Delegation in LoA^3 Space

SIFT Technical Report
05-TUSC-01

Dr. Christopher A. Miller
Dr. Robert P. Goldman
Mr. Harry B. Funk

Smart Information Flow Technologies
211 First St. N., Suite 300
Minneapolis, MN 55401

{ cmiller, rpgoldman, hfunk}@sift.info
612-339-7438

Using Delegation as an Architecture for Adaptive Automation

Prepared by: **Chris Miller, Robert Goldman, Harry Funk**

File name: **Delegation for AA.doc**

1. Document History

Version	Prepared	Description
.1	10/8/2004 11:37 AM	Initial draft
.2	10/19/2004	Edit for consistency
.3	10/21/04	RPG edits and comments
.4	10/25/2004	HBF edits and comments; some RPG responses.
1.0	10/27/04	Integration and final edits by CAM. First version distributed to IA Tech and AFRL.

2. Overview and scope

This document describes an approach to using a delegation framework, such as that provided by SIFT's Playbook[1], as a structure for coordinating, managing and organizing experiments on Adaptive Automation and "Levels of Automation"—that is, as an Adaptive Automation Architecture (AAA). We use a very simple Overfly play as an example. In our discussion, we show how over 400 different possible human-automation interaction configurations can be obtained, even with just this simple play.

3. Adaptive Automation and Levels of Autonomy

Adaptive automation (AA) is automation which adapts to the human user's needs—for example, providing more support in periods of high human workload or time stress, providing more support for a novice user and less for an expert, providing more support for a physically or cognitively impaired user and less for an unimpaired one, etc (See Morrison (1993) for a taxonomy of AA approaches). Levels of Automation (LOAs) are any of various convenient frameworks for describing the degree or amount of automation support that is provided. The two concepts are related in that AA can be described as varying over a spectrum of LOAs. LOAs are sometimes used as a mental model, or even as an interface concept, for understanding and managing AA.

An LOA scheme is simply a way of dividing up, describing and easily referencing combinations of human and automation behaviors. An LOA scheme can be used to characterize or describe the mix of human and automation control in a given system—usually in some less-to-more series of "levels". As such it is primarily of interest to theoreticians and, maybe, system designers. If an

[1] The term "Playbook" as used here and throughout to refer to a delegation architecture and interface for automation control, is Trademarked by SIFT, LLC.

Table 1. Levels of Automation (after Sheridan and Verplank, 1978).

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

LOA scheme is to be *used* as a means of controlling a system and achieving work goals, multiple levels must be available to an operator and it must be possible to change or adapt among them on the basis of various aspects of context: user preference, current workload, criticality and expected user performance, etc. In other words, it must be an adaptive (or adaptable)[2] automation system. The issue with various schemes for levels of automation is the granularity and sensitivity with which they carve up the range of possible human/automation interactions to control a system.

Sheridan (1987) proposed the best-known framework for LOAs, which we illustrate in Figure 1. Sheridan's initial work (Sheridan and Verplank, 1978) offered a one-dimensional spectrum of 10 patterns of human and automation behavior and responsibilities ranging from no automation involvement through increasingly autonomous behaviors from automation, to wholly automatic (no human involvement). This scheme has the advantage of being extremely easy to understand, use and remember, but it is not particularly sensitive.

Parasuraman, Sheridan and Wickens (2000) realized that this one-dimensional scheme failed to discriminate between some very different uses to which automation could be put. They characterized these uses in a second dimension (see Figure 2)— that of information processing stages or functions. They identified four processing stages: Information Acquisition, Information Analysis, Decision Selection and Action Implementation. They argued that more

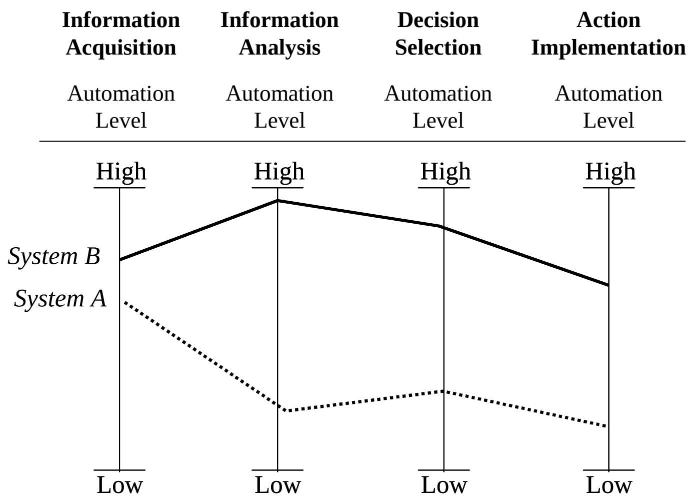

Figure 2. Levels of automation by information processing phase for two systems (from (Parasuraman, Sheridan and Wickens, 2000.)

[2] For the purpose of this paper, we are ignoring the distinction made by Opperman (1994) between adaptable and adaptive systems—those which can be made to exhibit alternate behaviors by a human and those which select their own alternate behaviors autonomously, respectively. Both types of adaptation will be referred to as "adaptive automation" in the context of this paper—as we understand is in keeping with the intention in the initial description of this SBIR topic.

or less automation support can be provided for each of these functions within a single system and the pattern of high and low automation support for each of these four functions characterizes every aiding system.

We have argued elsewhere (Parasuraman and Miller, 2003; in preparation) that the two dimensional LOA scheme proposed by Parasuraman, et al., while a major improvement in sensitivity over one-dimensional schemes, remains too coarse-grained for many uses. In particular, this scheme does not address delegation interactions between human and automation. In order to have the same kind of "conversation" about who should do what and how that human supervisors can have with intelligent human subordinates, machine automation needs to understand and be able to reason about the same kind of task-based, hierarchically and functionally organized knowledge structures that humans seem to use.

SIFT's Playbook approach to delegation interactions is one approach that strives to achieve this interaction (Miller, 2003). Playbook uses a hierarchical task model as a means of communication about intents and plans between a human operator and an intelligent planning and control agent. This task model serves as, essentially, a very fine-grained LOA spectrum, allowing users to say *exactly* what tasks they want automation to do and how rather than relying on setting a coarser-grained LOA . Furthermore, because Playbook's task model is hierarchically structured, users can interact with it at higher or lower levels of functional detail—commanding high level functions or drilling down and making specific stipulations.

The question we address in this document is how can Playbook (or delegation interfaces in general) be used as an architecture for adaptive automation? Another question of interest is how the Playbook architecture can be used to configure and allow control of experiments on adaptive automation and LOAs. Such experiments are necessary for understanding what kinds of human-automation interactions will be most useful in application domains of interest, such as the control of multiple UAVs.

Below, we first lay out a framework for what we believe is meant by a "Level of Automation" and then show how the Playbook's framework can be used to configure, manage and adaptively command different LOAs—both by an operator at mission planning or even run time, and by an experimenter interested in setting up different experimental conditions for exploring effective combinations of humans and automation in the performance of a range of tasks.

4. What is an LOA?

Humans and automation can interact in a huge range of different ways, and the number of ways increases as computer technology enables automation to do more, and do it via new and different modalities. A "Level of Automation" framework, as we defined it above, is simply a convenient parsing of the myriad different ways humans and automation can interact into some convenient set of categories or levels. In such frameworks, an LOA labels a range of relatively homogenous, alternate human-automation relationships.

A human + machine system has *more* automation (or, alternatively, a higher "level of automation") when any of three things is true:

1. The automation can be tasked at higher, more abstract levels—that is, it can be relied on to make more of the decisions about how to execute a broader, higher-level task. The system is making decisions that would otherwise have to be made by a human supervisor: decisions about how to achieve lower level goals. If I can tell a subordinate "Get me to LAX on Monday", that subordinate is more autonomous than if I have to tell him/her

"Book me a flight to LAX on Monday; make sure you use Orbitz because they're cheaper; it'll be good if you can make it a non-stop; and make sure to get me a car and hotel, too".

2. The automation can perform whatever it does more independently, with more authority; it doesn't have to submit its decisions to me for approval or review. If my subordinate has to have approval of each decision about my travel plans, I'd say s/he is less autonomous than if I s/he can simply hand me a complete itinerary with tickets.

3. The automation controls more resources or assets. If I allow my subordinate to make \$5000 decisions about SIFT's credit line (not to mention booking the corporate jet and limo fleet ☺), then s/he has more autonomy than if any decision over \$100 has to be routed to me for approval.

These dimensions are not entirely independent. Instead, they provide three views into the delegation relationship between a supervisor and subordinate and they together define a "delegation space" within which achievable relationships can be described.

It is not surprising that prior efforts to characterize Levels of Automation express one or more of these dimensions. Sheridan's levels (as shown in Figure 1 above) say less about specifically what tasks or goals automation is performing, and more about the relationship between the human and the automation. That is, the levels describe the autonomy or *authority* relationship between human and automation. Does the automation have the authority to do whatever it is it's doing without prior approval? If so, then it operates at Sheridan's level 10. If it can do whatever it does under its own recognizance but must report what it has/is doing to the user, then it is at Sheridan's level 7, etc. This spectrum describes *how* human and automation relate to each other, but doesn't say much about specifically *what* each of them is doing. For convenience, we will refer to this characterization of authority or autonomy relationships between human and automation as the ***Level of Authority*** dimension.

By contrast, Parasuraman, et al.'s two-dimensional scheme adds some description of *what* the automation is doing, albeit in terms of four coarse-grained information processing categories. This dimension of what is being done is crossed, however, with the same sense of how the relation is characterized (the Level of Authority) above. Parasuraman, et al., imply that the "high" and "low" dimension for each of their information processing functions is related to Sheridan's initial 10 level spectrum. Thus, the Parasuraman, et al. framework subsumes Sheridan's initial framework and now defines a Level of Automation as a combination of a level of authority for each of the four information processing functions.

What is the dimension that Parasuraman, et al. have added? As we have noted above, it begins to describe what each of the actors is doing within a specified LOA—therefore it is a description of activity. We argue that the activity dimension can be subdivided into many finer categories and represented by a hierarchical task model. In other words, the four information processing stages are just a coarse aggregation and categorization of specific tasks. Instead of saying that "information acquisition automation is high" we could more precisely say that automation is deciding how to direct and configure sensors and reporting those decisions to the operator. For a more detailed elaboration of this argument, see Miller and Parasuraman, 2003; forthcoming.

Since we can use a hierarchical task model to characterize what human and automation are doing, we can refer to this dimension of automation activity or behavior as a ***Level of Abstraction***. Automation can have responsibility for higher- or lower-level tasks within the task hierarchy. Having responsibility for higher level tasks presumes responsibility for the tasks which fall below them (although the person doing the delegation retains the right to place constraints or stipulations on the range of decision possibilities about how to perform lower level tasks). So a “Level of Automation” is, therefore, even in Parasuraman, et al.’s model, a combination of a level of authority and a level of abstraction—automation has responsibility for one or more tasks at a given level of abstraction and with a given level of authority.

Are we done? Not quite. The Level of Authority x Level of Abstraction framework described above characterizes who is doing what and how they relate to each other. Especially in military domains, however authority relationships are frequently defined along resource lines as well as task or functional relationships. Hence, we define a third dimension along which to characterize delegation—a ***Level of Aggregation***[3]. The level of aggregation identifies how much (and/or which type) of resource each actor is authorized to use. When a supervisor delegates a task to a subordinate, that subordinate will be granted authority over some set of resources (including access to his or her own time and energies). The subordinate is expected to be able to either come up with a plan for completing the task within those resources or to tell the supervisor why s/he cannot. On the other hand, it is also frequently the case that there are some shared resources that the subordinate may have access to only after performing various coordination activities which enable their use. The extent of the subordinate’s authority over resources can be categorized in a way similar to the way we have categorized the subordinate’s authority over his/her own activities.

These three dimensions, Level of Authority, Level of Abstraction and Level of Aggregation, therefore define a ***Delegation Space*** of human-automation relationships within which delegation occurs and can be characterized. In short, ***delegation means giving to a subordinate the responsibility to perform a task (with its subtasks), along with some authority to decide how to perform that task and access to some resources with some authority to decide how to use them to perform the task.*** Thus, the three scales must be used to specify four variables which define the delegation space: the level of abstraction and the level of authority on it, and the level of aggregation and the level of authority on it.

Figure 3 illustrates these relationships. Imagine a set of tasks which can be performed in a domain, arranged in a hierarchical, abstraction relationship. A supervisor *controls* some portion of those tasks—which means s/he has *authority* over deciding when and how they need to be performed and when and how to use resources (which s/he also controls) to accomplish them. When the supervisor delegates some of those tasks, s/he is delegating that control—over the decision(s) about how and when they need to be performed, and over the decision(s) about how and when to use allocated resources to accomplish them.

The authority to make those decisions need not be complete. The supervisor can assert constraints on how the subordinate makes those decisions and/or can require the subordinate to perform various degrees of checking and request approval before the proceeding with a plan, but there must be *some* authority to make those decisions handed over if there is to be any benefit from delegation. A “Level of Automation” is, therefore, a combination of tasks delegated at

[3] The choice of the term “aggregation” here to refer to the part-whole dimension of objects in a system, as well as the reference to a means-ends dimension of functions and sub-functions as an “abstraction” dimension, are conscious references to Vicente (1999) and Rassmusen’s (1984) framework for Cognitive Work Analysis, which use the terms in a similar fashion. Plus, they provide nicely alliterative LOA abbreviations. We do not otherwise claim to be using the Cognitive Work Analysis framework here.

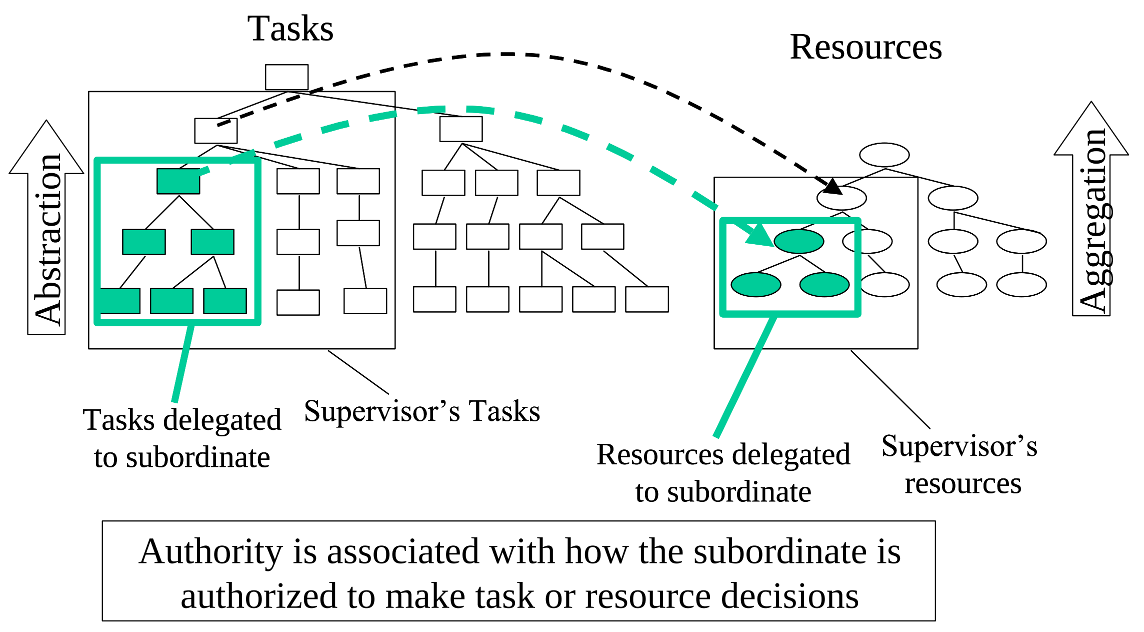
Tasks
Resources
Abstraction
Aggregation
Tasks delegated to subordinate
Supervisor's Tasks
Resources delegated to subordinate
Supervisor's resources
Authority is associated with how the subordinate is authorized to make task or resource decisions

Figure 3. Delegation relationships as characterized by abstraction, aggregation and authority dimensions.

some level of abstraction with some level of authority and resources delegated with some level of authority to be used to perform that (and perhaps other) task(s). The "level of automation" in a human-machine system increases if the level of abstraction, level of aggregation or level of authority (on either abstraction or aggregation) increases.

In the next section, we will work through an extended example illustrating how these dimensions can be used to identify and characterize various human-automation relationships. In the following section, we will describe how they can serve as a framework, especially in conjunction with a Playbook interface, for configuring and exploring different human-automation relationships in experimentation. In the final section of this paper, we will describe how this framework could be used before or even during a mission by an operator to actively manage, control and maintain awareness of what his or her automation was doing.

5. An Example of Interactions in the Delegation Space

As a simple example of the delegation space and its utility in characterizing different human-automation relationships, let's explore the following set of tasks in a hierarchical relationship:

- Assume that the highest level task/function is something we might call "Overfly target" which, for this example, is defined as getting an aircraft, equipped with an appropriate sensor, to fly within an acceptable distance of a designated target and having it take some sensor imagery. This function is a top-level "play" in a Playbook.

- Overfly is decomposed as illustrated in Figure 4. Each level of decomposition contains the sub-tasks (or sub-plays) required to accomplish the parent task. It also contains alternate methods (alternate strings of sub-tasks) that could be used to accomplish the parent (though none are illustrated here). For example, the task "Achieve Airborne" could be accomplished by obtaining a plane and having it take off (via various methods) or by requisitioning one that is already airborne.

- The only task that is further decomposed to a third level is "Fly to Target" which is decomposed into a sequence of "Fly-to-Waypoint" tasks. Each of the other tasks at the second level could be further decomposed but has not been for simplicity. The loop around "Fly-to-Waypoint" indicates that it may need to be repeated a number of times.

5.1 The Abstraction Dimension

Figure 4 illustrates the Abstraction dimension. Overfly is a more abstract task than is Fly-to-Waypoint. This is the dimension that Playbook has traditionally manipulated. Operators with a Playbook can accomplish an Overfly mission by commanding "Overfly" (i.e., at a high level of abstraction) and letting Playbook's Planning and Analysis Component (PAC) figure out the best way it can to perform all of the sub-tasks in Figure 4, or (in principle) by issuing commands at the intermediate level of abstraction—saying, for example, "Achieve Airborne" (and having the PAC chose a likely aircraft), then saying "Fly-to-Target" (and having the PAC develop a route and fly it) or by issuing a series of waypoint commands—each, essentially, a "Fly-to-Waypoint" command in it's own right. Thus, the operator with a Playbook can traverse the Abstraction dimension at will.

Does this ability for a Playbook to traverse levels in the Abstraction dimension make it an AAA? As with many things, it depends on how we define our terms. As one moves up in levels of abstraction (or tasks in a task hierarchy), one is referencing tasks at higher levels. If the automation is competent to handle the planning and execution at these levels, we enable operators to hand off higher-level tasks to automation, and we are generally saving them workload. This workload savings has always been a motivation of higher levels of automation. Thus, it seems entirely appropriate to call Playbook a type of Adaptive Automation Architecture.

On the other hand, as we saw above, Sheridan's Levels of Automation define a dimension of authority relationships that Playbook has largely ignored to date—what we characterized as the Levels of Authority dimension (especially as applied to task performance within the Abstraction dimension) above. We also saw that the Parasuraman, et al. model takes a two-dimensional approach to defining a level of automation—a level of authority in Sheridan's sense crossed with a type of task: an element of our abstraction dimension. Therefore, to qualify as an AAA in their terms, we must include this dimension of authority. Also, the set of resources controlled by automation is another useful and intuitive dimension to which an AAA ought to be sensitive. Thus, it will be important and useful to add these two new dimensions to the abstraction dimension that Playbook currently uses. In that way, we will provide a much richer vocabulary with which to understand and control adaptive automation.

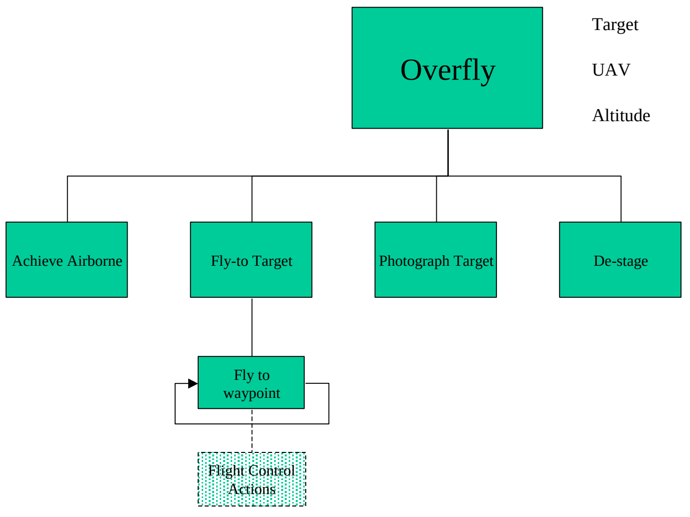

Figure 4. A simplified task decomposition for Overfly.

5.2 The Authority Dimension

Sheridan's original (1987) Levels of Automation describe a dimension of authority—how much can automation do on its own and with what required coordination. This dimension has been largely held constant in our implementations of Playbook. Automation has generally been free, within our

Playbook implementations, to develop any plan it could within the constraints stipulated by the human. After developing such a plan, the Playbook would submit it for approval, either prior to, or concurrently with, executing it.

But this has been a simplification on our part. In fact, delegation *inherently* involves some degree of authority—otherwise, there is no savings in workload. It seems a useful heuristic to say that, if a supervisor delegates a task, s/he is delegating the authority to make decisions about how to perform that task, although s/he retains the ability to impose constraints on the decisions that are made and the authority may not be complete—the subordinate may be required to check further.

So we should introduce an explicit, multi-level authority dimension into the Playbook architecture to extend our ability to mirror human-human delegation. Sheridan's levels form a nice spectrum, but we suggest the following distinct categories of authority relationships:

1. *Full*—The supervisor delegates full authority to the subordinate to decide how the task should be executed. The subordinate is charged to 'make it so' and is not required to have any further coordination, permissions or even information exchange with the supervisor.

2. *Inform*—the supervisor delegates full authority to the subordinate as above, but the subordinate is required to inform the supervisor of the decisions and execution actions taken. The supervisor retains no ability to override or approve those steps, however. It might make sense to inform human supervisors only after execution is complete if there is no override or alteration capability. Thus, this level might be better thought of as "Report" than "Inform".

3. *Override*—the supervisor delegates the task, but expects information (as above), and additionally retains the right to step in and override any or all of the subordinate's plan. The subordinate can proceed with execution until and unless the supervisor intervenes. (There might be an agreed upon lag period which the subordinate must wait for an override from the superior, and there is no guarantee here that the supervisor's intervention might not produce instabilities in the system, or that human intervention can be accomplished within a timely fashion. These are challenges that might make this, or other, levels of authority unfeasible for some systems, and are a challenge that is not directly addressed by either Playbook or by this AAA—though it does point to the increased importance of the types of human-in-the-loop experiments that this architecture will support. That is left to the supervisor to manage … and s/he might need some assistance in it).

4. *Approval*—the supervisor delegates decision/planning authority about how to accomplish the delegated task, but retains the right to explicitly approve actions before they are taken. The subordinate must submit the plan (or, perhaps, multiple plans) to the supervisor and cannot proceed until one of them has been approved for execution by the automation.

5. *Recommend*—the supervisor partially delegates planning authority alone; s/he retains execution authority. In other words, the supervisor authorizes the subordinate to recommend courses of action, but the supervisor will still make the final decision about which course to execute and who will do the executing. Recommend authority differs from Approval authority primarily in that the automation is doing the executing, after approval, in the latter case, while either the human or the automation (according to the human's choice) may do the execution in Recommend.

6. *Monitor*—the supervisor retains all decision and execution authority, but authorizes the subordinate to maintain awareness and to offer recommendations or critiques. In Monitor authority, the automation does not begin by providing a recommended plan; instead, the human supervisor begins planning and execution activities and the automation operates only by recognizing and offering critiques or improvement suggestions.

7. *None*—the supervisor delegates no authority for how to plan or execute the task. The task is not delegated. "None" delegation is the null or degenerate case and is included only for completeness. In practice, saying that one delegates with "No" or "None" authority is equivalent to not delegating.

Note that by defining the roles in the above authority spectrum as "supervisor" and "subordinate" we have avoided stipulating which role the human and automation play.

Whatever task (at whatever level in the abstraction hierarchy) the supervisor delegates, s/he may in principle delegate with any of these levels of authority. For example, if the supervisor delegates the "Fly-to-Target" task from Figure 4 above, s/he could do it with Full, Inform, Override, Approval, Recommend or Monitor Authority. Furthermore, while the default assumption may be that all the tasks under a parent task in the abstraction hierarchy should have the same level of authority, this doesn't have to be the case. It is entirely possible, for example, that the subordinate have Approval authority for the Achieve Airborne, Full Authority for Fly-to-Target, Recommend authority for Photograph Target and Inform Authority for Destage.

We may want to establish default authorities for specific kinds of tasks based on policy or where only certain authority levels are feasible or may make sense. A simple, feasibility example might be that the human + machine system does not afford the ability to inspect or approve/override specific flight control commands, therefore in that system those tasks would have to go to the automation with Full or Inform authority. Although there is no ability to modify that authority level, our approach still allows us to represent and reason about it. A more significant example, based on policy, might be the standing ROEs that weapons release must always be approved by a human operator—this means that for any task which involves weapons release, that task must be performed with Approval authority (at most) delegated to automation.

One particularly valuable use for default authority levels may be in assigning authority levels to resource usage, as we will see in the next section.

5.3 The Aggregation Dimension

Clearly, a system that controls more entities, or more subsystems has more (or a higher level of) automation. Although moving up in the abstraction dimension *usually* entails controlling more entities or subsystems, this is not always the case. In our work with Playbook, we have not explicitly broken out the set of entities or subsystems that automation is controlling as a separate dimension. Instead, we've relied on the simplifying assumption that automation could command any equipment to which it had access to to do anything it wanted, within the constraints imposed by the supervisor.

In the proposed AAA framework, however, we will explicitly represent the resources the subordinate has authority to control as the *Aggregation Dimension* since (as in the usage pioneered by Rasmussen, 1984 and Vicente, 1999) we are now referring to the subcomponents that, in aggregate, comprise the whole system. By contrast, in the abstraction dimension, we are identifying the sub-functions that, collectively, achieve higher-level (more abstract) functions. By explicitly representing the aggregation dimension we make it possible to handle situations in which the subordinate does not have complete access to all equipment all the time, but instead may need to ask permission or report usage.

For the moment, it will suffice to represent the Aggregation dimension by simply referencing the specific entities or subsystems in their natural groupings. For example, the specific UAVs that automation has access to, individually and in flights, squads, companies, etc. Subsystems can be represented similarly: for example, the (potentially multiple) sensor systems on board a given UAV and the specific sensors that comprise the overall sensor system.

As with the Abstraction dimension, aggregated resources must be delegated with a level of authority. For this purpose, we will use the same scheme of levels as for the abstraction dimension. For example, automation may be delegated the right to use a specific UAV (or the sensors on that UAV, or a whole squad of UAVs) with Full, Inform, Override, Approval, Recommend, or Monitor authority. Also, as for the Authority dimension above, we can designate default authorizations for the use of specific resources. For example, a subordinate may be authorized to use a specific UAV organic to company A with Full authority, but authorized to use the Battalion UAV only with Approval authority (i.e., ask permission first).

5.4 Summary of the 3D Model of Levels of Automation

Under this model, an act of delegation is a combination of a task, some resources to accomplish that task, and some level of authority to plan and execute that task with those resources. That is, a level of Authority (which may be heterogeneous over subtasks) to perform a task in the Abstraction dimension, combined with a level of Authority to make use of resources in the Aggregation dimension. A Level of Automation is the degree to which authority for performing that task with those resources has been delegated from the supervisor to the subordinate. But since the authority can vary along with the level of the Abstraction and Aggregation dimension, it is important to represent each of these dimensions to characterize the range of possibilities for human-automation interaction. Note that not all of these dimensions may be available or relevant to every system or every interaction, but the model needs to be rich enough to encompass them.

6. Using the Delegation Space Model as an Adaptive Automation Architecture

In the previous section, we laid out a descriptive architecture for the space within which delegation interactions at various automation levels occur. We need to know how to use this framework to control or manage automation behaviors before it will prove useful, however. In this section, we will outline three ways in which the framework can provide utility.

First, we will illustrate its use to characterize and even to predict or drive the creation of alternate forms of human-automation interaction. While this usage does not include an implementation or ability to directly interact with the framework, it nevertheless proves a powerful theoretical tool that could drive research and facilitate understanding—much as Sheridan's and Parasuraman, et al.'s frameworks have in the past.

Second, we will illustrate how the framework could be used, in conjunction with our Playbook, to create a highly flexible adaptive automation experimentation environment. In this approach, we would extend the Playbook to include a flexible range of alternative interaction styles along each of the three dimensions described above. Then we would provide an interface to allow an experimenter (not the operator of the Playbook) to determine which levels of authority, abstraction and aggregation to allow the operator (who serves as the participant in an experiment) to have access to and control over for each different experimental run. This, we believe, is the capability most in keeping with AFRL's goals in this project.

Third, we will illustrate allowing an the operator (rather than an experimenter) to interact with the framework, perhaps in a pre-mission setting, to configure the roles and responsibilities that automation will be allowed to take. This implementation may be identical to that designed for the experimenter above, or it may involve only the redesign of interfaces and/or the selection of a specific subset of most useful functionality. Thus, we may not be constructing two tools so much as simply putting one tool to two somewhat different uses. We suspect that this usage will have more commercial utility than the experimental design application—a useful consideration for a SBIR program.

6.1 Using the Framework to Characterize Different HAI Styles

One use for our framework, like all previous Levels of Automation frameworks, is characterizing various styles of human-automation interaction. This can drive experimentation and understanding even though it may not (immediately) result in tools. Below we present examples of applying the framework to a variety of superior-subordinate interaction types. This framework is richer, more flexible and more specific than previous frameworks such as those proposed by Sheridan and Parasuraman, et al. As we have noted elsewhere (Miller and Parasuraman, 2003; in preparation) this increased flexibility seems to be powerful for specifying delegation interactions.

Unlike Sheridan's (1987) one-dimensional framework ours no longer identifies the level of automation associated with a *system*, but rather the level of automation (actually, level of authority) that a system exhibits with regard to each task or function and with regard to the relevant resources. As a simple example, consider the junk mail deletion macros I can create for my email system. Under Sheridan's (1987) Levels of Automation framework, these might be said to be Level 8 on the scale in Figure 1: 'Computer executes selected alternative and informs human only if asked'. This is a single level of automation at the level of a single aggregate task, and we could label that task something like 'Filter Mail'.

On the other hand, that single `Filter Mail' task exists within an Abstraction dimension. It is a subpart of the more abstract task of "Managing Mail" and it decomposes into a host of sub-tasks, each of them representing an operation that must be performed. These subtasks include items such as (1) 'Detect New Mail,' (2) 'Read Mail Header', (3) 'Evaluate for Kill Criteria', (4) 'Route Message to Delete File' or (5) 'Route Message to Inbox', and when requested (6) 'Display Killed Messages' and (7) 'Display Kill Rules'. Each of these subtasks can have a different level of automation and, in fact, several different levels are represented. For example, the first five tasks above are all automated at at least Level 9 ('Computer executes selected alternative and informs human only if it decides to'), while the latter two are actually closer to Level 2 ('Computer offers alternatives') or even Level 1. Thus, the claim that the parent task is automated at Level 8 is, in fact, a function of the some combining of levels of automation of the lower level sub-tasks. In fact, as we have noted above, Sheridan's spectrum is primarily a parsing of the Authority dimension, thus the claim that the mail filtering system operates at Sheridan's level 8 seems to mean simply that it performs a task or function at a certain level of Abstraction with about that average level of Authority. In fact, it actually performs a variety of subfunctions at a variety of levels of Authority.

Similarly, when one uses a levels x stages approach as in Parasuraman, et al. (2000), one is clustering the subtasks by information processing stage, and again averaging. For example, 'Detect New Mail' is a type of Information Acquisition task, but 'Read Mail Header' and 'Evaluate for

Kill Criteria' are both Analysis tasks. To assign a single level of automation to an information processing task or phase offers more sensitivity than assigning it only to the parent task, but this is still a clustering. In practice, one could identify the specific sub-tasks to be performed and represent an automation (actually, authority) level for each of them.

In the above example, there is also some variation in the Aggregation dimension. We could explicitly declare a variety of Aggregation levels across the subtasks by simply identifying the resources that the automation is authorized to use, and the level of that authorization, for each of them. Tasks 1 and 2, for example, require the ability to inspect the mail stream. Task 3 requires the ability to read a file of kill criterion rules and the computer processing cycles to evaluate them. Tasks 4 and 5 require the ability to change the location of items from the mail stream, while tasks 6 and 7 require the ability to display information to a screen. In each case except tasks 6 and 7, the automation is presumed to have Full authority to manage the necessary resources in accordance with the task it has been delegated. Tasks 6 and 7 are more restricted, however. For them, the system can only exert control over the display screen if requested by the user. In this sense, it has no (i.e., "None") authority over the screen for the performance of these subtasks.

We can envision how various hypothetical mail filtering systems might be characterized in our framework. For example, let's say that instead of automatically detecting new mail, the system had to be "called" and specifically requested to apply its kill rules to a set of new mail messages. It's unclear how Sheridan's framework would characterize this new and less autonomous system—perhaps now as operating at level 5 ('Computer executes alternative if human approves') because the human is required to specifically tell it to execute its filtering actions. This is fine, but it obscures the fact that the automation is still operating exactly as before for most of its subtasks and only the initiating task has changed. Parasuraman, et al.'s framework would, presumably, identify the fact that the level of automation of the Information Acquisition portions of the system has been lowered. Although more precise than Sheridan's, the Parasuraman, et al. framework still obscures the precise change that has occurred.

Our framework, by contrast, would more precisely note the differences between the two human-automation interaction styles for the two different systems. We would note that the level of Authority over the Detect New Mail subtask (a specific point in the Abstraction dimension) has changed from Full or Inform to None (the system does it only when explicitly told to). Furthermore, this task now has a slightly different resource authorization in the Aggregation dimension. Instead of being applied to any and all new mail items in the mail stream, the supervisor is only authorizing it to be performed with regard to specific, designated mail items.

Finally, an example related to the Overfly play in a UAV control task such as MIIIRO affords may bring the model usage closer to home. We could posit one human-automation interaction system that allowed the human superior to task UAVs by providing waypoints alone and asking for a suggested method of flying them (i.e., in terms of speed, control settings, etc.) and then authorizing the execution of the resulting plan, and another that permits tasking them via the Overfly play alone and asking for a route and sensor actions to achieve it and authorizing the execution of the result. Our intuition tells us, quite rightly, that the latter system provides a higher "level of automation" than the former—but what do we mean by that? It is hard to characterize the difference between the two systems using Sheridan's spectrum of levels. In each case, the automation does what it is told to do—the authorization actions between the human and the automation are essentially the same. We could characterize this as Sheridan's level 5 ('Computer executes alternative if human approves') but this obscures the fact that the human is dictating the alternative and telling the system to execute it in both cases, *but is doing so at different levels*. Parasuraman, et al.'s framework does slightly better—it seems clear that the additional automation occurs within the Decision Selection portions of the overall task, but this gives us only an imprecise notion of exactly what has changed between the two implementations. Our framework allows us to dictate, specifically, what has changed: that the level of authority is the same (Recommend) across the two implementations, but the level of Abstraction has changed (from the lower level Fly-to-Waypoint tasks to the higher level Overfly task). The operator can now command/delegate at a higher level of abstraction—and doing so means delegating more authority over the lower level actions (waypoint selection).

6.2 Using the Framework as an Experimentation Control Tool

The theoretical approach outlined above can be applied as an experimenter's control tool to allow flexible and precise control of the kind and level of automation an operator can access for each experimental trial. There are at least two ways this can work. First, simply by aggregating low level behaviors (like waypoint following) and providing the planning intelligence to assemble them into contextually-appropriate, higher level, more complex and abstracted behaviors, Playbook *gives* the user another level (of abstraction) at which to command automation. This translates into what, in common parlance, is another, higher "level of automation". Playbook introduces another level of autonomy (actually, level of abstraction) into the system.

In previous Playbook implementations (and in the integration of Playbook with MIIIRO in Phase I), the operator can flexibly decide to command at various levels (of abstraction). It may be comparatively easy to create an experimenter's interface to provide the ability to lock the operator out of one or more of these levels of abstraction on a task by task basis. This would allow, for example, creating an experimental condition in which the operator only can only command Overfly plays, or only Fly-to-Target or Fly-to-Waypoint subplays (i.e., command by inserting targets or individual waypoint sequences), or only Flight Control actions (via joystick controls), or has flexibility over only Overfly and waypoints (but not joystick commands), etc.

But if this were all we did, we would only afford experimenter control over the abstraction dimension. We can do more—providing control over authority and aggregation dimensions as well. To accomplish this, and thereby provide a very rich framework for experiments on adaptive automation, we will need to add two capabilities to the current Playbook.

First, when one currently delegates a play or function to automation using Playbook, one is authorizing Playbook's planning component to develop a plan for achieving that function and then submit it to the operator for approval. If the operator approves[4], Playbook executes the plan. This interaction style represents an "Approval" level of authority for the delegation of tasks at the "Overfly" level of abstraction[5]. Note that when tasks are delegated at the Overfly level of abstraction, subtasks below that level have equal or higher levels of authority by default (the Fly-to-Target subtask is fully planned by the automation and, if execution authorization is given to the parent Overfly task, Fly-to-Target will be performed with Full authority). The implemented system does not currently allow additional tasking at the next deeper level of abstraction (e.g., by providing Achieve Airborne, Fly-to-Target, etc. level commands), but if it did, Approval level

[4] Our current Playbook implementations offer very little capability for the operator to modify a plan suggested by the Playbook Planner. This is a capability we would like to add in the future.

[5] In fact, we have altered this slightly in some Playbook implementations—primarily through giving Playbook to right to automatically begin execution of the plan it creates, while retaining the right to override it once it has begun (Override authority level).

authority might also be appropriate there. Using the inherited waypoint commanding capabilities of MIIIRO, Fly-to-Waypoint level tasking is currently done with Override authority (once given the waypoint, the automation determines how to fly to it and automatically begins doing so, but the operator can jump in an "override" automation's decisions by inserting new waypoints or providing joystick commands). When the operator is providing Flight Control Actions (via joystick), s/he is operating at the lowest abstraction level of human control in the system—the automation has Full authority about how to plan and execute the commanded Flight Control Actions. This phenomenon of increasing authority to automation the lower you go in an abstraction hierarchy is something we expect to see frequently.

These authority levels are fixed in the current system, but they need not be. Figure 4 gives an indication of some of the options that setting alternate authority levels within the abstraction dimension could afford. To accomplish an Overfly task/function, four subtasks must be accomplished. Currently, commanding at the Overfly level delegates the same level of authority to all subtasks (Approval), but this needn't be the case. An experimenter could configure an experimental run in which the operator delegates Full authority to Achieve Airborne (e.g., the automation can make it's own decisions and execute them, without further operator review or approval, about how to get an suitable aircraft airborne), Approval authority to Fly-to-Target, Monitor authority for Photograph Target, and Inform authority for Destage—or any other combination of tasks x authority levels that made sense, was feasible within the capabilities of the automation and was of interest to the experimenter.

Although there is a tendency for delegated authority to increase at the lower levels of the task hierarchy, this need not be uniformly the case. For example, let's say that there is a specific sub-subtask, deep within the definition of Photograph Target (perhaps the use of an active imaging device like Radar that might compromise stealth). The automation could be given Full authority to plan and execute Overfly tasks—including all possible methods of accomplishing that parent task—*except* for this specific "Activate Radar" task. That task could be configured to have nothing higher than Approval authority (or even, perhaps, Monitor authority if it were an action the user had to take). In this case, Playbook would behave fully autonomously whenever it was asked to perform an Overfly play in all cases except those where it deemed the use of the Activate Radar subsubtask desirable. In those cases, it would ask the user for approval to perform that specific task[6].

The above example illustrates the integration of Abstraction and Authority dimensions; a similar approach could work, together or separately, with the Aggregation dimension. Instead of associating a level of authority with a specific task (a specific part of the Abstraction dimension), it could work by asserting an authority level on a specific *resource* (the Radar system) and therefore be asserted against the Aggregation dimension.

Another example illustrates the integration of Authority and Aggregation dimensions in the MIIIRO domain. MIIIRO currently operates as if the automation is authorized to use any UAV with Full authority. But this behavior could be varied along both the Authority and Aggregation dimensions. Playbook could be tasked to develop and execute Overfly plans using any UAV with Full authority—or it could use any UAV with Approval authority, or it could use UAVs 1-3 with Full authority, but require Override authority for UAV 4, etc. Or we could move down an Ag-

[6] Note that although I've tried to describe this task as relevant to the surveillance domain that is represented by MIIIRO, it obviously covers the weapons release authorization issue that is a hotly debated topic with regards to proposed UCAV operations.

gregation level and apply these authority levels to specific UAV subsystems. Automation could (within the tasks and constraints delegated to it) be permitted to task navigation systems with Full authority, but sensors with only Monitor authority, or could be allowed to develop plans for all systems on all UAVs with Full authority—except for that throttle we've been having problems with and don't want to overstress on UAV 3. Finally, as an alternate method of achieving the same behavior as described above, where we authorized an Activate Radar task with Approval authority, we could similarly have authorized the use of the Radar subsystem with Approval authority. Again, a rich Experimenter's interface could allow the experimenter, using Playbook, to activate any combination of these features desired for individual experimental runs.

Below we provide an illustration of a portion of this functionality in a Playbook-based adaptive automation architecture. This description should be viewed as an exercise in applying the AAA framework described above and not (yet) as a description of proposed work. We need to evaluate each of the hypothetical combinations of Abstraction, Aggregation and Authority levels and see if they (a) make sense as a practical and reasonable distinction for the Playbook and potential operators to use, and (b) are technically feasible within the constraints of the MIIIRO testbed and the Playbook software we can develop. Furthermore, while this example uses only the simplified OverFly play, many other plays are possible and have been discussed during Phase I of our effort and might be implemented in Phase II.

6.3 Potential Demo Using OverFly

6.3.1 Phase I Capabilities in Terms of AAA Framework

At the end of Phase I, we will demonstrate two levels of abstraction: waypoint control and Overfly control (of single UAVs). In our model, one should not think of these as play control and non-play control, since they are both plays at different levels of abstraction (Fly-to-Waypoint vs. Overfly). We will not yet provide variation in authority or aggregation dimensions. The operator will have the ability to choose the level of abstraction at which to delegate. The level of aggregation is the single UAV with Approval authority (the Playbook's Planning and Analysis Component (PAC) can select any of the available UAVs, unless the delegation instructions constrain that choice, and task them to perform its plan—though that plan has to be submitted to the operator for approval). The level of authority that the PAC has for the subtasks when the operator commands plays at this Overfly level have been largely implemented as matters of convenience (based on what was available and achievable in the integration of MIIIRO and Playbook). For Achieve Airborne, the PAC has Approval authority (within the constraints imposed by the operator) since the operator must review and approve PAC's choice of UAV and method of getting it airborne[7]. The Fly-to-Target and Destage tasks are both comprised of a series of waypoints. Again, the PAC has Approval authority over these tasks—it can create a route within the constraints imposed by the operator but must submit it for approval prior to executing it. The Photograph Target task seems to have Full authority in the current implementation—the PAC has no need to submit it to the operator for approval prior to execution if the operator delegates an Overfly task.

The user also has the opportunity to task automation at the Fly to Waypoint level (which effectively skips a level in our Playbook task abstraction hierarchy). When the operator does this, s/he

[7] Note that there is no current option for this task—all aircraft begin on the ground so the only available method of performing Achieve Airborne is by selecting one and having it take off.

is now retaining authority for the parent tasks (Overfly and it's four children) although s/he is delegating the actual flight control commands to the automation with Override authority (since, via joystick activation, s/he can still intervene).

6.3.2 Enhancing Abstraction Flexibility (and Putting it Under Experimenter Control)

To make this framework a reality in the TUSC project, we would implement portions of the AAA framework as follows. First, we provide a UI, structured around a task tree expansion of the underlying Playbook Task Model, to enable an *experimenter* to selectively "turn on and off" levels of the abstraction hierarchy. This allows experimenters to allow operators to command either only Overfly plays, only waypoint commands or, with additional work, the intermediate level subplays. This interface also allows a range of flexibility to command across the abstraction levels—perhaps, Overfly plays and intermediate subplays but not waypoint commands, etc.

Next, we add some alternate (and experimenter selectable) levels of authority to these abstraction levels. Providing multiple authority levels at the Overfly abstraction level should be comparatively straightforward.

- ❑ Full authority simply means allowing the PAC to begin execution of its preferred Overfly plan without either informing or seeking permission from the user.
- ❑ Inform authority also allows immediate execution, but requires informing the user of the selected plan (a capability we already provide, to at least some level of utility).
- ❑ Override authority would mean allowing the PAC to begin execution while informing the operator, but allow the operator the ability to jump in and override portions of the PAC's plan. A trivial implementation of this capability would be to allow the operator to input novel waypoints or to "grab a joystick" and begin flying the aircraft. The issues which remain to be clarified are whether or not it is technically feasible for the operator to override only a portion of the play and allow the rest to remain intact. This is challenging and may not be feasible—the existence of a level of authority in no way guarantees that it is desirable or feasible.
- ❑ Recommend authority would be achieved by allowing the PAC to present a recommended plan, but requiring the operator to manually fly it, and
- ❑ Monitor authority would involve the PAC monitoring the human's plan and execution of it for critical parameter completion such as time to target and fuel usage.

We provide the experimenter the ability to configure which of these authority levels is available to the operator for each experimental run—for example, allowing the operator Approval authority in one run, and only Monitor authority in the next. Allowing the experimenter the ability to provide the operator flexible control across authority levels is more difficult, since it entails determining which levels of authority are supportable in MIIIRO/Playbook, but is clearly desirable to the extent we can determine the appropriate levels and offer them to the experimenter.

6.3.3 Enabling Authority Flexibility Over Abstraction Levels

Providing authority flexibility at the next level of the task abstraction hierarchy might be more difficult and/or less interesting in the current MIIIRO implementation:

- The Achieve Airborne task currently permits only one method/task/subplay—getting a UAV to take off.[8] Full, Inform, Approve and Override authority levels for that choice could be easily implemented. Recommend authority could be a recommendation for which UAV to use but not (unless authority levels on other subplays at this level permitted) providing detailed plans for how to use it (in terms of routes). Monitor authority could be the PAC monitoring the operator's choice of UAVs and critiquing problems such as ability to arrive on target or back to base within time or fuel constraints.

- The Fly-to-Target and Destage tasks both involve selecting waypoints for those legs of the mission. For the operator to delegate these tasks to automation with Full authority would simply mean that the PAC gets to plan and execute these legs of the mission without informing or seeking approval. Inform, Override and Approval authority all work similarly, with the caveat (as noted above) that Overriding a mission route in progress (either by inserting new waypoints or by grabbing the stick) may mean invalidating the plan that the PAC had and, thereby requiring the human to take over the remainder of the mission. Recommend authority would mean providing a route recommendation for that leg, and Monitor authority would mean allowing the PAC to monitor the operator's performance for parameters such as leg time and fuel usage constraints. A more elaborate user input capability would allow the operator to input a proposed route for this leg and receive feedback or critique on its feasibility and suitability.

- It is currently unclear what alternative methods exist for performing the Photograph Target task. If there is no ability to perform this task in alternate manners within MIIIRO (and no intention to provide such variance), then there are no decisions to be made and it makes no sense to allow anything other than Full or, more in keeping with current MIIIRO practices, None authority to automation for this task. (A reason to do so might be if execution of the task were something which Monitor level automation could aid the human in doing or Override level authority could improve performance by allowing human intervention). Similarly, if MIIIRO does or were to support variations in performing this task, perhaps through providing access to various sensors, imaging positions or sensor settings (i.e., FOV), then various levels of authority in performing this task might be feasible and interesting.

6.3.4 Enabling Aggregation Flexibility (and Authority Over It)

Aggregation works analogously to Abstraction. An early step would be to enablethe assertion of variable authority levels (by the experimenter) on the use of specific UAVs. For example, the PAC can currently use any UAV it desires (that has not already been tasked) with Override authority, but we could configure this so that Full, Inform and Approval authority levels are settable—with regards to all or to any specific UAV. Recommend authority could be achieved by allowing the PAC to recommend UAV usage, and Monitor authority would allow the PAC to critique the operator's choice of UAVs. We could provide the ability to set these authorization levels, against all or against specific UAVs, as a part of the experimenter's UI.

To the extent that there are existing groupings of resources, automation can be granted flexible Authority for any group. To the extent that it is interesting, new groupings can be defined, analo-

[8] Note that the limitation to UAVs on the ground is a limitation only of the current MIIIRO simulator and the Playbook/MIIIRO interaction protocol; this limitation could and should be lifted in Phase II.

gous to the insertion of new levels of abstraction in the task hierarchy. Next, we would begin by adding multiple levels to the aggregation dimension. UAVs could be organized into teams and subdivided into individual subsystems (e.g., sensors, flight controls, engines, etc.). This would enable the experimenter to set an authority level for the PAC's usage of groups of UAVs, or of specific subsystems of all or any specific UAV. Of course, not all combinations of aggregation levels and plays may make sense, or be technically feasible. In particular, it might make little sense to allow PAC to task multiple UAVs as a group within the current Overfly play—which only makes use of a single UAV. But additional plays currently under consideration, including superplays involving the coordination of multiple Overfly plays, will make this more reasonable.

6.3.5 Summary of Overfly Example

In using this architecture with even the comparatively simple Overfly play, we have defined at least three levels of Abstraction (Overfly Play, Intermediate subplays, waypoint-level sub-subplays) and seven levels of authority on the abstraction levels. We have also identified at least three levels of Aggregation (UAV groups or teams, individual UAVs and UAV subsystems) again crossed by seven levels of authority on the aggregation levels. This could potentially provide for a total of 3x7x3x7 (or 441) different human-automation interaction styles within the MIIIRO testbed. While not all of these levels may be technically feasible or worth investigating, this should serve as an illustration of how the proposed, Playbook-based AAA would allow the *systematic* investigation of a huge range of alternate human-automation interaction styles.

6.4 Using the Framework to Configure Human-Automation Roles and Responsibilities

Above, the assumption was that the power to turn on and off different abstraction, aggregation and authority levels for the operator was to be placed in the hands of an experimenter. We can foresee, however great utility to changing that arrangement and giving similar control to the operator him- or herself. To do this, we would provide an interface for the operator to, probably pre-mission, configure the permissions with which automation may operate when asked to plan and/or execute a play. Instead of the experimenter saying to the PAC automation, in effect, "this play, or UAV may only be used if you ask the operator first, but these others you can execute immediately"—we would be allowing the operator to make the same kinds of statements and thereby, to establish an a priori set of boundaries according to which automation must behave.

Such an approach would likely necessitate a different user interface than the experimenter's control station we envisioned above, and it might benefit from having a reduction in the degree of flexibility (and therefore, complexity) compared to what we outlined for the experimenter. It would, however, represent a significant enhancement to the Playbook architecture we have developed to date and, we suspect, would have substantially greater commercialization potential. There are, after all, a great many more potential customers in the world who need to be able to interact with automation in a way that is both easy and yet predictable than there are who need to perform a wide range of experiments on human-automation interaction.

7. References

Miller, C. and R. Parasuraman (2003). "Beyond levels of automation: An architecture for more flexible human-automation collaboration," in *Proceedings of the 47th Annual Meeting of the Human Factors and Ergonomics Society*, (Denver, CO), Oct. 2003.

Miller, C. and Parasuraman, R. (in preparation). Designing for Flexible Interaction Between Humans and Automation: Playbook Delegation Interfaces for Supervisory Control. To be submitted to *Human Factors*.

C. Miller, "Delegation architectures: Playbooks & policy for keeping operators in charge," in *Proceedings of the NATO Research and Technology Organization Special Workshop on Uninhabited Military Vehicles*, (Leiden, Holland), June 2003.

Morrison, J. (1993). "The adaptive function allocation for intelligent cockpits (AFAIC) program: Interim research and guidelines for the application of adaptive automation," Technical Report NAWCADWAR-93031-60, Naval Air Warfare Center.

Parasuraman, R., T. Sheridan, and C. D. Wickens, (2000). "A model for types and levels of human interaction with automation," *IEEE Trans. Systems, Man and Cybernetics, vol. 30*, pp. 286–297.

Rasmussen, J., Pejtersen, A. and Goodstein, L. (1994). *Cognitive Systems Engineering*. New York; Wiley.

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

Sheridan, T. and W. Verplank, (1978). "Human and computer control of underea teleoperators," Technical Report, MIT Man-Machine Systems Laboratory, Cambridge, MA, 1978.

Vicente, K. (1999). *Cognitive work analysis*. Mahwah, NJ; Erlbaum.