A Testbed for Investigating Task Allocation Strategies between Air Traffic Controllers and Automated Agents

Nathan Schurr[1], Richard Good[1], Amy Alexander[1], Paul Picciano[1], Gabriel Ganberg[1], Michael Therrien[1], Bettina L. Beard[2] and Jon Holbrook[3]

[1]Aptima Inc., 12 Gill Street, Suite 1400, Woburn, MA 01801
[2]NASA Ames Research Center, Moffett Field, CA 94035
[3]San Jose State University Research Foundation, San Jose, CA 95192
{rgood, nschurr, aalexander, ppicciano, gganberg, mtherrien}@aptima.com, {tina.beard, jon.holbrook}@nasa.gov

Abstract

To meet the growing demands of the National Airspace System (NAS) stakeholders and provide the level of service, safety and security needed to sustain future air transport, the Next Generation Air Transportation System (NextGen) concept calls for technologies and systems offering increasing support from automated systems that provide decision-aiding and optimization capabilities. This is an exciting application for some core aspects of Artificial Intelligence research since the automation must be designed to enable the human operators to access and process a myriad of information sources, understand heightened system complexity, and maximize capacity, throughput and fuel savings in the NAS.. This paper introduces an emerging application of techniques from mixed initiative (adjustable autonomy), multi-agent systems, and task scheduling techniques to the air traffic control domain. Consequently, we have created a testbed for investigating the critical challenges in supporting the early design of systems that allow for optimal, context-sensitive function (role) allocation between air traffic controller and automated agents. A pilot study has been conducted with the testbed and preliminary results show a marked qualitative improvement in using dynamic function allocation optimization versus static function allocation.

Introduction

According to the Federal Aviation Administration, "NextGen is an umbrella term for the ongoing, wide-ranging transformation of the United States' national airspace system." Fundamental to the Joint Planning and Development Office's (JPDO) definition of the Next Generation Air Transportation System (NextGen) is the notion that automated systems must coordinate with the human operator to "take advantage of the functions each can best perform" (JPDO, 2007). Under this specification, automated tools should be designed to support the optimal allocation of tasks between the system and the human operators using these systems. It follows that function allocation strategies among operators and automated systems must be studied early in the design process to ensure that the impact of these function allocations can be fully realized in an implemented system.

Function allocation strategies can be either static or dynamic. Static FA strategies assume that the automation and human will maintain unchanging roles and responsibilities. Dynamic FA strategies adjust seamlessly and appropriately as required to balance workload and capitalize on the human's and automation's strengths. Here we present a testbed, referred to as the Airportal Function Allocation Reasoning (AFAR) testbed, for investigating the critical challenges in supporting the early design of systems that allow for optimal, context-sensitive function allocation between human air traffic controller and automated systems.

AFAR applies the Artificial Intelligence research areas of mixed initiative (adjustable autonomy), multiagent systems, and task scheduling to the NextGen aviation environment. The testbed is designed to investigate the interaction between automation and human operators, specifically, adjustable autonomy between air traffic controllers managing aircraft on the airport surface (ground controllers) and automated agents capable of performing a subset of those associated tasks. Because the NextGen concept of operations (ConOps) involves new classes of functions and responsibilities, with relatively unstudied consequences, the testbed allows for assessing task allocation performance in terms of its adaptation to previously unplanned-for scenarios and changing ConOps. The intention is for early testing and analysis of human-system interaction concepts with dynamic function allocation that adjusts to the evolving goals, demands, and constraints of the operational environment. Consequently, the testbed uses dynamic function allocation strategies

(DFAS) that are represented by allocation policies that vary over time and circumstances. By casting function allocation as a problem of adjusting autonomy between human-automation (and human-human) links, one can leverage the current techniques for optimizing both strategy selection and strategy timing.

Related Work

One of the most fundamental challenges of building a human-multiagent team is that of "adjustable autonomy" (Reitsema et al. 2005); (Schurr, Patil, and Pighin 2006); (Scerri, Pynadath, and Tambe 2002); (Sellner et al. 2006); (Sierhuis et al. 2003); (Varakantham, Maheswaran, and Tambe 2005). Conventionally, adjustable autonomy refers to the ability of an agent to dynamically adjust its own autonomy, thereby altering the amount of control a human has over the agent at a particular time and context. Given the changing state of the environment and the team, it is beneficial for an agent to be flexible in its autonomy and not be forced to act either with full autonomy or with zero autonomy. The key to adjustable autonomy is when and where to transfer control of a decision. For example, much work has focused on adjustable autonomy in robotic assets to better cope with the unpredictable nature of military environments (De Visser et al. 2006); (Freedy et al. 2008); (Parasuraman and Wickens 2008). In this case, a multi-agent planning tool based on a proxy framework with adjustable autonomy is used to facilitate human-robot task allocation. The overall approach is to enhance human-robot team effectiveness by matching function allocation to contextual and situational demands.

In addition to adjustable autonomy, extensive work has been done in the last decade to evaluate the intricate interactions between automation and the human operators (Parasuraman and Riley 1997); (Parasuraman, Sheridan, and Wickens 2000). These solutions help support designers in developing systems and policies to exploit the capabilities of both human operators and automated systems. Computational and formal models have also been recently developed including automation verification (Degani and Heymann 2002), Bayesian modeling approaches (Sheridan and Parasuraman 2000), and mathematical models of automation reliance and compliance (Dixon and Wickens, 2006).

The Air Traffic Control Use Case

The motivating example for the AFAR testbed involves human and automated agents sharing ground control responsibilities at Dallas/Fort Worth International Airport (DFW). The use case was developed based on contributions from three sources: (1) interviewing experienced pilots, (2) visiting DFW and Boston Logan's control towers, and (3) interviewing experienced air traffic controllers. The resulting scenario consists of approximately two dozen aircraft at DFW over a period of approximately 30 minutes. During the scenario the surface operations duties were shared between a human ground controller and an automated agent. Duties were shared for managing traffic and ensuring safe taxi coordination for all aircraft. Responsibility for a given aircraft concluded when it was handed over to the local controller (a responsibility coordinated by a testbed agent) for departure or was parked at the designated terminal gate for arrivals. The scenario involved the management of arriving and departing traffic on the east side of DFW. Arrivals were inbound on Runway (RWY) 17C, and departing aircraft took off from RWY 17R. This configuration created a need to cross arriving traffic over the active departure runway. Departing and arriving aircraft were assigned terminal gates, destinations, squawk codes, airframe types, and call signs. The airport configuration included the designation of standard departure and arrival routes, as well as several predetermined taxi routings.

The simulation involved a ground controller position as the single human-in-the-loop participant. Researchers served as pseudo-pilots (one for departure and one for arrivals) managing all scenario aircraft to provide communications and manipulate aircraft taxi behavior as dictated by either the ground controller or automated agent. In order to manipulate the workload, attention allocation, and task prioritization, a number of experimental "events" were inserted during each run. These ranged from small perturbations such as intentional read-back errors to more demanding situations such as a pilot reporting from near the departure queue that they do not yet have their weights/balances for takeoff. The intent was to maintain realistic demands and interrupts that are likely to occur in operations. The events were collected through interviews with commercial pilots and controllers to validate these situations.

The AFAR Testbed

The AFAR Testbed consists of four main applications: the AFAR JADE Agents, the AFAR User Interface, the DFAS Task Allocation Engine, and Aptima's Distributed Dynamic Decision-making (DDD) simulation environment. While running experiments, the air traffic controller participant views the ground controller user interface and interacts verbally with experimenter confederates (or pseudo-pilots) who play the role of the arriving and departing aircraft pilots. These pseudo-pilots view and interact with a different set of user interfaces. Meanwhile, the Decision Maker Proxy Agents monitor the completion of tasks by the ground controller participant and report progress to the Function Allocation Agent. The Function Allocation Agent generate workflows based on the DFAS algorithm that specifies whether tasks should be

performed by the ground controller participant, or by automation. If the task is assigned to the ground controller, the ground controller participant is required to initiate communications with the pseudo-pilots to complete the task. If the task is assigned to automation, the pseudo-pilots complete the task without requiring input from the participant. A schematic showing the interactions of these components is shown in Figure 1 followed by a discussion of each of these components in more detail.

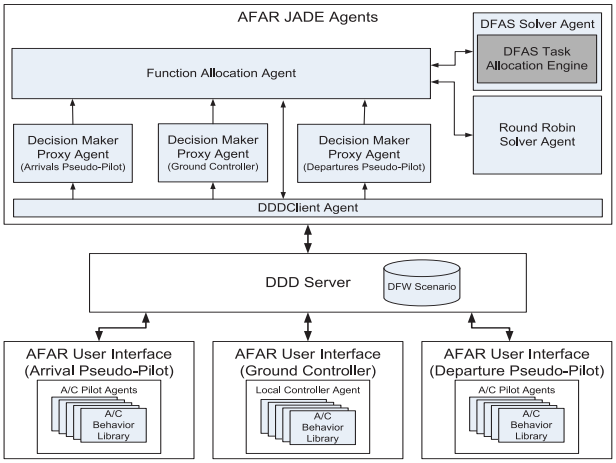

Figure 1: The AFAR Testbed architecture consists of four main components: AFAR JADE Agents, AFAR User Interface, DFAS Task Allocation Engine, and the DDD server.

Agent Design

There are two distinct agent architectures used in the AFAR Testbed. The AFAR JADE Agents are based on the Java Agent Development framework. They consist of four types of agents: Function Allocation, Decision Maker Proxy, Solver, and DDD Client agents.

The Function Allocation Agent generates work flows and adds them to the global task queue (i.e. generates an arrival work flow when it's informed a plane has arrived). It also maintains a list of active Decision Maker Proxy agents and their capabilities and assigns tasks to decision makers (e.g. Ground Controller or automation) based on input from the Solver Agents and the state of the global task queue.

Decision Maker Proxy agents come in two varieties, one that represents a human and one that represents an automated system. Decision Maker Proxy agents are responsible for detecting task completion and reporting progress to the Function Allocation Agent.

Solver Agents are responsible for scheduling and assigning tasks. There are two different solver agents for the two conditions in the pilot study. The DFAS Solver Agent applies the Dynamic Function Allocation Strategies algorithms when scheduling tasks and the Round Robin Solver applies a simple algorithm that statically distributes tasks between the Ground Controller and automation.

The DDD Client Agent is responsible for subscribing to relevant DDD Simulation events and provides the Function Allocation and Decision Maker Proxy agents with the necessary information to accomplish their goals.

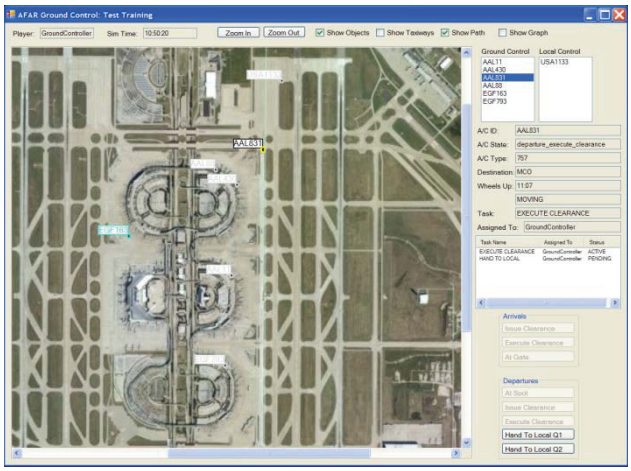

Figure 2: The AFAR interface enhanced with dynamic map

The second distinct agent architecture that the AFAR Testbed uses is the DDD Agent Framework, a DDD-specific agent framework that is used to develop aircraft pilot agents and an automated Local Controller agent. In general, the goal of these agents is to take the place of all the other decision makers and actors that would normally be involved in an air traffic control environment. The pilot agents monitor their environment using the AFAR interface (shown in Figure 2) to avoid collisions and can take taxi instructions from the Ground Controller through the pseudo-pilots. When given taxi instructions, they can navigate through the airport following the appropriate routes. The automated Local Controller agent serves as the owner of all aircraft when they are not directly under the Ground Controller or automation's control. It gives instructions to the pilot agents during landings until they are handed off to the Ground Controller or automation, and handles the takeoff of departing aircraft.

Dynamic Function Allocation Strategies

The DFAS approach assumes that functions will be allocated across heterogeneous decision makers, where a decision maker is either an automated system (agent) or a human operator. As the name suggests, the two main characteristics of the function allocation techniques are that they must be dynamic (functions allocated based on real-time state of the world, not static rules) and they must be strategic (functions are not only allocated for the current state of the world, but are predictive over a scheduling horizon). The role of the DFAS component is to determine the task assignment between human air traffic controller(s) and automated agents. As such, DFAS operates as a task

allocation scheduler in which DFAS determines which resource is most appropriate for what task(s) and when given the state, user preferences, and constraints in the operational environment. Some of the unique constraints and preferences that the DFAS engine must consider when creating a task assignment schedule, irrespective of the DFAS scheduling algorithm include: task quality and duration, the agent's real-time workload and ability to multi-task, task precedence (order), penalties from switching tasks between agents, tardiness penalties, rescheduling frequency, and algorithm execution time.

The preferences and constraints listed above are used as inputs into the DFAS engine as well as the world state. The world state (managed by the task manager) refers to the current list of tasks to be assigned, which tasks are currently assigned to whom, the tardiness of the tasks, and any available insight into the state of the task (for example, how close is a task from being completed). The output of the DFAS engine is a schedule for each of the agents (who should be doing what and when).

DFAS Solution Methods. Due to the discrete decisions involved, the task allocation problem is inherently combinatorial in nature and, therefore, very challenging from a computationally expense perspective. More specifically, the problem falls into a class of NP-complete problems (Garey and Johnson 1979). Many approaches have been proposed for solving the task allocation problem including utilizing Markov Decision Processes (MDP; Dolgov and Durfee 2006), discrete (DTS) and continuous time schedulers (Floudas and Lin 2004), and myriad heuristic methods (Pinedo 1992). Because discrete-time schedulers are able to formulate constraints and objectives in a relatively straightforward manner and globally optimal results can be reached within a reasonable execution time, the authors have chosen to first implement the DTS approach.

Discrete-Time Scheduler. The discrete-time scheduler's unique characteristic is that it breaks up the scheduling horizon into uniform time durations (often called shifts). A schematic of the DTS approach is shown in Figure 3 where the goal is to assign agent (*i*) to task (*j*) during shift (*k*). If an agent is assigned to a task then a fraction of the task is consumed during the shift (according to the task's duration). The fraction of the incomplete task remains for the agents to operate on during the following shift (*k*+1). To illustrate this concept, consider the case where a task's duration is twice the shift interval: half of the task can be completed during shift, *k*, and half of the task remains for shift, *k*+1. Resources are allocated to tasks until none of the task is remaining.

An exemplary formulation of the DTS uses a mixed integer program (MIP) of the following form,

$$\min_{X,Y,Z} J = \sum_{I,J,K} C_{i,j}X_{i,j,k} - \sum_{J,K} Q_{i,j}X_{i,j,k} + \sum_{J,K} P_j Y_{j,k} + \sum_{I,J,K} SV_{i,j,k} \quad (1)$$

subject to:

$$Y_{j,k} \geq Y_{j,k-1} - R_{i,j}X_{i,j,k} \quad \forall i = 1 \dots I,\ \forall j = 1 \dots J, \forall k = 1 \dots K \quad (2)$$

$$\sum_i X_{i,j,k} \leq 1 \quad \forall j = 1 \dots J, \forall k = 1 \dots K \quad (3)$$

$$X_{i,j,k} \leq 1 - Y_{j',k-1} \quad \forall i = 1 \dots I,\ \forall j = 1 \dots J, \forall k = 1 \dots K \quad (4)$$

$$V_{i,j,k} \geq X_{i,j,k} - X_{i,j,k-1} \quad \forall i = 1 \dots I,\ \forall j = 1 \dots J, \forall k = 1 \dots K \quad (5)$$

$$X \in (0,1), 0 \leq Y \leq 1, 0 \leq V \leq 1. \quad (6)$$

The objective function in (1) shows that the MIP has three types of variables: *X*, *Y*, and *Z*. $X_{i,j,k}$ is a binary variable that indicates if agent *i* is assigned to task *j* during shift *k*. The continuous variable $Y_{j,k}$ indicates how much of task *j* remains unfinished after shift *k*. $V_{i,j,k}$ is the change in task assignment between shifts *k-1* and *k*. The objective function in (1) also shows that the MIP is balancing four objectives: minimize the cost of a task assignment (weighted by *C*), maximize the quality of the task assignment (weighted by *Q*), minimize the tardiness of a task (weighted by *P*), and minimize the change in an agent's task assignment (weighted by *S*). The constraints in (2) through (6) are explained below:

- Eq. (2) is the task consumption balance. The amount of task *j* remaining at shift *k* is consumed by the agent assignment *i* where the task is consumed by at the rate $R_{i,j}$ tasks per shift. The rate is the shift duration divided by the task duration.
- Eq. (3) specifies that a task can only be assigned to one agent.
- Eq. (4) specifies the precedence. An agent cannot be assigned to task *j* in shift *k* unless the preceding task, *j'*, is completed at shift *k-1*.
- Eq. (5) is the increase in task assignments between shifts.
- Eq. (6) indicates that the task assignments *X* are binary variables and *Y* and *V* are continuous variables.

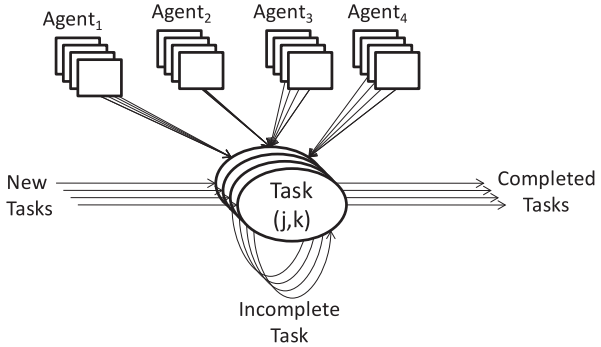

Figure 3: The discrete-time scheduler determines which agent is assigned to what task and at what time. Tasks are divided into complete and incomplete portions.

To illustrate the algorithm, the task flows for two events have been generated. The example includes four agents each of which is competent in a subset of the tasks and can accomplish the tasks at different durations. In addition, workloads, quality, and cost values has been assigned to each of the competent agent/task duples. The scheduling horizon was set at 15 minutes and the time interval was set at 15 seconds for a total of 60 shifts. Figure 4 illustrates the remaining fraction after each of the shifts in which green indicates that 100% of the task is remaining. As the task is completed over time, the color shifts to red. One can clearly see the propagation of tasks. As one task is finished, the agents can start working on completing the following task(s). We also see in Figure 4 that several of the tasks remain incomplete at the end of the planning horizon. As the horizon recedes and tasks are completed, these tasks will be included in the scheduling horizon at the next reschedule.

Figure 4: The amount of each task remaining at the end of each 15-second shift.

The task assignment is shown in Figure 5 where a task assignment is indicated with a green dot. Most of the time an agent is assigned to a single task. However, occasionally agents are assigned to multiple tasks if the workload of the tasks does not exceed the agent's capacity. Figure 5 also shows that the penalty on an agent switching tasks has the desired effect of keeping the agents designated to tasks. There are no examples of a task being switched between agents as the task is being executed.

AFAR Pilot Study

To test the feasibility of employing DFAS concepts, we performed a pilot investigation using our DFW ground control simulator. We ran two taxi scenarios requiring the participant (an experienced air traffic controller) to manage arriving and departing traffic safely and efficiently. The two scenarios involved a moderate traffic load and both scenarios included a few off-nominal events to make the scenarios more interesting and challenging for the participant. Several photographs of the AFAR setup are shown in Figure 6.

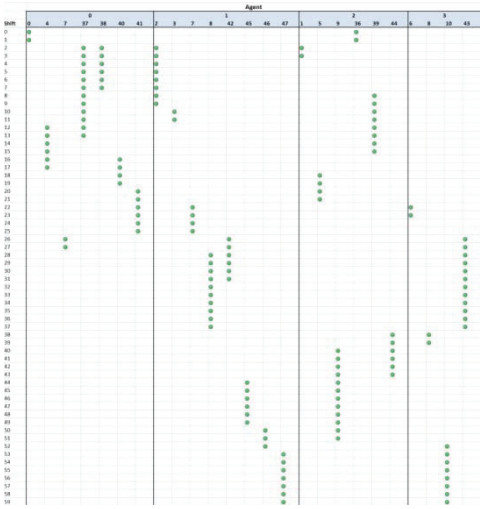

Figure 5: An example agent task assignment schedule.

The ground traffic control participant completed the two scenarios with and without the DFAS component managing the task allocation. In the non-DFAS condition, automation still assisted the ground controller; however, the tasks were assigned according to a static round-robin heuristic. The automation during the non-DFAS operation was preserved to prevent outcome differences from being confounded by an automation factor, but instead attempted to keep automation consistent with and without DFAS in order to engender a more controlled evaluation of the DFAS technology. In both conditions, an automated agent could be assigned tasks of receiving the aircraft (acknowledge), providing taxi instructions, and granting permission to taxi.

Figure 6: (Top) The experimental setup with the participant in the center and the pseudopilots on the left and right. (Left) The participant's desk. (Right) Participant using the system.

Each of the participants received a training brief and two training trials to familiarize himself with the automation and user interface. The two experimental scenarios were run, one with the DFAS algorithms and one with the static function allocation (round robin). There were 2 settings for the static allocation which manipulated the number of aircraft under automated control. Each participant performed two experimental scenarios on two different days (one DFAS and one static each day), for a total of eight experimental trials. The trial order pertaining to the inclusion of DFAS was balanced within and between participants.

Subjective ratings of mental workload were captured via the NASA Task Load Index (TLX), developed by NASA Ames Research Center (Hart & Staveland, 1988). The NASA TLX is a multi-dimensional rating procedure that provides an overall workload score based on a weighted average of ratings on six subscales: Mental Demand, Physical Demand, Temporal Demand, Own Performance, Effort, and Frustration. The degree to which each of the six factors contributed to the workload associated with each automation condition, from the participants' perspectives, was determined by their responses to pair-wise comparisons among the six factors. Magnitude ratings on each subscale were then obtained after each scenario. Ratings of factors deemed most important in creating the workload of a task were given more weight in computing the overall workload score, thereby enhancing the sensitivity of the scale. Table 1 provides the descriptive statistics for the NASA TLX data for each sub-scale, as well as for the composite workload score.

Table 1: NASA TLX Subjective Workload Rating Descriptive Statistics

	Mean		Standard Deviation	
Scale	Static	DFAS	Static	DFAS
Mental	48.75	43.75	12.37	26.52
Physical	27.50	12.50	17.68	0.00
Temporal	55.00	43.75	28.28	30.05
Performance	80.00	91.25	7.07	8.84
Effort	57.50	36.25	7.07	30.05
Frustration	25.00	6.25	28.28	1.77
Composite	49.00	55.00	27.58	12.02

While there were no statistically significant differences in workload ratings between the static and DFAS conditions (all $p > 1.0$), it is important to note the preliminary nature of this finding given the small dataset collected , as well as the variability in responses within the dataset. We therefore highlight a few trends that may bear significance in future studies:

- The dynamic function allocation condition showed a trend toward lower workload ratings across the "physical demand," "temporal demand," "effort," and "frustration" sub-scales.
- Variability was high in ratings of "mental demand," "temporal demand," and "effort. These sub-scales were considered by the controllers (via the subscale ranking/paired comparison procedure) to be major contributors to overall perceptions of workload.
- "Physical demand" and "frustration" were considered to contribute the least to overall perceptions of workload.

Although we lack the necessary data to draw statistically definitive conclusions, the results of the pilot study were encouraging. Perhaps most telling are the words of the participants themselves. When asked to qualitatively compare the runs with and without DFAS, the experienced ground controllers responded with the following comments:

> *Participant 1: "*Without a doubt, my temporal demand and mental workload increased substantially in the static [function allocation] environment. When working in DFAS mode, a feeling of great confidence came over me, knowing that DFAS would kick in and make my job manageable while traffic increased."
>
> *Participant 2: "*[The DFAS function allocation] was easier because it acted as more of a manager."

Conclusions and Future Work

In this paper we have introduced a testbed that is designed to evaluate adjustable autonomy technology early in the design cycle. A use case involving ground control at DFW airport was developed to exercise the testbed and a pilot study was conducted involving two experienced air traffic controllers. Although still preliminary, the results of the pilot study were encouraging in that the participants reported sufficient trust and confidence as well as DFAS's ability to mitigate workload. The testbed has shown to be easily configurable by running several operational scenarios utilizing two distinct function allocation algorithms: the round-robin heuristic and the DTS task allocation algorithm (DFAS). In addition, the fidelity and user interface sufficiently captured the airport surface domain to create realistic interactions and workload for the participants.

Results to date indicate that the flexibility and utility of the AFAR testbed will enable future investigation into human-system interaction and approaches to adjustable autonomy. As such, the next steps in this project involve investigating other DFAS algorithms and including multiple participants with expanded roles and in more complex operational scenarios. In addition, the testbed can be expanded to involve the air traffic control domain (in addition to ground control) including coordinating within airport multiplexes. AFAR can also be expanded to domains outside of air traffic management into myriad

areas involving teams of humans and automated systems. Finally, we see the need to develop generate and deploy more comprehensive performance metrics to evaluate the allocation strategies. While traditional metrics (e.g., accuracy, completion time) can be used for assessing total system performance, these methods have generally been extended from human performance measurement approaches and often lack sensitivity for assessing the complex performance of the collaborative human/ automation team.

Acknowledgments

The authors would like to thank the NASA Airspace Systems Program for its support of this work under Contract #NNA08BC68C.

References

De Visser, E., Parasuraman, R., Freedy, A., Freedy, E., & Weltman, G. (2006). A comprehensivc methodology for assessing human-robot team performance for use in training and simulation. *In Proceedings of the 50th Annual Meeting of the Human Factors and Ergonomics Society.* Santa Monica, CA.

Degani, A., & Heymann, M. (2002). Formal verification of human-automation interaction. *Human Factors, 44*, 28-43.

Dixon, S., & Wickens, C. D. (2006). Automation reliability in unmanned aerial vehicle control: A reliance-compliance model of automation dependence in high workload. *Human Factors , 48* (3), 474-486.

Dolgov, D., & Durfee, E. (2006). Resource allocation among agents with MDP-induced preferences. *Journal of Artificial Intelligence Research, 27*, 505-549.

Floudas, C. & Lin, X. (2004). Continuous-time versus discrete-time approaches for scheduling of chemical processes: A review. *Computers and Chemical Engineering*, *11* (28), 2109-2129.

Freedy, A., Sert, O., McDonough, J., Weltman, G., & Tambe, M. (2008). Multi-agent adjustable autonomy framework (MAAF) for multi-robot, multi-human teams. *In Proceedings of the Symposium on Collaborative Technologies and Systems.* Irvine, CA: CTS.

Garey, M. R., & Johnson, D. R. (1979). *Computers and intractability: A guide to the theory of NP- completeness.* New York: W.H. Freeman.

Hart, S. G., & Staveland, L. E. (1988). Development of a multi-dimensional workload rating scale: Results of empirical and theoretical research. In P. A. Hancock, & N. Meshkati (Eds.), *Human Mental Workload.* Amsterdam, The Netherlands: Elsevier.

JPDO. (2007). *Concept of Operations for the Next Generation Air Transportation System.* Retrieved from http://www.jpdo.gov/library/NextGen v2.0.pdf

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

Parasuraman, R., & Wickens, C. D. (2008). Humans: Still vital after all these years of automation. *Human Factors, 50* (3), 511-520.

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

Pinedo, M. (1992). Scheduling. In G. Salvendy (Ed.), *Handbook of Industrial Engineering* (2nd Edition ed.). Chichester: Wiley Interscience.

Reitsema, J., Chun, W., Fong, T., & Stiles, R. (2005). Team-centered virtual interactive presence for adjustable autonomy. *American Institute of Aeronautics and Astronautics (AIAA) Space* .

Scerri, P., Pynadath, D., & Tambe, M. (2002). Towards adjustable autonomy for the real world. *Journal of Artificial Intelligence Research*, 17, 171-228.

Schurr, N., Patil, P., & Pighin, F. (2006). Resolving Inconsistencies in Adjustable Autonomy in Continuous Time (RIAACT): A robust approach to adjustable autonomy for Human-Multiagent teams. *Fifth International Joint Conference on Autonomous Agents and Multiagent Systems.*

Sellner, B., Heger, F., Hiatt, L., Simmons, R., & Singh, S. (2006). Coordinated multi-agent teams and sliding autonomy for large-scale assembly. *Proceedings of the IEEE - Special Issue on Multi-Robot Systems , 94* (7), 1425-1444.

Sheridan, T., & Parasuraman, R., (2000). Human versus automation in responding to failures: An expected-value analysis. *Human Factors , 42*, 403-407.

Sierhuis, M., Bradshaw, J., Acquisti, A., Hoof, R., Jeers, R., & Uszok, A. (2003). Human-agent teamwork and adjustable autonomy in practice. *In Proceedings of the Seventh International Symposium on AI, Robotics and Automation in Space.* Nara, Japan.

Varakantham, P., Maheswaran, R., & Tambe, M. (2005). Exploiting belief bounds: Practical POMDPs for personal assistant agents. *In Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems (AAMAS).*