A Playbook™ for Real-Time, Closed-Loop Control

Harry Funk, Robert Goldman, Christopher Miller, John Meisner, Peggy Wu
Smart Information Flow Technologies, LLC
211 First St. North, Ste. 300
Minneapolis, MN 55108
+1 612-339-7438
{hfunk, rpgoldman, cmiller, jmeisner, pwu}@sift.info

ABSTRACT

SIFT has been developing an approach to adaptable automation control of multiple Unmanned Air Vehicles (UAVs) we call a Playbook™ because it is based on the metaphor of a sports team's book of acceptable plays. Playbook represents a "delegation" approach to human-automation interactions because it allows a human operator to task or delegate authority to automation with much of the same flexibility with which a human supervisor or team captain can delegate objectives, methods, constraints and even detailed instructions to subordinates. In previous work, we have described the Playbook architecture and approach, illustrated interfaces, presented initial data clarifying its benefits and describing collaborative interactions with it. Here, we review the Playbook concept and previous work and then report on newly implemented play capabilities including the ability for the Playbook to coordinate the activities of multiple UAVs to track a ground-based moving target—a play that requires Playbook to dynamically replan within authority delegated to it by the human operator.

Categories and Subject Descriptors

H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—*Playbook™*; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods and Search--*Control theory, Graph and tree search strategies, Plan execution, formation, , and generation, Hierarchical Task Networks;* I.2.9 [**Artificial Intelligence**]: Robotics—*Autonomous Vehicles, Operator Interfaces*.

General Terms

Human Factors.

Keywords

Delegation Interfaces, Tasking Interfaces, Unmanned Air Vehicles, Robotics, Hierarchical Task Networks, Mixed Initiative Control, Discrete Event Control.

Human Robot Interaction'06, March 2-3, 2006, Salt Lake City, Utah USA.

1. INTRODUCTION

Human interaction with complex automation (including robots) poses myriad challenges. Not the least of these is "tuning" automation performance so that a satisfactory, safe and effective mix of human and machine roles results. On the one hand, the "technological imperative" [1] argues for ever-increasing delegation of roles and performance duties to automation in order to reduce the costs (in terms of workload, training, person-hours, boredom and, in some cases, physical safety) to human operators. On the other hand, there are now well-understood drawbacks to the over-use of automation, especially when that automation operates in a less-than-perfect manner and/or is implemented in such a fashion that its use will be "clumsy" for the humans that must engage it. Some of these drawbacks include [2,3] unbalanced workload, loss of skills, over- and under-trust, lack of user acceptance and reduced overall system performance and safety.

Human interaction with Unmanned Air Vehicles (UAVs) and other semi-autonomous vehicles such as robots present these problems in novel, yet familiar ways. The technological imperative has manifested itself initially in the drive to develop and make extended use of UAVs in roles that humans either cannot perform, or cannot perform as safely, conveniently or efficiently as an unmanned vehicle can. Currently, the imperative manifests itself in a drive to reduced operator-to-platform ratios and divorce ourselves from the current practice of providing a dedicated workstation for each vehicle or vehicle type and to allow operators to control multiple, heterogeneous vehicle types with minimal workstation modifications and additional training.

Yet insofar as the UAV (or robotic vehicle in any form) is to serve the intent of a human supervisor, human interaction with it remains crucial and, thus, convenient and efficient methods of communicating that intent remain critical. The core challenge is to develop a method of interacting with sophisticated, variably-autonomous agents that allows the human supervisor to provide as much or as little instruction and constraint on the performance and behavior of the agent as the human deems necessary and desirable. We believe that such methods of interaction have already been largely developed for cases where the "agent" receiving instruction happens to be another human in a subordinate role. In these cases, we say that the supervisor "delegates" tasks, roles, responsibilities and authority to the subordinate via a more or less complex set of delegation instructions. Our challenge is to make such delegation interactions feasible for human-machine interac-

tions with at least the degree of flexibility and effectiveness as they are for human-human interactions.

We are developing an approach to UAV control that addresses this challenge [3-6]. We use the metaphor of a sports team's playbook to enable both quick and complex variable-initiative control with a variety of UAVs. The user of our Playbook™ "calls a play" to request a service from a team of UAVs. We have implemented this approach in the Playbook-enhanced Variable Autonomy Control System (PVACS) project [5], which combines SIFT's Playbook interfaces and Geneva Aerospace's Variable Autonomy Control System (VACS).

In this paper, we describe the general Playbook concept and present the architecture of our PVACS Playbook. We describe example "plays" that have been implemented previously, along with examples of the user interfaces for interacting with a Playbook to "call" them, and to review and monitor execution of the resulting UAV plan. Finally, we describe a recently implemented play to coordinate multiple UAVs in the tracking of a ground-based moving target. Since this play has required outer loop, discrete-event control in order to manage assets involved in the tracking task, it represents a step toward full, dynamic replanning control not present in earlier versions of our Playbook.

2. A PLAYBOOK™ APPROACH TO HUMAN-AUTOMATION INTERACTION

SIFT has pioneered a human-automation integration architecture, called Playbook™ [3-6] based on a domain-specific task model which is shared by human operators and by planning and control automation—and which serves as a "lingua franca" between them. The goal of Playbook is to enable the same degrees of flexibility in commanding automation (specifically, though not exclusively, unmanned vehicles) that a human supervisor has with knowledgeable, competent subordinates. Supervisors (or team captains) interactING with human subordinates can decide how much and what kind of instruction and constraints or stipulations to impose on their subordinates as a function of factors such as the time and workload capacity available, the supervisor's trust in the subordinate, and the specific constraints of the current context. When necessary or desirable, they can provide very high level and minimal instruction about the objectives and methods the subordinate should be pursuing (can "call a play") and leave most of the planning, decision making and execution responsibilities to the subordinate—albeit with less certainty about exactly what methods will be used. Alternatively, again when necessary or desirable, they can provide much more detailed instruction about specific methods to be used (i.e., subtasks to be performed or avoided, specific resources to be used or not used, etc.) with a reduction in uncertainty about how the task will be performed by the subordinate—albeit at the cost of additional time and workload spent in the tasking process.

The shared task model at the core of Playbook provides a means of human–automation communication about plans, goals, methods and resource usage—a process akin to referencing plays in a sports team's playbook. A play can be "called" very quickly and the authority required to adapt and apply the play to the current situation is left to the intelligence resident in the system (specifically, in a sports team, in the heads of the players). Alternatively, a coach or team captain can adapt, refine or specify a play with minimal effort (since everyone knows the basic play "vocabulary" to begin with) if time permits or the situation requires. The Playbook enables human operators to interact with subordinate systems with the same flexibility as with well-trained human subordinates. For Unmanned Vehicles (UVs), Playbook actively manages missions at the highest level enabling the human to delegate objectives and partial instructions to the UV and then autonomously develop plans to accomplish those overall mission objectives. Alternatively, and unlike many other approaches to control of multiple UVs, Playbook also allows the user to "dive into" a high level play and increasingly refine and specify it, thereby permitting both high and low level control of UVs. Thus, Playbook affords the variability and flexibility of user interaction which the UV control domain demands.

The basic Playbook architecture consists of a constraint-based planning engine that shares information with the user (via the interface) in terms of a shared task model. The task model is both hierarchical and sequential. When a task (or play as it is sometimes called) is activated, the system then knows all the required and optional methods that may be used to accomplish it.

Figure 1 presents our general architecture. In our approach, a User Interface (UI), in the form of a playbook, and an Analysis and Planning Component (APC) are both based on a Shared Task Model. The Shared Task Model is the framework for all communications between these two components and between the playbook and the human operator. The human operator communicates tasking instructions in the form of desired goals, tasks, constraints and/or policies. The APC is a constraint propagation and analysis system that uses Hierarchical Task Networks. The APC can understand tasking instructions and (a) evaluate them for feasibility, and/or (b) expand them to produce method alternatives. Once an acceptable plan of atomic actions is created, it is passed to an Executive component which performs Event Handling—an outer-loop control system capable of making in-flight adjustments to the plan. The Event Handling component sequences control algorithms that actually effect behaviors in the (possibly simulated) controlled vehicles.

Through its knowledge of viable task structures in the domain and through interaction with more sophisticated special purpose tools (such as dedicated route planners, senor planners, etc.), the planning component of Playbook is capable of both fleshing out a plan

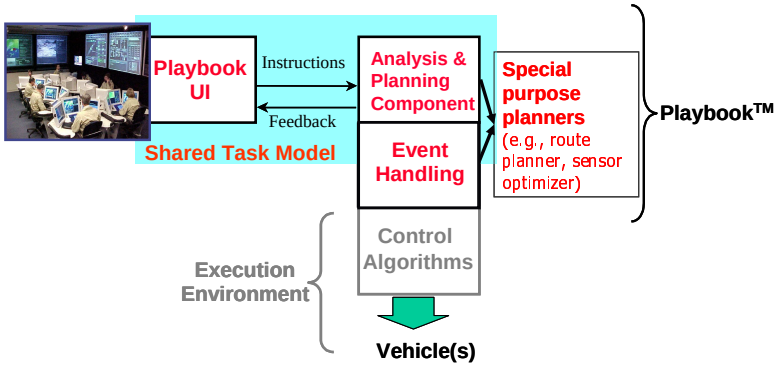

Figure 1. General Architecture for Tasking Interfaces.

within parameters specified by a user and of critiquing a plan for feasibility and goal accomplishment. The planner accomplishes this by access to knowledge about the resources (for example, quantities such as fuel or munitions, as well as less obvious resources such as time, distance or, potentially, human attention and cognition capabilities) used by specific tasks in the scenario, as well as knowledge of how legal task combinations are known to accomplish goals.

2.1 The Meaning of "Play"—The Shared Task Model

Correctly capturing the semantics of plays is critical to acceptance of the Playbook system. Plays are, in essence, compressed commands. The requestor has an expectation, when the request is issued, that some action will be taken that conforms to his/her idea of the request's meaning. The Shared Task Model illustrated in Figure 1 forms the 'language' through which humans and automation can communicate about task performance. The Shared Task Model is a hierarchical decomposition of tasks in the domain that captures the functional relationships and sequential constraints between labeled 'packets' of goal-directed behavior. A 'play' is an encapsulated set of behaviors—perhaps including alternatives—that provides a method of accomplishing domain goal(s). Plays are not scripted, static procedures, but rather dynamic templates that identify a labeled range of behaviors which humans and automation agree will fall under that label. Because plays (and their component tasks) are hierarchically defined, a very complex suite of behaviors can be very efficiently referenced by asserting the label of the parent task. This is the source of the efficiencies which come in 'calling plays'. Plays are largely synonymous with tasks, though they may represent a specific aggregation of lower level tasks with a defined label—a macro operator that can be adapted to the context in which it is 'called'. Even when modifications to existing plays are necessary, or whole new plays need to be created, we can do this very quickly by modifying existing plays.

As Boy [7] points out, the naming or labeling of tasks always involves a process of 'rationalization' or abstraction toward a 'template' that leaves the final customization of the behaviors for the actor at runtime. Another way to say this is that a task is a labeled set of potential behaviors which, in various compositions, represent alternate methods of accomplishing or performing the labeled task. When a subordinate (whether human or automation) is commanded or authorized to perform a task (by name), he/she/it is given authority to select and enact any of the various possible behaviors that fall within the space defined by the task label. The supervisor may further constrain the set of allowable methods by providing additional instructions about which methods can, must or cannot be used.

Figure 2 shows possible expansions of a task we are working with currently: Ground Moving Target Tracking (GMTT)—establishing and maintaining surveillance of a moving target. Each of the subordinate tasks shown in Figure 2 itself has a definition in terms of subtasks that must be performed to complete the parent task. This definition process repeats until an atomic task level is reached. Atomic tasks are those that are executable by a given platform, hence the atomic level is different for different platforms. Currently supported platforms include Geneva Aerospace's fixed-wing Dakota and the GTMax rotorcraft.

The task formalism we use incorporates four types of information, shown in Table 1.

These task templates contain a number of variables that must be set – parameters to the request. Some parameters must be specified for the task to be executable. Other parameters take their values from the current situational context and thus have reasonable default values that are passed to the Analysis and Planning Control Component as a part of the request. Remaining parameters are set (bound) by the planner during the planning process.

Tasks may include static and calculable information about the resources (including time) they need, required initiation conditions, expected outcomes, etc. Task models exist in three forms (1) a generalized form in a library, where they represent (in an abstract, uninstantiated form) the possible tasks which can be performed using this vehicle, (2) in a selected and partially instantiated form for use as a mission plan with additional parameters and selections that must be made at execution time, and (3) a final, executable and fully instantiated form at run time. At its lowest levels, the Shared Task Model contains primitive execution tasks that can be reliably executed by automation.

In prior work, we have used a variety of representational frameworks to encode the knowledge required for a Playbook tool to operate. Thus, we are familiar with a variety of potential approaches, including object-oriented styles and even Prolog representations. For our most recent Playbook effort, we are developing an XML representation based in part on the DARPA Agent Markup Language – Services (DAML-S) (and its successor, the Web Ontology Language – Services (OWL-S)) which should be both easier to use and more readily integrated with other tools.

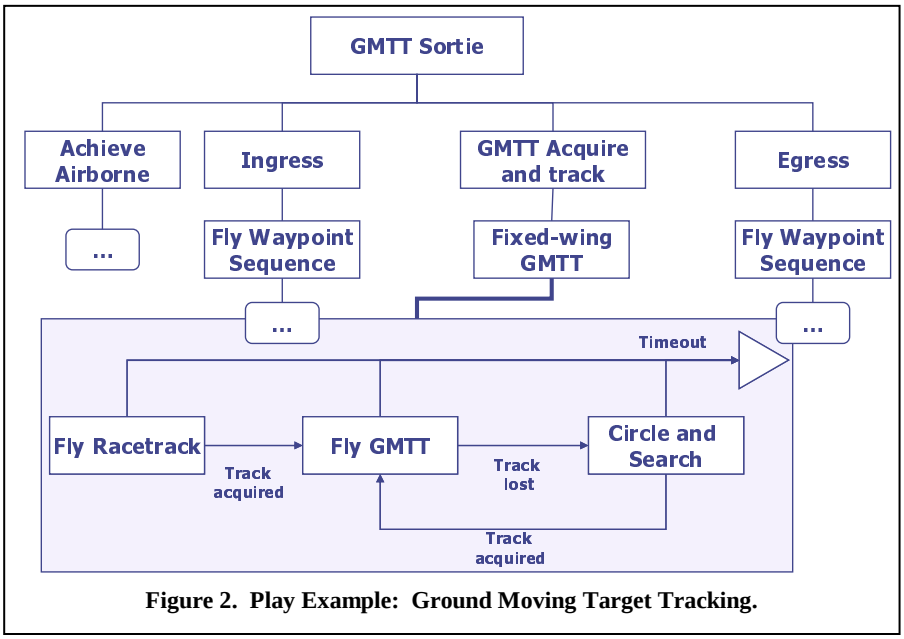

Figure 2. Play Example: Ground Moving Target Tracking.

Table 1. Task Model Element Definitions.

Task Model Element	Description
Tasks	Work domain tasks to be performed by simulated or actual human and automation actors in a constructive simulation model.
Hierarchical, Functional Task Relationships	A representation of the subtasks or methods which are capable of accomplishing a parent task.
Sequential, Conditional and Looping Relationships	Information about the ways in which tasks can be or must be strung together in order to accomplish parent tasks. Conditional branches indicate the conditions under which various branches are to be taken (including probabilistic branching). With regards to conditional branching points, we define the conditions with a simple textual rule representation as follows: *Condition A: If X, then Y.* Note that probabilistic branching can also be expressed in this format and "do-while" loop-like branching can be represented by including the 'while' clause in the conditional and placing it after the task to be repeated.
Commandable Constraints and Stipulations	Conditions and resources to be used or avoided in the plan to be produced, in terms of the specific tasks and bindings found in the plan.

2.2 Analysis and Planning Component

The Analysis & Planning Component (APC) is a central part of what we term 'Playbook'. The core of the APC is an enhanced version of a hierarchical task network planner known as SHOP2. The APC evaluates the feasibility of alternate methods of satisfying requested plays. When given a high-level play request, the APC selects among various feasible methods, issues instructions to the execution environment (either simulation or UV platform) and monitors for necessary revisions during performance. When given lower-level, more specific and detailed requests (such as a specific platform to use, specific paths to be followed, or specific scan patterns to use), the APC reviews them for feasibility and either (a) reports when requested actions are infeasible, (b) passes 'validated' user requested plans to the execution environment and monitors their performance, or (c) fleshes out high-level operator requests to an executable level (for example, choosing among available platforms, selecting waypoints for ingress and egress, identifying sensor steering parameters, etc.) within the constraints the operator has imposed. The APC is designed to use special purpose planners as adjuncts to its planning process; the route planner currently in use for one of our programs is a GIS package modified to perform the route plan function.

Whenever known resource violations occur, the planner can report that this is not a feasible plan. Similarly, whenever task combinations do not add up to the accomplishment of a parent goal, the planner can note this fact and either report it (if in critiquing mode) or choose another method for accomplishment (if in autonomous planning mode). This planning capability is not a full simulation, but rather a first-pass, coarse-grained constraint checking capability. It does not, for example, contain any ability to simulate world states or enemy actions. Nevertheless, it is a useful method of doing some plan generation and screening for obvious errors. More sophisticated simulations, and user feedback on candidate plays, could be incorporated to improve this capability, but at the cost of additional time and computational resources. This simple, first pass, screening of candidate plans has proven adequate for many applications to date—especially given the flexibility to adapt the play during execution (as discussed below).

2.3 Relaxing Constraints

Using Playbook, we have discovered that over-constrained missions are particularly challenging, because it is difficult for operators to know why a request has failed and what constraints to relax in order to find a viable solution. In essence, when a straightforward, constraint-based solver fails to find a solution, it behaves as a subordinate who says, simply "I can't" when asked to do a task. Even if honest, this isn't as helpful as it could be.

To solve this problem we have added a "best-effort" planning mode to the Playbook [8]. Best-effort plans may be directly useful, or can be used as a guideline in relaxing constraints. In essence, Playbook's APC begins a heuristic-guided relaxing of user-imposed constraints until it finds a method of accomplishing the goals of the play that it believes is feasible. This new, "relaxed-constraint" play can never be executed without supervisor approval, but it is a way of both communicating what needed to be modified in the initial request in order to arrive at a feasible plan and of having a plan available for quick execution. This new capability gives Playbook the ability to respond like a subordinate who says "I can't do what you wanted, but here is something that I think is close to what you want and that I know I can do. How about that?" As such, this suggests that presenting a relaxed constraint plan should serve as the first step in a negotiation process whereby the human supervisor and Playbook's APC arrive at a feasible and desirable solution—though we have not yet implemented such a process.

2.4 User Interfaces

The presence of the Shared Task Model and its use as a communication medium between the user and the APC affords the potential to create a wide variety of different user interfaces, each customized for different usage environments. For example, an interface designed for creating and editing plays will need much more detailed presentation and manipulation capabilities for the underly-

ing task structures than will one designed to rapidly call largely pre-defined plays in, say, a combat environment. Similarly, a Playbook used primarily for mission planning will likely permit much more detailed investigation of alternatives and shaping of the instructions provided to automation—and will benefit from auxiliary visualization tools such as detailed maps, plan tree graphics, Gantt charts for timeline visualization, etc. By contrast, a Playbook used for commanding UVs in a high stress, high tempo combat environment (such as an organic unit supporting a ground force or unmanned "wingmen" under the command of an airborne mission commander) might demand a reduced number of plays with limited customization options—perhaps even "callable" via a speech interface or presented in very minimal form on a PDA.

The majority of our recent work has focused on Playbook support for small, dismounted, ground-based units issuing "requests for service" to a heterogeneous pool of UAVs while continuing their ongoing tasks such as apartment searches or urban combat. "Service requests" are much like play commands in that they require a designation of the appropriate behavior to be executed at a high level (the "play" to be performed) and they will benefit from permitting flexibility in the amount of instruction provided about how that behavior is to be effected. Such a concept of operations assumes that issues such as deconfliction and basic pilotage are either fully automated or performed by a user other than the small unit soldier. For that soldier, though, several requirements become clear. First, current operator-to-platform ratios on the order of 4-1 will no longer be acceptable. Second, the current practice of providing a dedicated workstation for each vehicle or vehicle type will also be unacceptable when individuals must interact with multiple, heterogeneous vehicles. Operators will need a common interface for multiple vehicles. Third, new usability and training requirements will be imposed. Small unit soldiers will not be able to spend months training to be rated for a vehicle, nor will they devote full attention to vehicle management (much less to managing a single vehicle subsystem). Instead, UAVs must be controllable with much less training and while engaged in many other activities.

These requirements have led us, in this work, to an interface largely optimized for speed of command while still affording some flexibility and providing adequate execution monitoring to permit interventions. Generally, the human operator "calls a play" by designating it from a library and then, instead of drilling down into a graphic representation of the play such as the conceptual illustration in Figure 2, s/he stipulates values for a few key parameters which correspond to significant branch points in the hierarchical set of alternative methods which make up the play. The user is required to provide some such parameters (i.e., the target of an attack or surveillance play), but most can be provided with reasonable, heuristic-based defaults (perhaps configured pre-mission with the user's approval). Such defaults can be, but do not have to be, modified by the user at execution time. Examples include the time to begin and duration to maintain a surveillance, the type of vehicle to be used, the area of ground surveillance, etc. In general, they permit the calling of a reasonably accurate play very quickly—generally with as few as 3-5 mouse clicks.

Figure 3 provides two illustrations of user interfaces we have created for this domain. The left hand example is designed to be hosted on a laptop or portable notebook and provides more inspection and modification views of routes (via maps) and plans (via timelines and tree diagrams) and well as larger, higher resolution video imagery. The right hand example is a PDA-based implementation that affords limited plan/route visualization, video imagery and plan editing capabilities.

3. Recent Playbook Developments

Our most recent modifications to the Playbook design have included the incorporation of the Ground Moving Target Tracking (GMTT) play illustrated in Figure 2. GMTT is a play which enables a human commander/supervisor to say, effectively, "I want to track this (moving) target on the ground" with the ability (but not the requirement) to further specify the type of platforms to do the tracking, the period of time during which tracking is to be performed, the routes to be flown, etc. This play represents our first serious foray into outer loop, discrete-event control via Playbook and is, thus, something of a milestone in Playbook development.

Previous plays (primarily surveillance plays which consisted of variations on a route to fly and a surveillance pattern to execute for a fixed period of time once at a target area) could generally be constructed and then passed in "batch" mode to an Executive which manages event handling for execution. The Executive would then simply watch for arrival at the target area and, when it occurred, would issue a command to turn appropriate sensors on (and reverse this process upon departure). To properly do

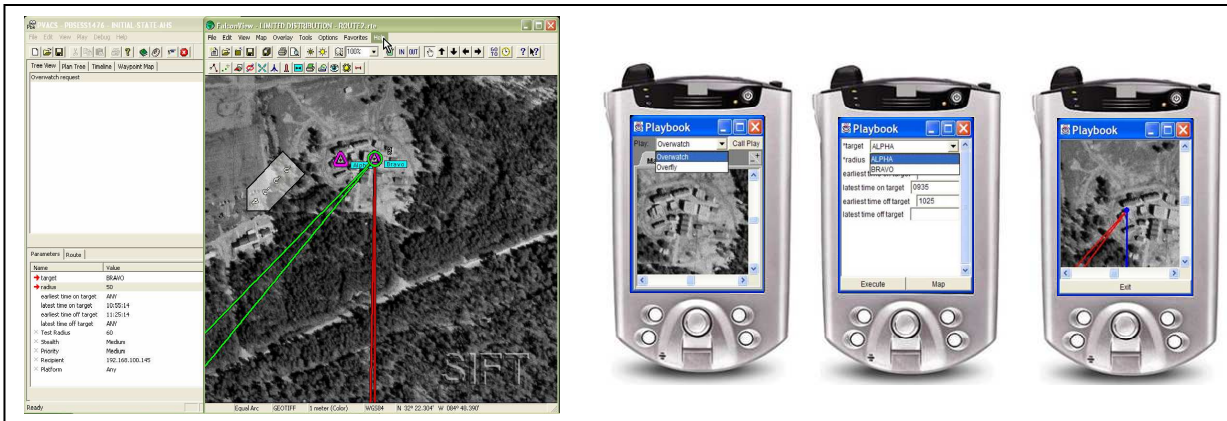

Figure 3. Two interfaces for a Playbook for Dismounted Small Unit Soldiers.

GMTT, the Executive must behave in a more complex fashion. It must watch the UAV's track status to see when it acquires a track, change the control mode accordingly, and then watch to see if track is lost again. If so, it will put the UAV into one of (potentially) multiple alternate search patterns to try to recover the track. This process iterates throughout the vehicle's tracking period (until the requested track period, or the vehicle's portion of the tracking period, is over, or until the vehicle has exhausted a time-out limit for search) until the vehicle can progress to its egress phase and return to home or to a destaging point.

This is not yet full replanning, as will be necessary for intelligently handling dramatic upsets. We have not yet implemented such capabilities (and the user interactions that would be necessary to support reporting of such events and negotiations about how to address them). The outer-loop, discrete event control required for GMTT, however, represents a step in that direction. Batch planning is no longer possible; instead, there is a presumption that when the user delegates authority to perform a GMTT "play", s/he is authorizing the Playbook to coordinate UAV behaviors within the boundaries defined for the play. These include unpredictable transitions between tracking and searching behaviors. Furthermore, this degree of unpredictability makes subsequent events in the play less predictable as well. For example, because we don't know where the vehicle will end up when its tracking period is over, we can no longer generate a batch plan which includes an egress route. Instead, we generate an initial placeholder waypoint sequence at the start of the mission for returning home, and then after tracking is done, we generate a new, executable waypoint sequence and download it. Furthermore, while executing the play, Playbook must continually monitor and update parameters such as available fuel to maintain its level of assured performability of the commanded play—that is, to ensure that the vehicle has sufficient fuel to perform its egress sub-task. This ongoing monitoring and updating of an evolving play becomes particularly complex in those cases where Playbook must coordinate the behaviors of multiple UAV's to, for example, supply continual coverage for a tracking period longer than any one of them could provide alone.

There are many challenges remaining as we begin to move from comparatively simple, waypoint-based plays to more complex and dynamically changing plays such as GMTT. We have demonstrated the possibility for a Playbook-like delegation interface to behave appropriately within commanded constraints, but substantial work remains to be done (and, in all likelihood, done in a application-specific fashion) to tune the interaction between human and automation for behavior in such unpredictable situations. The human supervisor should have the ability (at least for some applications) to provide a much richer set of constraints (for example, "track the target for the next 30 minutes, but if it crosses the border into the neighboring country, cease track"), and perhaps, a richer set of interactive and reporting or querying behaviors (e.g., "Maintain track for 30 minutes, but let me know if you drop below 20% fuel reserves and if the target starts firing at you, ask me what you should do next"). Such interactions are very much in the spirit of the delegation interaction we are seeking to provide with complex and sophisticated automation, and they are within reach of our current architecture, but providing them must await future work.

Finally, it is worth noting that as we build more plays into our Playbook prototypes, we are beginning to see evidence of the scalability of this approach and ease of composing novel plays once an existing play library begins to be populated. For example, adding the GMTT play on top of existing surveillance plays made extensive reuse of waypoint flying behaviors for ingress and egress, loiter modes for search, and sensor manipulation behaviors. In all, the addition of this complex, novel behavior required only about 125 new lines of code to be added to the Playbook APC and 430 lines to the Executive event handler. Most of the latter were devoted to simply reading and parsing ground vehicle track messages from sensors and are therefore not a core part of Playbook's reasoning. If one deducts the amount of code necessary to simply read these messages and translate them into Playbook data structures, less than 200 LOC were required overall. This represents extraordinarily good reuse and bodes well for future scale-up of Playbook data structures.

4. Playbook Benefits

Playbook has been designed to provide the flexibility that comes from providing an intelligent supervisor and intelligent subordinates the ability to collaborate flexibly about the precise task and method that the subordinate is to perform. That is, both the delegated task and the degree of specificity with which that task is delegated and constrained must be under the flexible control of the supervisor. We have anticipated [3] that providing such an interaction style will provide multiple benefits for the human + machine collaboration, including:

- Increased user satisfaction and acceptance
- Decreased human skill loss
- More balanced workload
- More accurate and balanced automation reliance decisions
- Increased situation awareness (relative to a more fully automated or autonomously adaptive automation approach)
- Improved human + machine system performance (especially in flexible and unpredictable domains which offer enough time for human awareness and planning)

In previous research [9], we obtained empirical evidence for the efficacy of Playbook type interfaces for mission efficiency. This work, involving human control of multiple robots in the RoboFlag capture-the-flag simulation, used a Playbook-like interface that permitted flexible control at various hierarchical task levels. The results showed that the multi-level tasking provided by the Playbook interface allowed for effective user supervision of robots, as evidenced by the number of missions successfully completed and the time for mission execution. In addition, the flexible Playbook interface was superior to fixed control conditions in which the operator had access only to either manual control of individual robots or automated plays alone, but not both. Finally, the superiority of the flexible Playbook interface was particularly apparent in conditions when the opponent posture was unpredictable. These findings provide strong support for the view that the Playbook allows for effective tasking of multiple robots while keeping the operator in the decision-making loop, without increasing op-

erator mental workload, and allowing the human operator to adapt successfully to unpredictable changes in the environment. These benefits are important because traditional human-automation interfaces have often been found to result in significant system and human performance costs—including mode errors, user under- and over-reliance on automation, and reduced situation awareness [2,10]. Such limitations are sometimes severe enough to result in catastrophic accidents, as evidenced by numerous analyses of aviation incidents, including unmanned aircraft [11,12]. Hence, the development of appropriate human-automation interfaces is critical for effective human supervision of autonomous agents, including robots and unmanned vehicles. Playbook provides such an interface concept. Its benefits may be particularly apparent in situations of environmental uncertainty and where unexpected events occur, which can make pre-programmed automated behaviors ineffective.

5. Ongoing and Future Work

Far more than 'just' a user interface, Playbook provides a complete architecture for the integration of human input, intelligent a priori planning, reactive planning and event handling, and ongoing vehicle control loops. To date, development on this tasking interface architecture has been directed at ground-based control of remote vehicles. However, our general tasking interface architecture extends to work with software components and is not limited to the vehicle control domain. SIFT is pursuing the application and extension of Playbook in a number of different directions. One particular direction is in developing methodologies to build more extensive task models, such as the ability to derive Playbook task knowledge from results of Cognitive Work Analysis (CWA) of a task domain and then use the Playbook architecture (including UI and planning components) to produce useful task timeline inputs for a constructive simulation. Thus far, our emphasis in developing a representation has not been on computational efficiency or even on specific software representations, but rather on ease of accurately and comprehensively expressing knowledge requirements. Another current project, sponsored by a Navy Small Business Innovation Research grant (N05-017) is focusing on the integration of Playbook's interactive and relaxed-constraint planning into a sophisticated user interface environment.

Under these and other projects we are learning more about how to modify the Playbook approach for a wide variety of human tasking contexts. Playbook and other tasking approaches to adapting (rather than adaptive) automation behavior represent a sophisticated and increasingly competent approach to interacting with sophisticated automation, unmanned vehicles and robots with unique payoffs in terms of human control, awareness and overall performance.

6. ACKNOWLEDGMENTS

The PVACS program is funded by a Phase II Small Business Innovation Research grant from DARPA IXO, administered by the U.S. Army Aviation & Missile Command, contract number DAAH01-03-C-R177. We would like to thank Dr. John Bay for his oversight and advice during the investigation. We are also grateful for the work of Geneva Aerospace personnel, especially Billy Pate, on the VACS system and for educating us about it.

7. REFERENCES

[1] Sheridan, T. Supervisory control. In G. Salvendy, (Ed.), *Handbook of human factors*. John Wiley & Sons, New York, (1987), 1244-1268.

[2] Parasuraman, R. and Riley, V. Humans and automation: Use, misuse, disuse, abuse. *Human Factors, 39* (1997), 230-253.

[3] Miller, C. and Parasuraman, R. (in press). Designing for Flexible Interaction between Humans and Automation: Delegation Interfaces for Supervisory Control. Accepted for publication in *Human Factors*.

[4] Miller, C., Pelican, M. and Goldman, R. "Tasking" Interfaces for Flexible Interaction with Automation: Keeping the Operator in Control. In *Proceedings of the Conference on Human Interaction with Complex Systems*. Urbana-Champaign, Ill. May, 2000.

[5] Miller, C., Goldman, R., Funk, H., Wu, P. and Pate, B. 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*. Baltimore, MD; June 7-10, 2004.

[6] Parasuraman, R. and Miller, C. A. Delegation interfaces for human supervision of multiple unmanned vehicles: Theory, experiments, and practical applications. In N. Cooke, N. and H. Pedersen (Eds.) *Human Factors of Remotely Piloted Vehicles*..Elsevier, New York. (2006, in press).

[7] Boy, G. *Cognitive Function Analysis*. ABLEX; Stamford, CT, 1999.

[8] Goldman, R., Miller, C., Wu, P., Funk, H. and Meisner, J. Optimizing to Satisfice: Using Optimization to Guide Users. In *Proceedings of the American Helicopter Society's International Specialists Meeting on Unmanned Aerial Vehicles*. Chandler, AZ, January 18-20, 2005.

[9] Parasuraman, R., Galster, S., Squire, P., Furukawa, H. & Miller, C. 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*(4), 2005, 481-493.

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

[11] Parasuraman, R. and Byrne, E. Automation and human performance in aviation. In *Principles and practice of aviation psychology*, P. Tsang and M. Vidulich, (Eds.), Erlbaum, Mahwah, NJ., 2003, 311-356.

[12] Manning, S., Rash, C., LeDuce, P., Noback, R., and McKeon, J. The Role of Human Causal Factors in U.S. Army UAV Accidents, Technical Report USAARL-2004-11, U.S. Army Aeromedical Research Laboratory, Ft. Rucker, AL, 2004.