Submitted for presentation at the Fifth Annual Symposium on Human Interaction with Complex Systems, April 30-May 2, 2000, Urbana-Champaign, Ill.

"TASKING" INTERFACES TO KEEP THE OPERATOR IN CONTROL

Christopher A. Miller, Michael Pelican , Robert Goldman
Honeywell Technology Center
3660 Technology Dr.
Minneapolis, MN 55418 U.S.A.

ABSTRACT

The ongoing debate in the HCI community between direct manipulation and intelligent, automated agents points to a fundamental problem in complex systems. Humans want to remain in charge even if they don't want to (or can't) make every action and decision themselves. We have been exploring a middle road through "tasking interfaces"—interfaces which share a task model with a projective planner to enable human operators to flexibly "call plays" (i.e., stipulate plans) to various levels of abstraction, leaving the remainder to be fleshed out by the planner. The result is akin to 'tasking' a knowledgeable subordinate with more or less detailed instructions. We describe a prototype tasking interface for Uninhabited Combat Air Vehicles (UCAVs).

INTRODUCTION

As systems become more complex, the temptation is to control them via "automation" (e.g., Billings, 1997)—either with systems which fully perform the task, or with 'decision aids' which provide guidance but leave final execution to humans. In spite of extensive technological achievements in automation, advanced automation suffers from a basic sociological problem. Human operators want to remain in charge. For example, in developing the Rotorcraft Pilot's Associate Cockpit Information Manager (Miller & Funk, 1997), we asked multiple pilots and designers to develop a consensus list of goals for a "good" cockpit configuration manager. Two of the top three items on the list were "Pilot remains in charge of task allocation" and "Pilot remains in charge of information presented."

There are good reasons to design systems at the higher levels of automation. By definition (Sheridan, 1987), such systems share responsibility, authority and autonomy over many work behaviors with human operator(s) to accomplish their goals of reducing workload and information overload. While operators may wish to remain in charge, today's complex systems no longer permit them to be fully in charge of all system operations—at least not in the same way as in earlier cockpits and workstations.

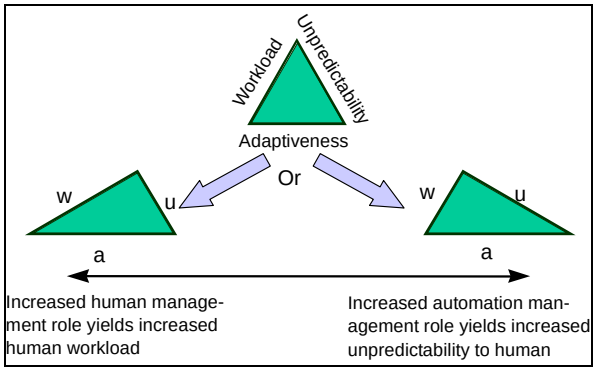

Figure 1a. Conceptual relationship between system adaptiveness, human workload and unpredictability.

The problem is shown conceptually in Figure 1a. The relationship between the adaptiveness of a human-machine system is a function of the workload and unpredictability it causes the operator. This implies that for any increase in adaptiveness (the ability of to perform in an appropriate, context-dependent manner across situations) there must be an accompanying increase in one or both of the other legs of the triangle. Either human workload or unpredictability (the human's inability to know what the automation will do) must increase. Since adaptiveness is generally the goal of added complexity (though systems can be complex without achieving it), this is equivalent to saying that any increase in system complexity must affect the operator in two ways—either (1) the added complexity must be controlled by the human, resulting in increases in workload, or (2) the added complexity must be managed by automation, resulting in increases in unpredictability. The ongoing debate (Maes, 1994; Schneiderman, 1997) in the HCI design community over intelligent agents vs. direct manipulation interfaces is a manifestation of these alternatives.

In recent work developing an interface for Uninhabited Combat Air Vehicles (UCAVs), we have explored a middle road between 'full' human control and delegation to automation. In brief, our solution is to allow human operators to interact with advanced automation *flexibly* at a

variety of levels. This allows the human operator to smoothly vary the 'amount' of automation s/he uses depending on such variables as time available, workload, criticality of the decision, degree of trust, etc.—variables known to influence human trust and accuracy in automation use (Riley, 1996). It also allows the human to flexibly act within the limitations imposed by the capabilities and constraints of the world—a strategy shown to produce superior aviation plans and superior human understanding of plan considerations in (Layton, Smith & McCoy, 1994) and which lies at the heart of recent advances in ecological interface design (Vicente, 1996). While this does not eliminate the dilemma presented in Figure 1a, it mitigates it by allowing operators to choose various points on the spectrum for interaction with automation (see Figure 1b). We call human-machine systems of this sort "tasking" interfaces, because they allow posing a task to automation at all the different levels one might 'task' a knowledgeable subordinate.

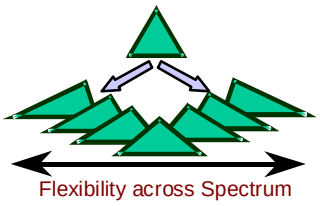

Figure 1b. A 'Tasking Interface' provides multiple methods of increasing adaptivenss.

PROPOSED SOLUTION— TASKING INTERFACES: REQUIREMENTS AND ARCHITECTURE

There are three primary challenges involved in the construction of a tasking interface:

(1) A shared vocabulary must be developed, through which the operator can flexibly pose tasks to the automation and the automation can report how it intends to perform those tasks.
(2) Sufficient knowledge must be built into the interface to enable making intelligent choices within the tasking constraints imposed by the user.
(3) One or more interfaces must be developed which will permit inspection and manipulation of the tasking vocabulary to pose tasks and review task elaborations in a rapid and easy fashion.

Figure 2 presents our general architecture for tasking interfaces. The three primary components each address one of the challenges described above. In our approach, a Graphical User Interface (GUI) in the form of a "playbook" and a Mission Analysis Component (MAC) are based on and communicate with each other and with the

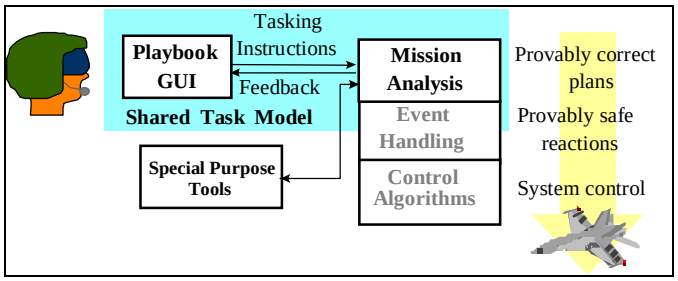

Figure 2. General Architecture for Tasking Interfaces.

human operator via a Shared Task Model. The human operator communicates tasking instructions in the form of desired goals, tasks, partial plans or constraints, via the Playbook GUI, in accordance with the task structures defined in the shared task model. These are, in fact, the methods used to communicate commander's intent in current training approaches for U.S. battalion level commanders (Shattuck, 1995). The MAC is a *projective* planning system capable of understanding these instructions and (a) evaluating them for feasibility or b) expanding them to produce fully executable plans. The MAC may draw on special purpose planning tools to perform these functions, wrapping them in the task-sensitive environment of the tasking interface as a whole. Outside of the tasking interface itself, but essential to its use, are two additional components. Once an acceptable plan is created, it is passed to an Event Handling component, which is a *reactive* planning system capable of making moment by moment adjustments to the plan during execution. The Event Handling component then passes these instructions to control algorithms that actually effect behaviors in the controlled system automation.

USAGE SCENARIO

Additional description of these components is beyond the scope of this paper; see Miller & Goldman, 1997 and Miller, Pelican & Goldman, 1998 for more details. Instead, we will focus on an illustration of the human interaction with one implementation of a tasking interface.

The following scenario illustrates how a user interacts with the tasking interface prototype we have developed to plan a UCAV mission. Current and emerging UCAV interfaces either require operators to remotely control the aircraft via a dedicated cockpit mockup, or they rely on very high level behaviors (e.g., Close Air Patrol circuits or waypoint-designated routes) which can be easily but inflexibly commanded. The first approach provides high predictability and adaptiveness, but at the cost of very high operator workload; the second approach minimizes operator workload and provides highly predictable behavior, but only at the cost of adaptiveness. Neither approach

is sufficient for usefully placing one or more UCAVs at the disposal of an operator who is concurrently piloting his/her own aircraft.

Building on prior Honeywell control algorithms and simulation work supporting scenarios of multiple uninhabited F-16s, we have developed a tasking interface to enable a human leader to lay out a mission plan for the UCAV's. This interface will support the stipulation of full and partial plans and constraints for the UCAVs either separately or in conjunction. To date, we have concentrated on a ground-based tasking interface due to its lighter demands on user, simulation and interface design. However, we believe that with suitable GUI modifications, this approach will be suited to in-flight tasking as well.

Our prototype tasking interface as illustrated in Figures 3-7. The interface illustrated in these figures is our current Playbook GUI. This interface runs in conjunction with a prototype MAC, communicating via a Shared Task Model. The MAC interacts with a specialty route planning algorithm. Plans produced by this tasking interface can be 'flown' in simulation using realistic control algorithms developed by Honeywell's Guidance and Control group.

Figure 3 (and, in larger format, Figure 7) shows the five primary regions of the Playbook GUI. The upper half of the screen is a Task Composition Space that the plan composed thus far. At startup, this area presents the three top-level tasks (or 'mission types') the system currently knows about: Interdiction, Airfield Denial, and Suppress Enemy Air Defenses (SEAD). 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. As noted above, some planning interactions can be more easily performed or understood via direct, graphical presentation against the terrain of interest. The space between these two lower windows (empty at startup) is a Resource in Use Space—once resources are selected, they will be moved to this space, where they can be interacted with in more detail. Finally, the lower set of control buttons is always present. This includes options such as "Finish Plan" for handing the partial plan off to the MAC for completion and/or review, "Final Route" for accessing the route planner to create a route within constraints specified, "Print" and "Show Schedule" for obtaining a Gantt chart timeline of the activities planned for each actor, etc.

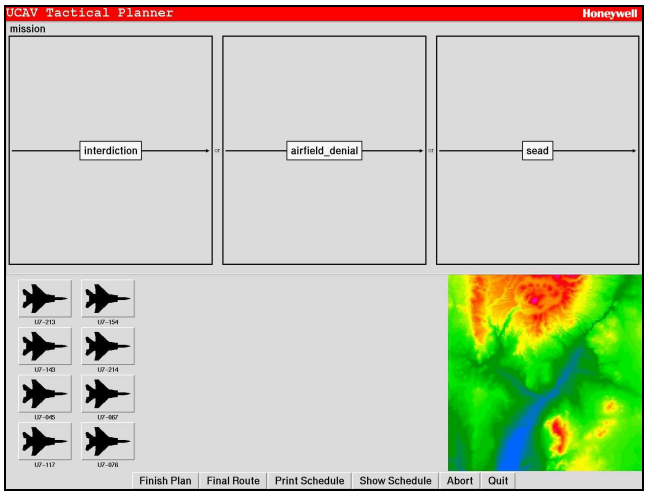

Figure 3. The Tasking Interface prototype at startup.

The mission leader would interact with the tasking interface to, first, declare that the "task" for the day was "Airfield Denial." Most interactions with the Playbook GUI are via mouse buttons. In this case, the three top-level mission task boxes have "or" relationships between them, meaning the pilot must select one. Figure 4 shows the pop-up window when the user clicks on "Airfield Denial."

From this pop-up menu, the human "tasker" can tell the MAC to "Plan this Task" (that is, develop a plan using this task) or indicate that s/he will "Choose airfield denial" as a task that s/he will flesh out further. The pop-up menu also contains a context-sensitive list of subtasks that the tasker can choose to include under this task (only one is appropriate at this level: "Suppress Air Defenses").

At this point, having been told only that the task for the day is "Airfield Denial," a team of trained human pilots would have a very good general picture of the mission they would fly. Similarly, our interface (via the MAC and the Shared Task Representation) knows what a typical airfield denial plan consists of. But just as a leader instructing a human flight team could not leave the instructions at that, so the tasker is required to provide more in-

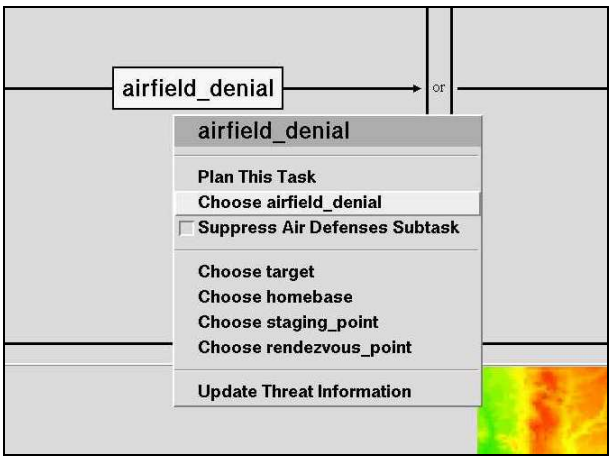

Figure 4. Pop-up window for "Airfield Denial".

formation to instantiate the high level task. Here, the tasker must provide the items listed in the next block in the pop-up window in Figure 4 (e.g., a target, a homebase, etc.) and, as for all plans, the specific aircraft to be used. S/he selects aircraft by clicking on the available aircraft resources and moving them to the Resources in Use Area. By clicking on "Choose Target", the user activates the Terrain Map and a subsequent click there will designate a target location.

The human leader could provide substantially more detail (such as route, munitions, roles for wingmen, etc.) but s/he could also hand the task off to at this point and let a human team develop the plan. The tasker can also do this, handing the task to the MAC via the "Finish Plan" button. We will assume that the user provides more plan specifications. Both the tasker and the interface know, thanks to their Shared Task Model, that any Airfield Denial plan must consist of Ingress, Strike and Egress subtasks, in that order. "Airfield Denial" may also include a Defense Suppression subtask that runs in parallel with Strike. After the tasker selects "Choose Airfield Denial", the Task Composition Space is reconfigured as in Figure 5 to show the next level of options. The small arrows between the task blocks indicate that all subtasks are sequential.

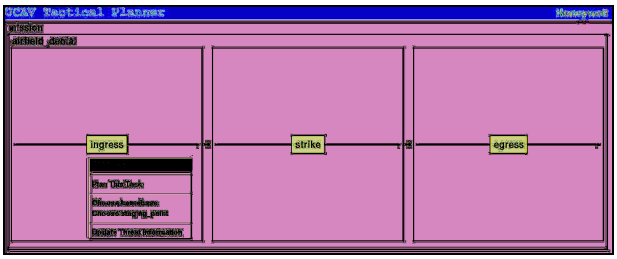

Figure 5. Second-level expansion of 'Airfield De-

To provide detailed instructions about how to perform the Ingress task, the tasker must choose it, producing the "generic" Ingress task shown in Figure 6. Note that this is not a default method of doing "Ingress" so much as a generic, uninstantiated template—roughly what a human expert knows about how Ingress can or should be performed. A trained pilot knows that Ingress can be done either in formation or in dispersed mode and, in either case, must involve a "Take Off" subtask followed by one or more "Fly to Location" subtasks. Figure 6 illustrates the GUI after the user selects "Formation Ingress". The MAC automatically fills in the set of steps it knows must be accomplished to do Ingress in formation— that all aircraft must takeoff and then fly to a specified location. In the event that there are further specifications the user can or must supply, the MAC will generate them in context-sensitive pop-up windows (e.g., specifying the airfield to take off from.)

The user can continue to specify and instantiate tasks down to the "primitive" level where the subtasks are behaviors the control algorithms (see Figure 2) in our simu-

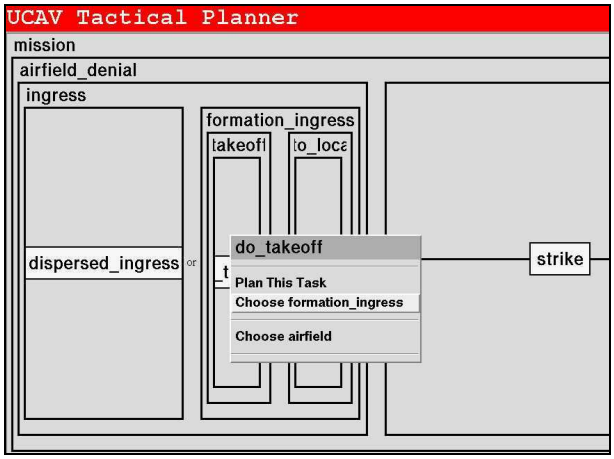

Figure 6. Stipulating subtasks under "Ingress".

lator can be relied upon to execute in flight. Alternatively, at any point after the initial selection of mission task and its required parameters, the tasker can hand the partly developed plan over to the MAC for completion and/or review by means of the "Finish Plan" button. In extreme cases, a viable "Airfield Denial" plan could be created with as few as five choices (select aircraft, target, homebase, staging and rendezvous points). If the MAC 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) MAC will inform the user of these difficulties via pop-up windows.

Figure 7 shows the completed plan provided by the MAC after the stipulation of the Ingress task as described above. The user has asked for a Final Route generated by the route planner on the basis of the MAC's inputs. This output is presented in the Terrain Map window.

LESSONS AND CONCLUSIONS

Our prototype interface builds plans for 3 types of multi-UCAV missions (corresponding to 3 top-level tasks). Other types of interfaces are entirely possible (see Miller & Goldman, 1997, for a more ambitious, though unimplemented example) and we hope to explore their development as we carry the tasking interface concept into alternate domains and gain more experience with user interactions with this type of automation. Our approach to tasking interfaces builds a bridge of flexibility between 'pure' direct manipulation interfaces where the user is always in charge but must incur the associated workload,

and 'pure' intelligent agent automation where the user is free from undue workload but at the cost of surrendering control and predictability to a system that may not do what s/he wants. While operators of complex systems are rarely comfortable giving over authority to automation at all times, they are quite willing to use it when helpful. Tasking interfaces are a method of allowing the operator to remain fully in charge, yet of enabling almost full autonomy for an aiding agent. Perhaps by requiring (and enabling) automation to behave more like an intelligent subordinate, operators will be more tolerant of its weaknesses and more willing to let it show its capabilities in some settings. Through use, then, users may become more familiar with their automation's strengths and weaknesses and, ultimately, better at using it when it is needed.

ACKNOWLEDGEMENTS

This work was funded by a Honeywell Initiatives Grant. The authors would like to that Dan Bugajski, Don Shaner and John Allen for their help in the development and implementation of the ideas presented.

REFERENCES

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

Layton, C., Smith, P, and McCoy, E. (1994). Design of a cooperative problem solving system for enroute flight planning: An empirical evaluation. *Human Factors, 36*(1). 94-119.

Maes, P. (1994). Agents that Reduce Work and Information Overload. *Communications of the ACM, 37*(7), 31-40.

Miller, C. & Funk, H. (1997). Task-based Interface Management: A Rotorcraft Pilot's Associate Example. *Proceedings of the AHS Crew Systems Technical Specialists Meeting*, (Philadelphia PA, September).

Miller, C. & Goldman, R. (1997). HEC paper

Miller, C., Pelican, M., and Goldman, R. (1999). "Tasking" Interfaces for Flexible Interaction with Automation: Keeping the Operator in Control. In *Proceedings of the International Conference on Intelligent User Interfaces.* Redondo Beach, CA: January 5-8.

Riley, V. (1996). Operator reliance on automation: Theory and data. In R. Parasuraman and M. Mouloua (Eds.), *Automation and Human Performance: Current Theory and Applications.* Lawrence Erlbaum: Hillsdale, NJ, 19-36.

Shattuck, L. (1995). Communication of Intent in Distributed *Supervisory Control Systems*. Unpublished dissertation. The Ohio State University, Columbus, OH..

Sheridan, T. Supervisory Control. (1987). In G. Salvendy (Ed.), *Handbook of Human Factors*. John Wiley & Sons, New York. 1244-1268.

Shneiderman, B. (1997). Direct manipulation for comprehensible, predicatable, and controllable user interfaces, Proceedings of the ACM International Workshop on Intelligent User Interfaces '97, New York, NY, 33-39.

Vicente, K. J. (1996). Improving dynamic decision making in complex systems through ecological interface design: A research overview. System Dynamics Review, 12. 251-279.

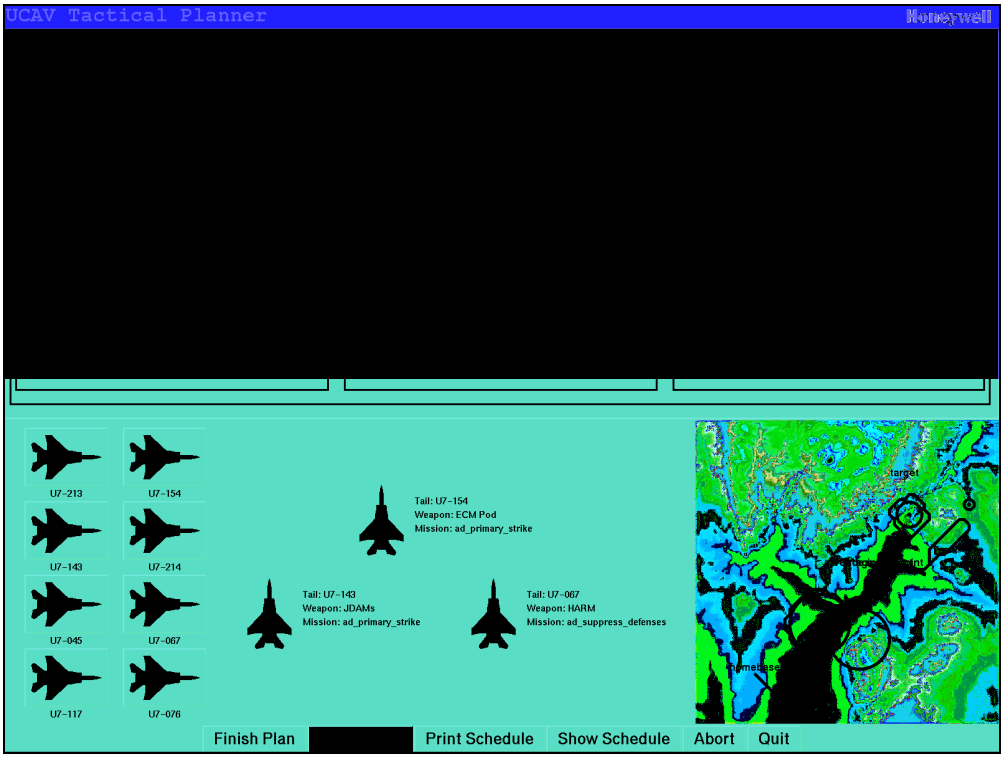

Figure 7. Finished plan returned by the MAC after User 'calls' the Ingress task.