used (Billings, 1997; Bright, 1955; Lorenz, Nocera, Rottger, & Parasuraman, 2002; Parasuraman & Riley, 1997; Parasuraman et al., 2000; Reason, 1997).

The case against always automating at the highest level technologically feasible has been harder to make. Nevertheless, much research shows disadvantages from reducing the human role in system control and problem solving (Amalberti, 1999; Bainbridge, 1983; Billings, 1997; Lewis, 1998; Parasuraman & Riley, 1997; Parasuraman et al., 2000; Rasmussen, 1986; Sarter et al., 1997; Satchell, 1998; Sheridan, 1992; Wickens et al., 1998; Wiener & Curry, 1980).

High LOAs, particularly of decision-making functions, may reduce the operator's awareness of system and environmental dynamics (Endsley & Kiris, 1995; Kaber, Omal, & Endsley, 1999). Humans tend to be less aware of changes when those changes are under the control of another agent (whether automation or human) than when they make them themselves (Wickens, 1994). Mode errors also illustrate the impact of automation on awareness of system characteristics (Sarter & Woods, 1994, 1995). Another major issue influencing how effectively human operators use automation is trust (Lee & Moray, 1992, 1994). Operators may not use well-designed, reliable automation if they believe it to be untrustworthy, or they may continue to rely on automation even when it malfunctions if they are overconfident in it. Highly automated systems are prey to both sorts of errors (Parasuraman & Riley, 1997).

If higher LOAs can result in complacency and loss of awareness, it is perhaps not surprising that they can also result in skill degradation. The pilots of increasingly automated aircraft feared this effect with regard to psychomotor skills such as aircraft attitude control (Billings, 1997), but it has also been demonstrated for decision-making skills (Kaber et al., 1999).

Automation can also produce workload extremes, both low and high. That high LOAs could leave an operator bored is, perhaps, understandable, but that automation can increase workload is one of the "ironies of automation" (Bainbridge, 1983), given that many automated systems are introduced as workload-saving moves. If automation is implemented in a "clumsy" manner (e.g., if executing an automated function requires extensive data entry when operators are already very busy), workload reduction may not occur where it is most needed (Wiener, 1988). Also, if engagement of automation requires considerable "cognitive overhead" (Kirlik, 1993; Parasuraman & Riley, 1997) to evaluate the benefit of automation versus the cost of performing the task manually, then users may experience greater overall workload in using the automation.

The most significant effect of using too high a LOA may be degraded overall performance of the human-machine system. In an experiment involving aircraft route planning, Layton, Smith, and McCoy (1994) provided operators with one of three levels of support, ranging from "sketching only," in which the human sketched a desired route and the system provided feasibility feedback, to "full automation," in which a recommended best route was automatically provided. An intermediate level allowed the user to ask for a route with specific characteristics, which the system then provided if possible. Operators in the intermediate and high automation conditions were found to explore more routes because the manual sketching condition was too difficult to allow much exploration, and in the full automation condition users tended to accept the first route suggested without exploring it or its alternatives deeply. Even when they did explore, the system's recommendation tended to narrow and bias their search. Particularly in trials when the automation performed suboptimally (e.g., because it failed to adequately consider uncertainty in weather predictions), humans using the intermediate LOA produced better overall solutions.

Finally, high LOAs frequently lack user acceptance, perhaps particularly from the highly skilled operators of complex, high-criticality systems such as aircraft, military systems, and process control. For example, Miller (1999) interviewed several pilots and designers to develop a consensus list of prioritized goals for a "good" cockpit configuration manager for the Rotorcraft Pilot's Associate (Colucci, 1995), an adaptive automation system for managing information displays and cockpit functions to conform to pilot intent. Although the participants were generally enthusiastic about the technology, two of the top three items on the consensus list were "pilot remains in charge of task allocation" and "pilot remains in charge of information presented." Similarly, Vicente (1999) and Bisantz, Cohen, Gravelle, and Wilson (1996) cited examples of human interactions with even low-criticality automation, such as food planning aids in fast-food restaurants, showing that operators can become frustrated when forced to interact with automation that removes their authority to do their jobs in the best way they see fit. Vicente (1999) summarized findings from Karasek and Theorell (1990) showing that jobs in which human operators have high psychological demands coupled with low decision latitude (the ability to improvise and exploit one's skills for job performance) lead to higher incidences of heart disease, depression, pill consumption, and exhaustion.

More complete reviews of issues in human interaction with automation may be found elsewhere (Degani, 2004; Lee & See, 2004; Parasuraman &

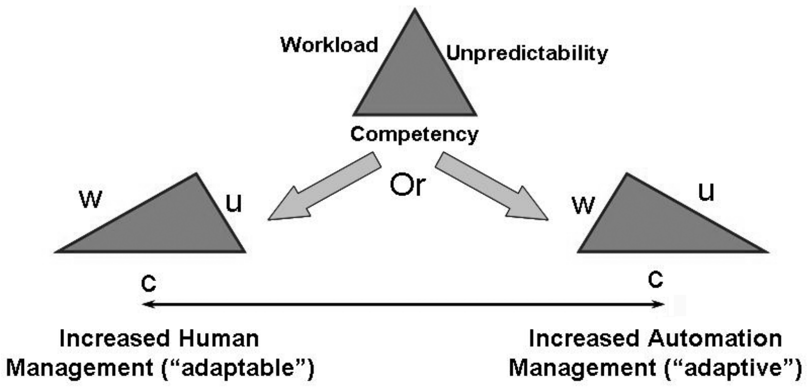
Workload
Unpredictability
Competency
Or
w
u
c
w
u
c
Increased Human Management ("adaptable")
Increased Automation Management ("adaptive")

Figure 1. The spectrum of trade-off relationships among system competency, human workload, and unpredictability. Adapted with permission from Miller, Pelican, and Goldman (2000), © IEEE.

Riley, 1997; Parasuraman et al., 2000; Sheridan, 2002; Vicente, 1999).

Trade-Off Space for Effects of LOA

The findings summarized in the previous section show that a mixture of human and automation involvement is frequently desirable, as compared with the extremes of full or no automation. Thus, human-machine systems must be designed for an appropriate relationship, allowing both parties to share responsibility, authority, and autonomy in a safe, efficient, and reliable fashion. The effects of different human-automation relationships over a task or function can be viewed as a trade-off space. Figure 1 presents a conceptual view of this space over three relevant parameters:

1. The competency of the human-machine system: *Competency* refers to correct behavior in context. Therefore, a system becomes more competent whenever it provides correct behaviors more frequently or encompasses a greater number of contexts (e.g., by making finer distinctions between contexts).
2. The mental workload for the human to interact with the system: *Workload* refers to the attentional and cognitive "energy" the human must exert to use the system (Moray, 1979; Parasuraman & Hancock, 2001; Wickens & Hollands, 2000). Mental workload for the human is modeled because it is the major constraint on the other two dimensions.
3. The unpredictability of the system to the human operator: *Unpredictability* refers to the inability of the human to know exactly what the automation will do when. Unpredictability is a consequence of the human not personally taking all actions in the system – of not being "in control" directly and immediately. Unpredictability is inversely related to the workload required to maintain awareness of the actions being taken by the system. It impacts overall situation awareness by reducing awareness of those specific aspects of the situation that pertain to the understanding of the automation behaviors and the system functions they control. Doing tasks directly costs more workload, but the payoff is greater awareness of how the task is being done. For example, while driving a car, a driver's awareness of the behavior of the car, the roads, hazards, and so forth will likely be very good, although his or her workload will be comparatively higher than if someone else were driving. Good system and interface design, improved feedback to the user about automation (Nikolic & Sarter, 2000; Parasuraman, 2000), increased reporting requirements, and good hiring and training practices can all serve to reduce unpredictability. However, in general, any form of task delegation – whether to automation or other humans – must necessarily result in added unpredictability if it offloads tasks.

The triangle in Figure 1 illustrates the relationship, or trade-off space, among these three dimensions. Other dimensions are certainly relevant to achieving competency, including operator skill, competencies and training, understanding of the domain, fatigue, and environmental stressors. We have emphasized these three dimensions as both relevant to supervisory control interactions and amenable to modification during use or interaction.

Performance to a given level of competency can be achieved only through some mix of workload and unpredictability (Milewski & Lewis, 1999). A user's workload can be reduced by allocating some functions to automation, but only at the expense of increased unpredictability (at least with regard to those functions); conversely, reducing unpredictability by having the user perform functions increases workload. It is sometimes possible to reduce both workload and unpredictability for a given level of competency through better design, corresponding to shortening the height of the triangle.

In this trade-off model, any increase in human-machine system competency must affect the human in that either (a) the added functionality must be controlled by the human, resulting in workload increases, or (b) it must be managed by automation, resulting in unpredictability increases. Opperman (1994) identified these alternatives as "adaptive" and "adaptable" approaches to automation usage, respectively (see also Scerbo, 2001) depending on whether the automation or the human determines and executes the necessary adaptations, respectively.

Another implication of the model is that these approaches are the end points of a spectrum with many alternatives in between, each representing a different trade-off among workload, unpredictability, and competency and each a different mix of human and automation roles and responsibilities. The range of alternatives may be constrained by automation and/or human capabilities, but within the feasible range, an alternative must be selected. This selection *is the process of choosing an automation "level" or relationship* – that is, of selecting who will be responsible for which tasks and what the performance constraints, authorizations, and reporting requirements will be (Parasuraman et al., 2000). When this division of labor is done by a designer prior to operation, it is part of the design for that system. When it is done by a supervisor for a human team (either prior to an operation or dynamically during it), the process may be called "delegation" or, more generally, "tasking" and task management.

Flexible LOA

Traditionally, the chief difference between task delegation as performed by a supervisor and task allocation performed by a system designer has been that the supervisor had much more flexibility in what, when, and how to delegate and better awareness of the task performance conditions. By contrast, the system designer had to fix a relationship at design time for static use in all contexts for the to-be-built system. More recently, adaptive automation (Banks & Lizza, 1991; Miller & Hannen, 1999; Parasuraman, Mouloua, & Molloy, 1996; Rouse, 1988; Scallen et al., 1995; Scerbo, 1996) has been developed, which chooses an LOA for tasks based on contextual criteria such as location, workload, experience, and physiological state. In such systems, the division of labor between human and machine is not fixed but, rather, varies under the control of the automation.

The adaptive automation concept was proposed about 30 years ago (Rouse, 1976). Only recently, however, have technologies matured to enable empirical evidence of its effectiveness to be provided. Studies have shown that adaptive systems can regulate operator workload and enhance performance while preserving the benefits of automation (Hilburn, Jorna, Byrne, & Parasuraman, 1997; Kaber & Endsley, 2004; Moray et al., 2000; Prinzel et al., 2003). The performance costs described previously of certain forms of automation – reduced situation awareness, complacency, skill degradation, and so forth – may also be mitigated (Kaber & Riley, 1999; Parasuraman, 1993; Parasuraman et al., 1996; Scallen et al., 1995), although if adaptive systems are implemented "clumsily," performance can be even worse than with fully manual systems (Parasuraman et al., 1999). For a recent review of adaptive automation research, see Inagaki (2003). Adaptive aiding systems have been successfully flight tested in the Rotorcraft Pilot's Associate (Dornheim, 1999).

Adaptive automation is truly "adaptive" in Opperman's (1994) sense because it chooses the LOA to apply. By contrast, a supervisor's task delegation within a team is more nearly "adaptable" (Cooke, Salas, Cannon-Bowers, & Stout, 2000) because the supervisor can choose which tasks to give to a subordinate, how much to dictate about how to perform subtasks, and how much attention to devote to monitoring, approving, reviewing, and correcting task performance.

This ability to delegate tasks to subordinates is clearly a powerful form of organizing and performing work. It is also deeply familiar to anyone who has worked in a team, especially in a supervisory setting. As we will describe, we have sought to provide delegation-like interactions for adaptable control over flexible automation. Before presenting that work, however, we must first define a "level of automation" and then argue for an extension to traditional definitions to

TABLE 1: Levels of Automation

1.	Human does it all
2.	Computer offers alternatives
3.	Computer narrows alternatives down to a few
4.	Computer suggests a recommended alternative
5.	Computer executes alternative if human approves
6.	Computer executes alternative; human can veto
7.	Computer executes alternative and informs human
8.	Computer executes selected alternative and informs human only if asked
9.	Computer executes selected alternative and informs human only if it decides to
10.	Computer acts entirely autonomously

After Sheridan and Verplank (1978).

support delegation-based approaches to supervisory control.

CHARACTERIZING LOAS

To discuss alternative forms of automation, a scheme for characterizing roles and responsibilities is helpful. Such characterizations have traditionally been made in terms of LOAs: defining a spectrum of possible relationships ranging from full human to full automation control.

Prior Taxonomies for LOAs

Bright (1955), perhaps the first to propose such a spectrum of LOA, viewed automation as evolving through 17 levels of competency and noted that intermediate stages might well demand greater human workload, skill, and training requirements than lower levels. Sheridan's (1987; see also Sheridan & Verplank, 1978) seminal definition of *supervisory control* included the provision that control automation allow the human to behave as if he or she were interacting with an intelligent, human subordinate – in Sheridan's (1987) words, "setting initial conditions for, intermittently adjusting and receiving information from a computer that itself closes a control loop (i.e., interconnects) through external sensors, effectors, and the task environment" (p. 1244). Sheridan and Verplank (1978) described a 10-point spectrum of automation levels in which the end points are full control autonomy for the human (essentially no role for automation) and vice versa. The intermediate levels in this spectrum, then, represent alternative forms of supervisory control interactions. A version of this list is shown in Table 1.

A problem with unidimensional models of human-automation relationships is that they are ambiguous about the application domain of the relationship. Parasuraman et al. (2000) noted that Sheridan's (Sheridan & Verplank, 1978) levels referred mainly to automation that makes decisions, offers suggestions, and/or executes actions. There are, however, other jobs automation can do: for example, sensing and analyzing information to detect situations of interest, without necessarily offering any advice on what to do with the information. Parasuraman et al. (2000) applied a simple, stage model of human information processing to arrive at four functions that must be accomplished to perform most tasks: (a) information acquisition, (b) information analysis, (c) decision and action selection, and (d) action implementation.

Because these functions can be performed by either human or automation in various mixes, in effect Parasuraman et al. (2000) added a second dimension to Sheridan's (Sheridan & Verplank, 1978) spectrum – that of the function or task the relationship is defined for. Most human-automation systems can be characterized by a mix of LOAs across these four functions, as in Figure 2. One system (System A) might be highly autonomous in information acquisition but comparatively low on the other functions, whereas a second (System B) might offer a high LOA across all four functions. For example, some flight-booking Web sites currently provide a high LOA for information acquisition – searching across multiple airlines and other Web sites – and a few also perform some information analysis (sorting flights by their cost or proximity to desired departure times). None currently provides specific recommendations, much less actually executes (autonomously) the ticket purchase, but these capabilities are easily envisioned and technologically feasible.

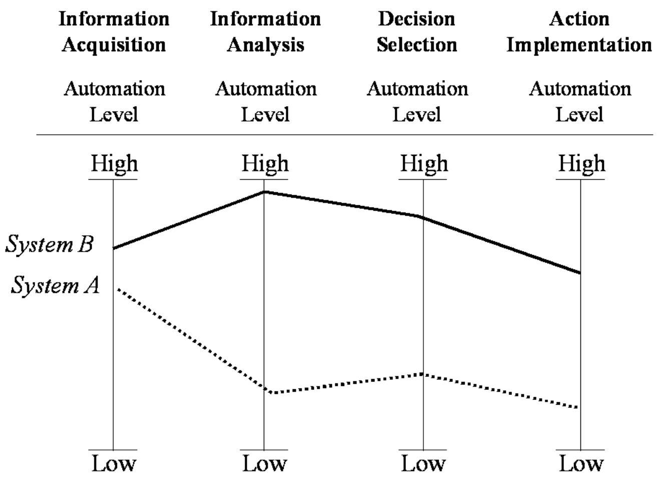

Figure 2. Levels of automation by information-processing phase for two systems. Used with permission from Parasuraman, Sheridan, and Wickens (2000), © IEEE.

Extending LOAs

An implication of the Parasuraman et al. (2000) Levels × Functions model is that a parent task can be decomposed and that a single automation level need not be applied homogenously across the subtasks. However, in this model a parent task is decomposed into abstract subfunctions based on information-processing stages, whereas other decomposition methods could provide more insight into how a task may be performed. Alternatives include (a) finer-grained functional decompositions such as operator functional modeling (Mitchell, 1987), Goals-Operations-Methods-Scripts (GOMS) models (Card, Moran, & Newell, 1983), and Plan-Goal Graphs (Geddes, 1989) that stress the subfunctions causally required to achieve a parent, and/or (b) sequential process models such as Program Evaluation and Review Technique (PERT) or Critical Path Method charts (e.g., Martinich, 1996) and Petri nets (Murata, 1989) that stress the temporal ordering and duration of subfunctions. Such decompositions are inherently hierarchical and may proceed through any number of levels to some primitive "stopping" level (Diaper, 1989) that may be imposed by biology, physics, or, more commonly, the purpose of the decomposition.

Thus, although the two-dimensional LOA model offered by Parasuraman et al. (2000) represents a major advance over earlier unidimensional models, it arguably does not go far enough. The subdivision of a parent task into four information-processing phases represents only a single level of decomposition into abstract task categories. In practice, tasks are accomplished by hierarchically decomposable sequences of specific activities: the parent task's subtasks. Automation may be applied differently to every subtask constituting the parent task. Thus, the profile of automation levels sketched in Figure 2 could stretch instead over as many subtasks and levels as one wants or needs to divide a parent task into.

In fact, the relationship between automation "level" and task decomposition is more complex than this. As is well understood (Billings, 1997; Parasuraman et al., 2000; Woods, 1996), automation does not merely shift responsibility for tasks but can change their nature as well. In a task decomposition, this means that some subtasks may be eliminated while others are added. This implies that there will generally be multiple alternate decompositions depending on, among other things, what LOA is used. Each alternative constitutes a different combination of human and automation subtasks and, thus, a different method of accomplishing the parent task.

When one identifies an LOA for a complex system using Sheridan's (Sheridan & Verplank, 1978) dimensions, one is in principle identifying something like an average or modal level over the subtasks the human-automation system accomplishes. Similarly, when one uses a Levels × Stages model such as in Parasuraman et al. (2000), one is performing an abstract and coarse-grained decomposition of the parent task into subtasks clustered by information-processing stage. Assigning levels by subtask stages offers more sensitivity than assigning them only to the parent task, but it is still an abstraction. In practice, one could identify the specific subtasks to be performed and represent an automation level for each of them.

Why would one want to? More and finer-grained subtasks are not necessarily better, and in fact Parasuraman et al. (2000) explicitly stated that they chose a four-stage model to simplify design considerations. Precision may be inherently desirable for some purposes (e.g., training and detailed design), but our interest is in supporting flexible task delegation. As we discussed, for any intermediate LOA for a task, there are roles for both humans and automation in its subtasks, yet someone must coordinate those roles. Insofar as human supervisors are required to manage, or at least be aware of, that division of labor, they must understand the decomposition of the task and of the allocation and coordination requirements among its subtasks. Supervisory control is a process of task delegation, and delegation requires task decomposition.

Task decomposition seems to reflect the way humans delegate responsibilities to subordinates and reason about task performance when receiving feedback (Klein, 1998). Hierarchically decomposed task models, which embody both functional and sequential relationships among tasks, may in fact be an accurate representation of the mental model (Gentner & Stevens, 1986) that supervisors and workers in supervisory control domains use and share (Stout, Cannon-Bowers, Salas, & Milanovich, 1999), but we are not yet making so strong a claim. Instead, we claim merely that such models form a useful structure for discussing work in a domain and, therefore, a useful vocabulary for issuing supervisory control instructions, reasoning about work to be performed, and organizing reports on the progress of such work.

In short, delegation is a process of assigning specific roles and responsibilities for the subtasks of a parent task for which the delegating agent retains authority (and responsibility). Furthermore, communication about intent to subordinates is frequently in terms of specific goals, methods, and/or constraints on how, when, and with what resources the subtasks should be accomplished (Jones & Jasek, 1997; Klein, 1998; Shattuck, 1995). For these reasons, it is essential that those subtasks be modeled explicitly if delegation interactions are to occur.

Using Extended LOA in Design

The realization that an LOA is a defined combination of roles and responsibilities between human and automation for a task to be performed, and that that task can generally be decomposed into subtasks with their own LOAs, opens the doors for more explicit communication between human and automation concerning the tasks to be accomplished. We are striving to push this communication into the run-time environment, more closely emulating delegation in human-human work relationships. Our method allows the operator to smoothly adjust the "amount" or "level" of automation used by permitting, but not requiring, instructions to automation about what specific behaviors it should perform and which behaviors should be left to the human. Supervisors could be expected to make such decisions depending on variables such as time available, workload, decision criticality, and trust.

Although this does not eliminate the trade-off between workload and unpredictability presented in Figure 1, delegation mitigates it by allowing operators to choose a point on the spectrum for their interaction with automation. In other words, the decision about how much and what kind of responsibilities to delegate must still be made, but instead of a designer making it at design time (as in traditional automation design), our approach places it in the hands of the operators at execution time, enabling them to behave more like a supervisor in human-human interactions. This strategy follows the advice of both Rasmussen, Pejtersen, and Goodstein (1994) and Vicente (1999) to allow the operator to "finish the design" opportunistically in the context of use.

It is critical that such an approach avoid two potential problems. First, it must make the task of instructing automation to behave as desired achievable without excessive workload. Second, it must ensure safe and effective overall behavior. We have created a design metaphor and a system architecture that address these concerns. We call the general class of systems that can take instruction at various, flexible levels *tasking* or *delegation interfaces* because they allow posing a task to automation as one might delegate to a knowledgeable subordinate. Examples we would label tasking interfaces can be found in Jones and Jasek (1997) and Hayes and Pande (2002). We call our particular approach Playbook because it uses the metaphor of a sports team's book of approved plays. We have conceptualized and constructed various Playbook applications to date. Implemented prototypes in the domains of premission planning for uninhabited air vehicles (UAVs), ground mobile robot control, a robot-based "capture the flag" game, and dynamic tasking of heterogeneous UAVs by dismounted soldiers are described and illustrated in Parasuraman and Miller (2006).

Next we will describe the Playbook concept as an example of a delegation interface and provide a specific, detailed example of a prototype Playbook proof-of-concept, ground-based mission planning tool for commanding UAVs. Finally, we will discuss future extensions, open questions, and anticipated payoffs of delegation interfaces in more depth.

A PLAYBOOK APPROACH TO TASKING INTERFACES

One aspect of Sheridan's (1987) formulation of supervisory control is the need for a human supervisor to teach or program the automation based on the requirements of the task and context. Tasking interfaces facilitate instructing automation by creating a shared knowledge structure of tasks that serves as the vocabulary for tasking interactions. This same knowledge can be used to improve the safety and efficacy of plans developed by allowing automation to review and critique human planning. Finally, the Playbook approach streamlines delegation by offering a compiled set of plans, or "plays," with short, easily commanded labels that can be further modified as needed.

We see three primary challenges involved in the construction of any tasking interface:

1. A shared vocabulary must be developed, via which the operator can flexibly pose tasks to the automation and the automation can report how it will perform them.
2. Sufficient knowledge must be built into the automation to enable it to make intelligent choices within the instructions provided by the user. If intelligence (including awareness) is insufficient in the subordinate system, tasking may still be possible, but the results will be unsafe, untrustworthy, and generally unacceptable.
3. One or more interfaces must be developed to permit inspection and manipulation of the vocabulary to pose and review tasks rapidly and easily.

Figure 3 presents our general architecture for tasking interfaces. Although we are still developing applications that achieve the vision we describe in this section, we have constructed various partial prototypes, all of which obey the basic structure illustrated. One of these prototypes will

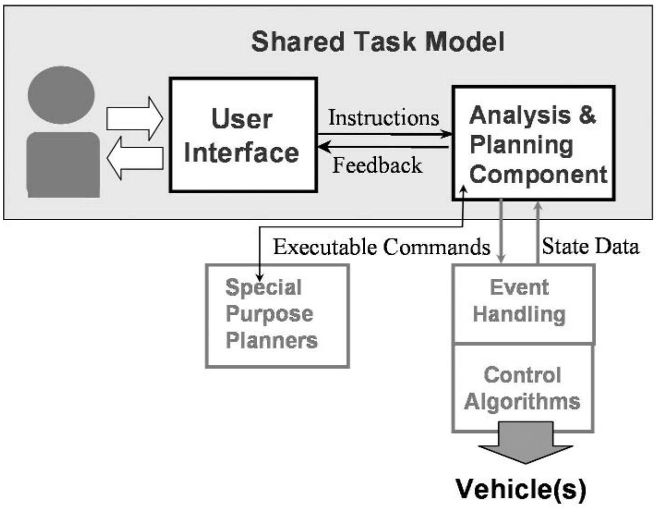

Figure 3. General tasking interface architecture. Adapted with permission from Miller, Pelican, and Goldman (2000), © IEEE.

be described in detail after a general description of the architecture and function of each component in Figure 3.

The three primary components in Figure 3 each address one of the challenges described previously. A user interface (UI) and an analysis and planning component (APC) communicate with each other and with the operator via a shared task model. Because the shared task model is the "language" in which the user and the Playbook's reasoning components interact, it is drawn as encompassing both the UI and APC. The operator communicates tasking instructions as desired goals, tasks, partial plans, or constraints via the UI, using the task structures of the shared task model. The APC is an automated planning system that understands these instructions and (a) evaluates them for feasibility and/or (b) expands them to produce fully executable plans. The APC may draw on special-purpose planning tools to perform these functions, although these are not, strictly speaking, a part of the Playbooks we have developed. Outside of the tasking interface, but essential to its use, are two additional components for making momentary adjustments during execution: an event-handling component for real-time modifications to plans and the vehicle (or other controlled element) control algorithms themselves. Because the shared task model, UI, and APC are unique to tasking interfaces, we will describe them in detail.

Shared Task Model

A critical enabling technology for tasking interfaces is the ability to represent and reason about the goals and plans operators use to talk about work to be performed. By explicitly representing these entities in a format that is familiar to a human yet interpretable by a planning system, one gains a level of coordination beyond that previously possible.

The concept of modeling user tasks is well established in human factors engineering (Diaper, 1989; Kirwan & Ainsworth, 1992). Task modeling techniques are typically comparatively informal (e.g., paper and pencil or spreadsheet based) means of capturing the steps that knowledgeable users go through to accomplish goals in known circumstances. In essence, they decompose an identified method of goal accomplishment into a series of partially ordered steps along with a procedure or script for when and how to employ the steps (Shepherd, 1989). The process can be repeated iteratively, producing a means-ends hierarchy for goal accomplishment.

To be used in a tasking interface, a task model must meet several requirements. First, it must be organized via functional decomposition: Each task or goal must be decomposable into alternate methods of achievement, down to some layer of primitive actions that are executable by the event-handling and control algorithms of the to-be-controlled system. This form of decomposition is common in task analysis but is not universal. Some approaches to using task analyses (especially those for instructional purposes) decompose a single method into progressively smaller steps without representing alternative methods of achieving the parent goal. For tasking interactions, if there are no alternative performance methods, then a flexible tasking interface is not necessary.

Second, the decomposition should be specific, representing executable methods of achieving goals rather than abstract categories such as information-processing stages (Parasuraman et al., 2000). A "task," therefore, represents an encapsulated set of behaviors that constitute a known method of accomplishing a domain goal. Tasks in the hierarchy therefore provide a vocabulary of goals with associated partial plans and leave degrees of freedom (i.e., which subtask methods to use) for the operator during execution. Thus, this approach partially overcomes criticism (Suchman, 1987; Vicente, 1999) against using static, prescriptive procedures or scripts in task-based instruction as brittle and impossible to exhaustively specify. Whereas our task hierarchies are, indeed, prescriptive, this is precisely the nature of delegation interactions: to prescribe a goal and a method of its performance. The method prescribed is never exhaustively stipulated, however, because some authority must be left to the performing agent if true delegation (and consequent workload reduction) is to occur. In our delegation approach, the task model ceases to be a static artifact around which a system is designed; instead, it becomes the language that a human supervisor uses to communicate appropriate performance methods dynamically in the context of use.

Third, tasks must be drawn from the way operators think about their domain and, ideally, should make use of the same labeling conventions with which operators are familiar. Task labels and definitions may be drawn from training materials, operator interviews, and so forth. Finally, the task model must be understandable by the automation using it. As with a football team's playbook, each player must know what to do when a given play is called. The whole task model need not be shared by all members of the team, though both understanding and flexibility will likely be facilitated by greater overlap and sharing between portions of the model, but each member must understand his or her portion, and the portions must be "linked" in that those pieces a subordinate understands and will perform when the parent play is called are, in fact, what the superior expects and intends. Coordination is, thus, an emergent function of sharing the same playbook, and complicated behaviors can be activated rapidly by use of their labels.

Our task modeling representation is based on a PERT chart-like formalism developed initially by McDonnell-Douglas for use in their pilot's associate program (Keller & Stanley, 1992). Using a representation that explicitly models the distinction between goals and plans, such as Geddes's (1989) plan-goal graph, would enhance this approach. The plan-goal distinction is a natural one for users to make in communicating intent (Klein, 1998; Shattuck, 1995).

The operator who "calls plays" must interact directly with the task model, activating and combining tasks at various levels of decomposition. This capability is provided via the user interface, which is described next. We also provide a planning system that can understand the operator's commands and either evaluate them for performability or develop an executable plan that obeys, yet fleshes them out. This analysis and planning component is described subsequently.

User Interface (UI)

Operators must interact with the task model, both to understand possible actions and to declare those tasks they wish the system to pursue. Just as a coach can activate a complex behavior via a simple play name or can spend additional time combining or modifying play elements to create new plays, so operators should be able to tune automation actions via the UI.

Some requirements for the UI are that (a) the set of "plays" (maneuvers, procedures, etc.) represented must be those any well-trained operator should know; (b) the general play "templates" must be capable of being composed and instantiated to create mission-specific plans; (c) the operator must be able to select plays at various hierarchical levels, leaving the lower levels to be composed by the APC; and (d) operators must be able to either require or prohibit the use of specific plays or resources.

Using the shared task model as an infrastructure, many interfaces are possible, each customized for its usage context. Some useful Playbook UI components will be described, and one interface, designed for ground-based UAV mission planning, will be described in depth.

Playbook UIs must include some ability to "call" predefined plays from a play library, usually at various hierarchical levels, and thereby command their performance by automation. Play calling may be more or less elaborate according to the demands of the domain. An onboard UI for fighter combat might enable literally "calling" (via a speech interface) simple play-based commands, though most other applications will benefit from more detailed communication. This can be provided minimally by allowing the operator to instantiate the parameters of a play, thereby providing constraints or stipulations on resource usage, initiation time, and so forth. More elaborately, plays/tasks may need to be composed into longer sequences (e.g., missions). A "play composition" work space and tool separate from a "play command" tool will help in these cases. Many domains will require creating new plays, either from scratch or by storing the results of earlier play composition. A different tool, or mode, should support this type of interaction. Most domains will need to visualize (and intervene in) the execution of commanded plays. Normal automation interfaces may provide these in a raw form, but referencing performance in terms of the intended plays should improve user understanding. The UI must also support interaction with the APC via issuing partial tasking instructions, receiving critiques, and previewing, accepting, and/or modifying APC-generated plans. Finally, unlike the following example, if the Playbook is to be used to monitor and retask automation during execution, then access to performance information, probably referenced against the initial, authorized plan, and ongoing access to fine-grained play-commanding capabilities will be necessary.

Interacting directly with an explicit task model meets most of these requirements, but we have found that it helps to make the UI multimodal.

Visualization of the task model shows causal and sequential relationships, but it does not do a good job of conveying the particular assets involved in each task, temporal duration of events, geographical location and progression of events, and so forth. As we will illustrate in the following example, access to auxiliary tools (such as Gantt charts and interactive map displays) can meet these needs and enhance overall usability.

Analysis and Planning Component (APC)

The APC (Goldman, Haigh, Musliner, & Pelican, 2000) operates over full or partial plans provided via the UI to (a) analyze the operator's plan for feasibility and goal achievement and (b) automatically generate candidate plan completions in keeping with the requirements and prohibitions the human imposes. The APC can critique (Guerlain, Obradovich, Rudmann, Sachs, Smith, et al., 1999) the operator-specified plan for feasibility and constraint violations, and it can weed out candidate subplays that have been made infeasible by earlier decisions. Finally, the APC can complete (i.e., develop to an executable level) a partial plan from whatever level the human chooses. The APC will either incorporate and obey human-specified portions of the plan or report that no plan can be completed within the constraints.

The APC uses a hierarchical task network planner (Erol, Hendler, & Nau, 1994) in conjunction with constraint propagation techniques (Hentenryck, 1989; Jaffar & Michaylov, 1987) to perform the functions described. Many domain-relevant task structures are known before planning begins (e.g., the general steps in a UAV mission). Rather than rediscover this outline from scratch (as in more traditional artificial intelligence planning approaches), the APC uses task knowledge from the shared task model. This both ensures that the APC's concepts of available plays mirror those of the users and makes its job of creating plans much easier. In turn, the APC must manage the resources, deadlines, and so forth, checking for feasibility and conflicts. The APC represents these limited quantities as constraints, and *feasibility* is defined as the projected plan's ability to achieve the declared goal state (i.e., accomplish the top-level task) within resource limitations. As a plan is built, constraints propagate both up, from subtasks to tasks, and down, from tasks to their expansions. This constraint propagation process enables the planner to identify flaws in the plan early.

When asked to complete a plan, the APC fleshes out nonprimitive tasks by asserting one or more subtask methods that can fill the parent goal. The APC selects its best completion according to resource usage or other optimization criteria. The UI then displays the planning decisions to the user, who can retract choices or make better-informed decisions. When asked to critique a user-defined plan, the APC again expands subtask methods to an executable level and reports whenever it discovers constraint or goal violations or the lack of any feasible, executable plan. Finally, the APC can also aid user decisions by having feasible plays at the next decomposition level be presented and infeasible ones eliminated or "grayed out," if desired.

In combination, feasibility checking and plan expansion make it possible, but not necessary, for the APC to generate effective plans with minimal user involvement. Continual feasibility analysis minimizes effort expended on dead ends while encouraging the user to specify mission-critical aspects early. Once these are stipulated, development can (but does not have to) be left entirely to the APC with the assurance that it will produce a plan that is both feasible within its constraint knowledge and in keeping with the operator's stipulations. If time permits (or lack of trust demands), the user may provide increasingly detailed instructions down to the lowest level primitive operators supported.

In fulfilling its role as a plan generator and/or plan checker, the APC operates as a type of decision-aiding or decision support system. Unlike traditional decision support systems, however, it operates at a flexible LOA, allowing operators to provide whatever level of tasking they prefer. The shared task model, and the UI that accesses it, operates as a form of programming language that the APC can understand. In this sense, it is similar to the tasking instructions that supervisors can give to subordinates: "programming" them to perform tasks according to the supervisor's intent but allowing some (flexible) degree of autonomy to accomplish that intent in the prevailing conditions at execution time. As such, the Playbook and its APC are not designed to always reduce workload. The act of providing instructions is potentially workload intensive and, as previously noted, Playbook is designed to provide operators the flexibility to decide whether or not to incur additional workload in order to reduce uncertainty.

Finally, although our early versions of Playbook, including that described later, operated primarily in premission (or "batch") planning, the Playbook concept encompasses the execution and monitoring of plans as well. Recent work (see Goldman, Miller, Wu, Funk, & Meisner, 2005; Miller, Goldman, Funk, Wu, & Pate, 2004) has begun to incorporate plan execution capabilities in Playbook. The delegation concept still applies even during execution; the time span between issuing an instruction and the subordinates' execution of it simply shrinks. Stability of control becomes an issue and may limit control authority. Current Playbook implementations have limited capability to autonomously retask assets within the commander's declared intent – for example, adjusting speed or route to maintain critical times on target. Real-time execution, monitoring, and control via Playbook raise many interesting issues that need further analysis but because of space limitations they cannot be discussed here.

Auxiliary Components for a Tasking Interface

The shared task model, UI, and APC are not sufficient for most applications. As illustrated in Figure 3, a Playbook must be integrated with a reactive planning capability for event handling and with low-level, real-time control algorithms if it is to enact plays on real-world equipment. In practice, this means that tasking interfaces leave some portion of the decisions about how to enact a plan to an intelligent agent at run time. This problem, the gap between analysis/planning and execution, is a familiar one in control and robotics. Many of our implementations have used a reactive planning architecture called *cooperative intelligent real-time control architecture* (CIRCA; Musliner, Durfee, & Shin, 1993), which integrates projective and reactive planning with control actuation. CIRCA bridges the analysis-execution gap by synthesizing logically correct reaction programs to achieve high-level goals. However, little thought had been given in CIRCA's development to how an operator would provide those goals. Our approach fills that gap.

At the bottom layer of the architecture (see Figure 3), advanced control algorithms provide for continuous vehicle control in a robotics or UAV domain. The plan created jointly by human and APC, along with reactive adaptations provided by the event-handling component, gives these control functions the instructions they need.

USAGE SCENARIO

The Playbook approach described has been partially implemented in multiple proof-of-concept demonstration systems (Parasuraman & Miller, 2006), including ones for tactical mobile robots and UAVs. None of these fully achieves the Playbook vision, but one representative UAV prototype will be described briefly here.

Traditional UAV interfaces either require operators to control the aircraft remotely via a dedicated cockpit mockup or rely on very high-level behaviors (e.g., close air patrol circuits or waypoint-designated routes) that can be easily but inflexibly commanded. Playbook provides flexibility between these points. Here, we concentrate on a ground-based tasking interface to be used for a priori mission planning. We intend this prototype, however, as an illustration of the general concept and, with appropriate interface modifications, it will be suited to dynamic, real-time, and even in-flight tasking as well. The prototype is illustrated in Figure 4.

Figure 4 shows the five primary regions of this Playbook UI as they appear after the user has interacted with it to fully develop an executable airfield denial mission. The upper half of the screen is a mission composition space that shows the plan composed thus far. The lower left corner of the interface is an available resource space, currently presenting the set of aircraft available for use. The lower right corner contains an interactive terrain map of the area of interest, used to facilitate interactions with significant geographic content. The space between these lower windows (empty at startup) is a resource in use space; once resources (UAVs, munitions, etc.) are selected, they will be moved to this work space, where they can be interacted with in more detail. Finally, the lower set of control buttons is always present for interaction. This includes options such as "finish plan" for handing the partial plan off to the APC for completion and "show schedule" for obtaining a Gantt chart timeline of the activities planned.

At startup, the mission composition space presents the three top-level plays (or mission types) the system currently knows about: interdiction, airfield denial, and suppress enemy air defenses

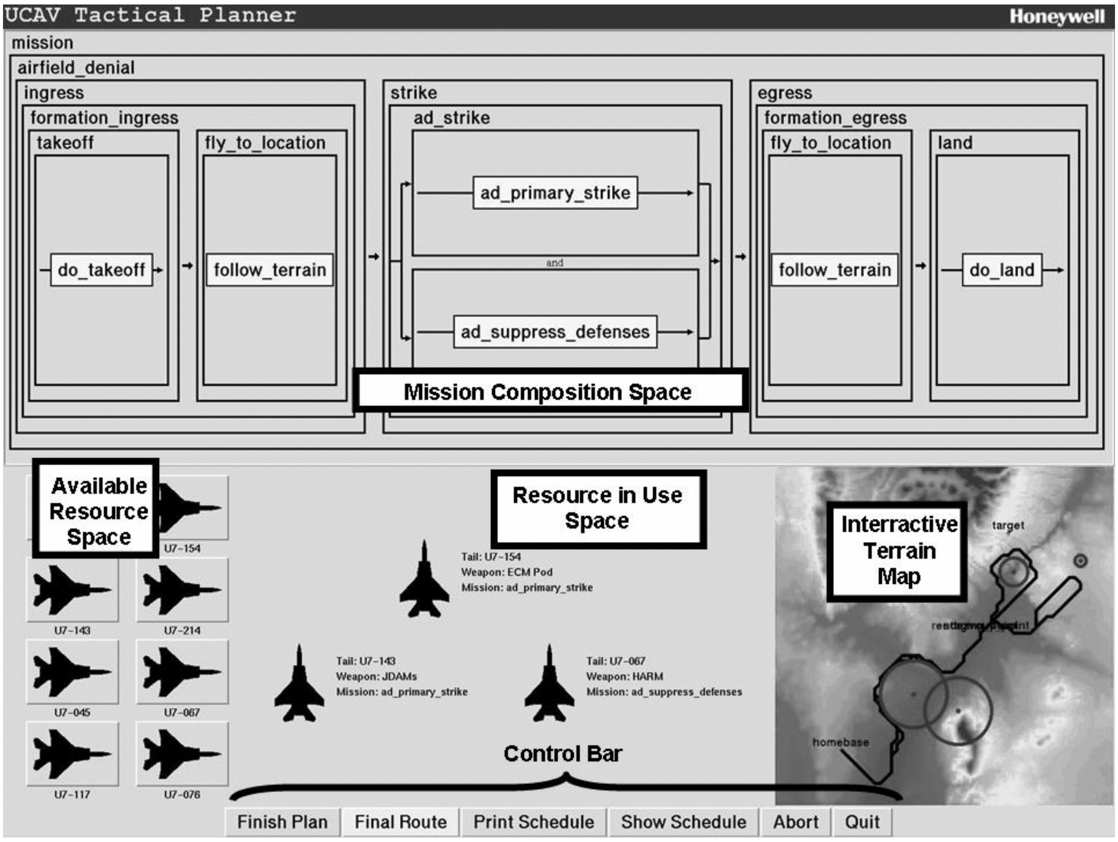

Figure 4. Prototype Playbook interface for UAV mission planning. Adapted with permission from Miller, Pelican, and Goldman (2000), © IEEE.

(SEAD). The mission leader interacts with the Playbook to first, declare that the overall mission play for the day was, say, "airfield denial." In principle, the user could define a new top-level play either by reference to existing play structures or completely from scratch, but this capability has not been designed or implemented as yet.

Clicking on "airfield denial" produces a menu with options for the user to tell the APC to "plan this task" (i.e., develop a plan to accomplish it) or to "choose airfield denial" as a task that the operator will flesh out further. The pop-up menu also contains a context-sensitive list of optional subtasks that the operator can choose to include under this task. This list is generated by the APC with reference to the structures in the play library, filtered for current feasibility.

At this point, having been told only that the task for the day is "airfield denial," a team of trained pilots would have a very good general picture of the mission they would fly. Similarly, the tasking interface (via the shared task model) knows that a typical airfield denial plan consists of ingress, strike, and egress phases and that it may also contain a suppress air defense task before or in parallel with the attack task. However, just as a leader instructing a human flight team could not leave the delegation instructions at a simple "Let's do an airfield denial mission today," so the operator of the Playbook must provide more information. Here, the user must provide four additional items: a target, a home base, a staging point, and a rendezvous point. These items are geographical in nature, and users typically find it easier to stipulate them via a map. Hence, by selecting any of them from the menu, direct interaction with the terrain map is enabled. Because the Playbook knows what task and parameter a map point is meant to indicate, appropriate semantics are preserved. The specific aircraft to be used may be selected by the user or left to the APC. If the user wishes, available aircraft can be viewed in the available resource space and chosen by clicking and moving them to the resources in use space.

The mission leader working with a team of human pilots could – if time, mission complexity, or trust made it desirable – hand the mission planning task off to the team members at this point. The Playbook operator can do this as well, handing the task to the APC via the "finish plan" button. The leader might wish, however, to provide more detailed delegation instructions. This can be accomplished by progressively interacting with the Playbook UI to provide deeper layers of task selection or to impose constraints or stipulations on the resources to be used, way points to be flown, and so forth. For example, after the user chooses "airfield denial," the system knows, via the shared task model, that this task must include an ingress subtask (as illustrated in Figure 4) and, therefore, the user does not need to tell it this. However, the user can provide detailed instructions about how to perform the ingress task by simply "drilling down" into it. Choosing "ingress" produces a "generic" ingress task template or "play." This need not be a default method of doing "ingress" but a generic, uninstantiated template, corresponding to what a human expert knows about what constitutes an ingress task and how it can be performed. A trained pilot knows that ingress can be done either in formation or in dispersed mode and must involve a "takeoff" subtask followed by one or more "fly to location" subtasks. Similarly, the Playbook user can select from available options (formation vs. dispersed ingress, altitude constraints, etc.) on context-sensitive, APC-generated menus at each level of decomposition of the task model.

The user can continue to specify and instantiate tasks down to the primitive level, where subtasks are behaviors the control algorithms (see Figure 3) can be relied upon to execute. Alternatively, at any point after the selection of the top-level task and its required parameters, the user can hand the partly developed plan over to the APC for completion and/or review. In extreme cases, a viable airfield denial plan involving multiple UAVs could be created in our prototype with as few as five selections, and more sophisticated planning capabilities could reduce this number further. If the APC is incapable of developing a viable plan within the constraints imposed (e.g., if the user has stipulated distant targets that exceed aircraft fuel supplies), it will inform the user. The prototype described here gives users limited ability to modify APC-generated plans (only the ability to reject a plan and request a new one with modified constraints), but we realize that full, fielded implementations will require richer "negotiations" about what is and is not feasible, thereby making the planning process more collaborative. We are taking the first steps toward this by enabling the APC to respond with suggested alternatives when the user's request proves infeasible (Goldman et al., 2005).

IMPLICATIONS, PRELIMINARY DATA, AND FUTURE WORK

The Playbook we have described enables a human supervisor to make use of varying LOAs for the task of planning a UAV mission. It accomplishes this flexibility by enabling explicit reference to and manipulation of the levels in a task hierarchy of domain functions. Once referenced, desired actions or constraints can be readily commanded with regard to those functions. Equally important is the presence of an intelligent planning and control environment that understands the human expressions of intent and plans execution within them.

In short, this approach allows "delegation" of the responsibility for planning and commanding a UAV mission in many of the same ways a mission commander might delegate to a team of experienced pilots. Using the task hierarchy, the operator can delegate almost complete autonomy to the automation and receive a complete plan for review. This represents LOA 8 for the overall task of mission planning in Sheridan's (Sheridan & Verplank, 1978) hierarchy (see Table 1). It also represents a trade-off among workload, uncertainty, and competency that is at the right side of the spectrum illustrated in Figure 1. Alternatively, the human can also lay out very detailed instructions, making decisions and stipulating specific actions to complete a mission plan to the primitive level. This represents LOA 2 or 1 in Table 1 and a trade-off near the left side of Figure 1.

Most importantly, the human can also interact, on a task-by-task basis, as desired or needed. If the operator wishes to plan mission ingress in detail, review computer-suggested alternatives for attack, and tell the system that anything it wishes to do for egress is acceptable as long as the UAVs return by 03:00 hr, this level of variability is supported. The resulting experienced LOA for the overall mission-planning task might be somewhere in the middle of the spectrum, but this is merely a convenient averaging. In fact, the LOA will have varied from task to task according to the needs of the operator. Thus, our motivation to decompose a task into its possible components and assign automation to each of them individually not only provides a theoretical expansion of the LOAs that previous authors have proposed but also provides the framework for intent communication required to enable complex but flexible, adaptable delegation actions.

Such interactions should result in significant payoffs. Adaptable delegation approaches may mitigate the "ironies" of automation (Bainbridge, 1983) and minimize the risks of traditional automation design, as noted previously. User-adaptable delegation approaches may allow the benefits of real-time adaptation while keeping the operator in charge. Because such an interaction style allows the user to cooperate with automation across a range of task abstraction levels, it fulfills the principle of allowing the operator to "finish the design" at the time of use. This should provide users with more control authority, address their desire to remain in charge, and ameliorate frustration, stress, and lack of user acceptance (Karasek & Theorell, 1990; Miller, 1999; Vicente, 1999). Also, providing the opportunity for true supervisory control, with all the flexibility of delegation available to human supervisors, should encourage enhanced human engagement with automation in task performance. This in turn should provide the following benefits:

- improved situation awareness relative to an interface that automatically provides a solution at a fixed LOA, at least with regard to those information elements with which the human is more directly concerned;
- improved workload relative to an interface that provides less automation support and requires more human task performance, again with regard to those tasks for which a higher LOA is used; and
- improved overall human-machine performance relative to either human alone or machine alone.

We have begun to demonstrate the feasibility of tasking interfaces in general and Playbook in particular, though much work remains. Early work on adaptive automation (Hancock et al., 1985; Parasuraman et al., 1992; Rouse, 1988) needed validation with empirical studies, which were carried out subsequently (Kaber & Riley, 1999; Parasuraman et al., 1996; for reviews, see Inagaki, 2003; Scerbo, 1996, 2001). Similarly, delegation prototypes need to be evaluated to ascertain both the usability of the specific interface design presented here and the performance benefits of the concept as a whole. We recently conducted a series of studies investigating some of the claimed benefits using a simplified delegation interface in a human-robot interaction task. Initial results (reported in detail in Parasuraman, Galster, Squire, Furukawa, & Miller, 2005) are promising. Space does not permit detailed reporting here, but these experiments, using a simplified, flexible, delegation interface for a robotic "capture the flag" game, showed that participants chose to make use of flexible LOAs in response to differing environmental conditions. Furthermore, the use of such flexibility was generally associated with the best mission success rates and the shortest mission completion times across varying environmental and "enemy" behavior conditions; these effects could be nullified when the added workload of controlling large numbers of robots conflicted with the workload of selecting among flexible interaction modes, but even in that case, performance with the delegation interfaces was as good as that with the best static or restricted UIs.

The claim that delegation interfaces do not increase workload to unmanageable levels is a critical one that requires validation. Kirlik (1993) reported that workload costs associated with deciding what LOA to employ can overwhelm any payoffs of flexible automation (see also Parasuraman & Riley, 1997). Tasking interfaces, like human-human delegation, require thought about what to delegate and what to retain. Whether human operators (as opposed to designers) can determine the best mix of tasks to delegate in various contexts is perhaps the largest open question in pursuing flexible human supervisory control. For example, Jones and Jasek (1997) reported data suggesting that at least in the domain of satellite communication planning, operators tended to consistently use the highest LOAs available, regardless of context or workload.

We have sought to create a style of interaction with automation that mirrors delegation to an intelligent assistant who knows the task domain. We have adopted a playbook organization using hierarchical task decomposition to facilitate and speed the communication of intent. This is an extraordinarily powerful form of human-human interaction and, as such, has been adopted by military, government, and commercial organizations – not to mention sports teams – throughout history. However, just as an inefficient assistant or a poorly trained teammate can require more work than doing the job oneself, so a poorly designed tasking interface may make the task more difficult. Nevertheless, we believe that the delegation concept offers a viable method for flexible, multilevel interaction between humans and automated agents in a manner that enhances system performance while maintaining user workload at a manageable level. We have illustrated one method of constructing such a delegation Playbook for rich, flexible user interactions and are beginning the process of illustrating benefits from it. Future work will continue this process of tuning and evaluating the delegation interaction format.

ACKNOWLEDGMENTS

The initial formulation of the delegation interface concept and the development of the prototype interface described here was funded by a Honeywell Laboratories Initiatives Grant. The preparation of this paper was funded, in part, by grants from Defense Advanced Research Projects Agency to R. Parasuraman and C. Miller (subcontract, Cornell University) and by Army Research Laboratory to R. Parasuraman. The authors would like to thank Dan Bugajski, Don Shaner, and John Allen for their help in the development and implementation of the ideas presented. Robert Goldman and Michael Pelican were responsible for the specific design and implementation of the example shown in Figure 4. Harry Funk and Dave Musliner participated in the initial development of these concepts. We also thank Scott Galster and Peter Squire for helpful discussions on the use of Playbook interfaces.

REFERENCES

Amalberti, R. (1999). Automation in aviation: A human factors perspective. In D. Garland, J. Wise, & V. Hopkin (Eds.), *Handbook of aviation human factors* (pp. 173–192). Mahwah, NJ: Erlbaum.

Bainbridge, L. (1983). Ironies of automation. *Automatica, 19,* 775–779.

Banks, S., & Lizza, C. (1991). Pilots' Associate: A cooperative knowledge-based system application. *IEEE Expert, 6*(3), 18–29.

Billings, C. E. (1997). *Aviation automation: The search for a human-centered approach.* Mahwah, NJ: Erlbaum.

Billings, C. E., & Woods, D. D. (1994). Concerns about adaptive automation in aviation systems. In M. Mouloua & R. Parasuraman (Eds.), *Human performance in automated systems: Current research and trends* (pp. 24–29). Hillsdale, NJ: Erlbaum.

Bisantz, A., Cohen, S., Gravelle, M., & Wilson, K. (1996). *To cook or not to cook: A case study of decision aiding in quick-service restaurant environments* (Rep. No. GIT-CS-96/03). Atlanta: Georgia Institute of Technology, College of Computing.

Bright, J. (1955). Does automation raise skill requirements? *Harvard Business Review, 36,* 85–98.

Card, S., Moran, T., & Newell, A. (1983). *The psychology of human computer interaction.* Mawah, NJ: Erlbaum.

Casey, S. (1993). *Set phasers on stun.* Santa Barbara, CA: Aegean.

Colucci, F. (1995, March–April). Rotorcraft Pilots' Associate update: The army's largest science and technology program. *Vertiflite, 41,* 16–20.

Cooke, N. J., Salas, E., Cannon-Bowers, J. A., & Stout, R. (2000). Measuring team knowledge. *Human Factors, 42,* 151–173.

Degani, A. (2004). *Taming Hal: Designing interfaces beyond 2001.* New York: Palgrave MacMillan.

Diaper, D. (1989). *Task analysis for human-computer interaction.* Chichester, UK: Ellis Horwood.

Dornheim, M. (1999, October 18). Apache tests power of new cockpit tool. *Aviation Week and Space Technology, 151,* 46–49.

Duley, J., & Parasuraman, R. (1999). Adaptive information management in future air traffic control. In M. Scerbo & M. Mouloua (Eds.), *Automation technology and human performance: Current research and trends* (pp. 86–90). Mahwah, NJ: Erlbaum.

Endsley, M., & Kiris, E. (1995). The out-of-the-loop performance problem and level of control in automation. *Human Factors, 37,* 381–394.

Erol, K., Hendler, J., & Nau, D. (1994). UMCP: A sound and complete procedure for hierarchical task network planning. In K. Hammond (Ed.), *AI planning systems: Proceedings of the 2nd International Conference* (pp. 249–254). Menlo Park, CA: AAAI Press.

Geddes, N. (1989). *Understanding human operators' intentions in complex systems.* Unpublished doctoral dissertation, Georgia Institute of Technology, Atlanta.

Gentner, D., & Stevens, A. (1986). *Mental models.* Hillsdale, NJ: Erlbaum.

Goldman, R., Haigh, K., Musliner, D., & Pelican, M. (2000). MACBeth: A multi-agent, constraint-based planner (Technical Report #WS-00-02). In *Notes of the AAAI Workshop on Constraints and AI Planning* (pp. 1–7). Menlo Park, CA: AAAI Press.

Goldman, R., Miller, C., Wu, P., Funk, H., & Meisner, J. (2005). Optimizing to satisfice: Using optimization to guide users. In *Proceedings of the American Helicopter Society's International Specialists Meeting on Unmanned Aerial Vehicles* [CD-ROM]. Alexandria, VA: American Helicopter Society.

Guerlain, S. A., Obradovich, J., Rudmann, S., Sachs, L., Smith, J. W., Smith, P. J., et al. (1999). Interactive critiquing as a form of decision support: An empirical evaluation. *Human Factors, 41,* 72–89.

Hancock, P., Chignell, M., & Lowenthal, A. (1985). An adaptive human-machine system. In *Proceedings of the 15th IEEE Conference on Systems, Man and Cybernetics* (pp. 627–629). Piscataway, NJ: IEEE Press.

Hayes, C., & Pande, A. (2002). Square-it and plan-merge: Decision support systems for high quality manufacturing setup sequences. In *Proceedings of the 2002 ASME International Design Engineering Technical Conferences and Computers & Information in Engineering Conference* [CD-ROM]. New York: American Society of Mechanical Engineering.

Hentenryck, P. (1989). *Constraint satisfaction in logic programming.* Cambridge, MA: MIT Press.

Hilburn, B., Jorna, P., Byrne, E., & Parasuraman, R. (1997). The effect of adaptive air traffic control (ATC) decision aiding on controller mental workload. In M. Mouloua & J. Koonce (Eds.), *Human-automation interaction* (pp. 84–91). Mahwah, NJ: Erlbaum.

Inagaki, T. (2003). Adaptive automation: Sharing and trading of control. In E. Hollnagel (Ed.), *Handbook of cognitive task design* (pp. 147–169). Mahwah, NJ: Erlbaum.

Jaffar, J., & Michaylov, S. (1987). Methodology and implementation of a CLP system. In *Proceedings of the 4th International Conference on Logic Programming* (pp. 196–216). Cambridge, MA: MIT Press.

Jones, P., & Jasek, C. (1997). Intelligent support for activity management (ISAM): An architecture to support distributed supervisory control. *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans, 27,* 274–288.

Kaber, D. B., & Endsley, M. (2004). The effects of level of automation and adaptive automation on human performance, situation awareness and workload in a dynamic control task. *Theoretical Issues in Ergonomics Science, 5,* 113–153.

Kaber, D. B., Omal, E., & Endsley, M. (1999). Level of automation effects on telerobot performance and human operator situation awareness and subjective workload. In M. Scerbo & M. Mouloua (Eds.), *Automation technology and human performance: Current research and trends* (pp. 165–170). Mahwah, NJ: Erlbaum.

Kaber, D. B., & Riley, J. (1999). Adaptive automation of a dynamic control task based on workload assessment through a secondary monitoring task. In M. Scerbo & M. Mouloua (Eds.), *Automation technology and human performance: Current research and trends* (pp.129–133). Mahwah, NJ: Erlbaum.

Karasek, R., & Theorell, T. (1990). *Healthy work: Stress, productivity, and the reconstruction of working life.* New York: Basic Books.

Keller, K., & Stanley, K. (1992). Domain specific software design for decision aiding. In *Proceedings of the AAAI Workshop on Automating Software Design* (pp. 86–92). Menlo Park, CA: AAAI Press.

Kirlik, A. (1993). Modeling strategic behavior in human-automation interaction: Why an "aid" can (and should) go unused. *Human Factors, 35,* 221–242.

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

Klein, G. (1998). *Sources of power: How people make decisions.* Cambridge, MA: MIT Press.

Layton, C., Smith, P., & McCoy, E. (1994). Design of a cooperative problem-solving system for en-route flight planning: An empirical evaluation. *Human Factors, 36,* 94–119.

Lee, J., & Moray, N. (1992). Trust, control strategies, and allocation of function in human-machine systems. *Ergonomics, 35,* 1243–1270.

Lee, J., & Moray, N. (1994). Trust, self-confidence, and operators' adaptation to automation. *International Journal of Human-Computer Studies, 40,* 153–184.

Lee, J. D., & See, K. A. (2004). Trust in computer technology: Designing for appropriate reliance. *Human Factors, 46,* 50–80.

Lewis, M. (1998, Summer). Designing for human-agent interaction. *Artificial Intelligence Magazine, 19,* 67–78.

Lorenz, B., Nocera, F., Rottger, S., & Parasuraman, R. (2002). Automated fault management in a simulated space flight micro-world. *Aviation, Space, and Environmental Medicine, 73,* 886–897.

Martinich, J. (1996). *Production and operations management: An applied modern approach,* New York: Wiley.

Maybury, M., & Wahlster, W. (1998). *Readings in intelligent user interfaces*. San Francisco: Morgan Kaufman.

Milewski, A., & Lewis, S. (1999). *When people delegate* (Tech. Memorandum). Murray Hill, NJ: AT&T Laboratories.

Miller, C. (1999). Bridging the information transfer gap: Measuring goodness of information fit. *Journal of Visual Language and Computation, 10,* 523–558.

Miller, C., Goldman, R., Funk, H., Wu, P., & Pate, B. (2004). A playbook approach to variable autonomy control: Application for control of multiple, heterogeneous unmanned air vehicles. In *Proceedings of FORUM 60, the Annual Meeting of the American Helicopter Society* (pp. 2146–2157). Alexandria, VA: American Helicopter Society.

Miller, C., & Hannen, M. (1999). The Rotorcraft Pilot's Associate: Design and evaluation of an intelligent user interface for cockpit information management. *Knowledge Based Systems, 12,* 443–456.

Miller, C., Pelican, M., & Goldman, R. (2000). "Tasking" interfaces to keep the operator in control. In *Proceedings of the Fifth Annual Symposium on Human Interaction with Complex Systems* (pp. 87–91). Piscataway, NJ: IEEE.

Mitchell, C. (1987). GT-MSOCC: A domain for research on HCI and decision aiding in supervisory control systems. *IEEE Transactions on Systems, Man, and Cybernetics, 17,* 553–572.

Moray, N. (1979). *Mental workload.* New York: Plenum.

Moray, N., Inagaki, T., & Itoh, M. (2000). Situation adaptive automation, trust and self-confidence in fault management of time-critical tasks. *Journal of Experimental Psychology: Applied, 6*, 44–58.

Murata, T. (1989). Petri nets: Properties, analysis and applications. *Proceedings of the IEEE, 77,* 541–580.

Musliner, D., Durfee, E., & Shin, K. (1993). CIRCA: A cooperative intelligent real-time control architecture. *IEEE Transactions on Systems, Man, and Cybernetics, 23,* 1561–1574.

Nikolic, M. I., & Sarter, N. B. (2000). Peripheral visual feedback: A powerful means of supporting attention allocation and human-automation coordination in highly dynamic data-rich environments. *Human Factors, 43,* 30–38.

Opperman, R. (1994). *Adaptive user support*. Hillsdale, NJ: Erlbaum.

Parasuraman, R. (1993). Effects of adaptive function allocation on human performance. In D. J. Garland & J. A. Wise (Eds.), *Human factors and advanced aviation technologies* (pp. 147–157). Daytona Beach, FL: Embry-Riddle Aeronautical University Press.

Parasuraman, R. (2000). Designing automation for human use: Empirical studies and quantitative models. *Ergonomics, 43,* 931–951.

Parasuraman, R., Bahri, T., Deaton, J., Morrison, J., & Barnes, M. (1992). *Theory and design of adaptive automation in aviation systems* (Progress Rep. No. NAWCADWAR-92033-60). Warminster, PA: Naval Air Warfare Center.

Parasuraman, R., Galster, S., Squire, P., Furukawa, H., & Miller, C. (2005). A flexible delegation-type interface enhances system performance in human supervision of multiple robots: Empirical studies with RoboFlag. *IEEE Systems, Man, and Cybernetics – Part A: Systems and Humans, 35,* 481–493.

Parasuraman, R., & Hancock, P. (2001). Adaptive control of workload. In P. A. Hancock & P. E. Desmond (Eds.), *Stress, workload, and fatigue* (pp. 305–320). Mahwah, NJ: Erlbaum.

Parasuraman, R., & Miller, C. A. (2006). Delegation interfaces for human supervision of multiple unmanned vehicles: Theory, experiments, and practical applications. In N. Cooke, H. Pringle, H. Pedersen, & O. Connor (Eds.), *Human factors of remotely piloted vehicles* (pp. 251–266). Amsterdam: Elsevier JAI Press.

Parasuraman, R., & Mouloua, M. (Eds.). (1996). *Automation and human performance: Theory and application.* Mahwah, NJ: Erlbaum.

Parasuraman, R., Mouloua, M., & Hilburn, B. (1999). Adaptive aiding and adaptive task allocation enhance human-machine interaction. In M. W. Scerbo & M. Mouloua (Eds.), *Automation technology and human performance: Current research and trends* (pp. 119–123). Mahwah, NJ: Erlbaum.

Parasuraman, R., Mouloua, M., & Molloy, R. (1996). Effects of adaptive task allocation on monitoring of automated systems. *Human Factors, 38,* 665–679.

Parasuraman, R., & Riley, V. (1997). Humans and automation: Use, misuse, disuse, abuse. *Human Factors, 39,* 230–253.

Parasuraman, R., Sheridan, T., & Wickens, C. (2000). A model for types and levels of human interaction with automation. *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans, 30,* 286–297.

Perrow, C. (1986). *Normal accidents: Living with high-risk technologies.* New York: Basic Books.

Prinzel, L., Freeman, F., Scerbo, M., Mikulka, P., & Pope, A. (2003). Effects of a psychophysiological system for adaptive automation on performance, workload and event-related potential P300 component. *Human Factors, 45,* 601–614.

Rasmussen, J. (1986). *Information processing and human-machine interaction.* Amsterdam: North-Holland.

Rasmussen, J., Pejtersen, A., & Goodstein, L. (1994). *Cognitive systems engineering*. New York: Wiley.

Reason, J. (1997). *Managing the risks of organizational accidents.* Aldershot, UK: Ashgate.

Rouse, W. (1976). Adaptive allocation of decision making responsibility between supervisor and computer. In T. B. Sheridan & G. Johannsen (Eds.), *Monitoring behavior and supervisory control* (pp. 221–230). New York: Plenum Press.

Rouse, W. (1988). Adaptive automation for human/computer control. *Human Factors, 30,* 431–488.

Sarter, N., & Woods, D. (1994). Pilot interaction with cockpit automation: II. An experimental study of pilots' model and awareness of the flight management system. *International Journal of Aviation Psychology, 4,* 1–28.

Sarter, N., & Woods, D. (1995). How in the world did we ever get into that mode? Mode error and awareness in supervisory control. *Human Factors, 37,* 5–19.

Sarter, N., Woods, D., & Billings, C. (1997). Automation surprises. In G. Salvendy (Ed.), *Handbook of human factors and ergonomics* (2nd ed., pp. 1926–1943). New York: Wiley.

Satchell, P. (1998). *Innovation and automation.* Aldershot, UK: Ashgate.

Scallen, S., Hancock, P., & Duley, J. (1995). Pilot performance and preference for short cycles of automation in adaptive function allocation. *Applied Ergonomics, 26,* 397–403.

Scerbo, M. (1996). Theoretical perspectives on adaptive automation. In R. Parasuraman & M. Mouloua (Eds.), *Automation and human performance: Theory and applications* (pp. 37–63). Mahwah, NJ: Erlbaum.

Scerbo, M. (2001). Adaptive automation. In W. Karwowski (Ed.), *International encyclopedia of ergonomics and human factors* (pp. 1077–1079). London: Taylor & Francis.

Shattuck, L. (1995). *Communication of intent in distributed supervisory control systems.* Unpublished doctoral dissertation, Ohio State University, Columbus.

Shepherd, A. (1989). Analysis and training in information technology tasks. In D. Diaper (Ed.), *Task analysis for human-computer interaction* (pp. 15–55). Chichester, UK: Ellis Horwood.

Sheridan, T. (1987). Supervisory control. In G. Salvendy (Ed.), *Handbook of human factors* (pp. 1244–1268). New York: Wiley.

Sheridan, T. (1992). *Telerobotics, automation, and supervisory control.* Cambridge, MA: MIT Press.

Sheridan, T. (2002). *Humans and automation: System design and research issues.* Santa Monica, CA: Human Factors and Ergonomics Society/New York: Wiley.

Sheridan, T., & Verplank, W. (1978). *Human and computer control of undersea teleoperators* (Tech. Rep.). Cambridge: Massachusetts Institute of Technology, Man-Machine Systems Laboratory.

Stout, R. J., Cannon-Bowers, J. A., Salas, E., & Milanovich, D. M. (1999). Planning, shared mental models, and coordinated performance: An empirical link is established. *Human Factors, 41,* 61–71.

Suchman, L. (1987). *Plans and situated actions: The problem of human machine communication*. Cambridge, UK: Cambridge University Press.

Vicente, K. (1999). *Cognitive work analysis: Towards safe, productive, and healthy computer-based work.* Mahwah, NJ: Erlbaum.

Vicente, K. (2003). *The human factor.* New York: Alfred Knopf.

Wickens, C. D. (1994). Designing for situation awareness and trust in automation. In *Proceedings of the International Federation of Automatic Control (IFAC) Conference* (pp. 174–179). Baden-Baden, Germany: IFAC.

Wickens, C. D., & Hollands, J. G. (2000). *Engineering psychology and human performance.* New York: Longman.

Wickens, C. D., Mavor, A., Parasuraman, R., & McGee, J. (1998). *The future of air traffic control: Human operators and automation.* Washington DC: National Academy Press.

Wiener, E. (1988). Cockpit automation. In E. Wiener & D. Nagel (Eds.), *Human factors in aviation* (pp. 433–461). San Diego, CA: Academic.

Wiener, E., & Curry, R. (1980). Flight-deck automation: Promises and problems. *Ergonomics, 23,* 995–1011.

Woods, D. (1996). Decomposing automation: Apparent simplicity, real complexity. In R. Parasuraman, M. Mouloua, & R. Molloy (Eds.), *Automaton and human performance: Theory and applications* (pp. 3–17). Mahwah, NJ: Erlbaum.

Christopher A. Miller is the chief scientist of Smart Information Flow Technologies, LLC. He received his Ph.D. in psychology from the Committee on Cognition and Communication in the Psychology Department of the University of Chicago in 1991.

Raja Parasuraman is a professor of psychology and a member of the Arch Lab at George Mason University. He received his Ph.D. in psychology from Aston University, UK, in 1976.

Date received: January 2, 2005
Date accepted: November 4, 2005