A Human-System Interface with Contingency Planning for Collaborative Operations of Unmanned Aerial Vehicles

Joseph B. Mueller[†], Christopher A. Miller[‡], Ugur Kuter[§], Jeffrey Rye[§], Josh Hamell[§]
SIFT, 319 1st Ave N, Suite 400, Minneapolis, MN 55316

The U.S. military is researching capabilities for collaborative and highly autonomous unmanned aircraft systems to conduct missions in denied or contested airspace, where multiple teams of aircraft would work together under the supervision of a single operator. One of the key challenges in this paradigm is the need for an effective human-system interface (HSI): one that could provide sufficiently detailed command and control authority and monitoring of plan execution for the entire system without overwhelming the operator. Another interesting challenge is the need to proactively find and approve multiple contingency plans, effectively granting the highly autonomous agents more freedom to operate when communication to the human supervisor is denied. This paper describes our recent efforts to design and develop a combined human-system interface and contingency planning tool that addresses these challenges. We present the design of our SuperC3DE system, provide examples of how certain interface components and contingency planning methods would be used in a notional scenario, and discuss key takeaways from formative evaluations by subject matter experts.

Nomenclature

HSI	=	Human-System Interface
SCOPE	=	Supervisory Contingency Plan Evaluator
CPA	=	Counter Plan Analysis
HPR	=	Hierarchical Plan Repair
CSI	=	Contingency Status Inferencing
FIT	=	Fusion/IMPACT Testbed
TAP	=	Team Autonomy Planner
UAV	=	Unmanned Aerial Vehicle
XINT	=	Mobile Interdiction
IPOE	=	Intelligence Preparation of the Operational Environment
AMASE	=	AVTAS Multi-Agent Simulation Environment
AVTAS	=	Aerospace Vehicles Technology Assessment and Simulation

I. Introduction

In 2014, the Defense Advanced Research Projects Agency (DARPA) launched the Collaborative Operations in Denied Environment (CODE) program. CODE's main objective is to develop and demonstrate the use of collaborative autonomy for systems of unmanned vehicles. Various CODE participants from industry are providing the technology components necessary for vehicles to perform autonomous planning and control, and to leverage each other's resources opportunistically during mission execution. The expectation is that this collaborative autonomy will enable teams of unmanned vehicles to overcome the challenges of denied communications and more capable adversarial systems.

In support of this vision, Smart Information Flow Technologies (SIFT, LLC) has designed and prototyped Supervisory Control for Collaboration and Contingency Delegation (SuperC3DE). SuperC3DE provides an integrated and holistic suite of human-computer interaction and intelligent decision-support tools. The system is designed to enable a single human operator to effectively command, monitor, and supervise a system of many

[†] Senior Researcher, SIFT, AIAA Senior Member
[‡] Chief Scientist, SIFT
[§] Senior Researcher, SIFT

unmanned aerial vehicles (UAVs) at once. The CODE operational environment also poses a unique set of challenges from a human-system interaction perspective: 1) maintain situational awareness (SA) of a complex, multi-actor, largely autonomous force with much less than full attentional capacity under conditions of high workload; 2) interact with and maintain supervisory control over highly autonomous vehicles even when they are out of communications; 3) make informed use of highly autonomous capabilities without surrendering all human oversight and control; and 4) achieve this level of SA with potentially reduced screen size and resolution, and in potentially harsh operational environments (vibration, noise, operators wearing gloves, etc.).

Our approach to addressing the above challenges builds on more than a decade of research into a human-automation interaction approach we call Playbook®[1]. Plays are not scripted behaviors, but rather constrained templates for action that offer a constrained range of actions for subordinates. Thus, playbooks are a convenient, human-familiar, and efficient way of delegating or transferring bounded authority, responsibility, and resources to a subordinate by codifying a supervisor's intent. In Playbook implementations, this play-based delegation has improved overall performance in unpredictable contexts[2] and reduced human workload relative to both static and autonomously adaptive systems[3].

Our research has culminated in the adoption of a Playbook-like delegation approach by portions of the Supervisory Control and Cognition Branch of the 711th Human Performance Wing at the Air Force Research Laboratory (AFRL). Our approach was employed in a three-year exploratory development and demonstration effort known as FLEX-IT[4] and, currently, is being expanded by the multi-force Autonomy Research Project Initiative (ARPI) known as IMPACT[5]. Both efforts have constructed play-based delegation HSIs that are similar, in many respects, to those needed for CODE. We have participated in both efforts and drew on the results, prior designs and some existing software implementations as the starting point for our CODE HSI.

Building on the Playbook paradigm, SuperC3DE provides a comprehensive, multi-view HSI that integrates play-calling and play-status visualization directly into the map display. To complement the core HSI, SuperC3DE also provides a computational evaluation and advisory component, the Supervisory Contingency Plan Evaluator (SCOPE). SCOPE emulates the process of planning deliberation between a supervisor and subordinate, serving to enhance understanding, both for machine and human operator, of their shared plan. SCOPE uses counter-planning techniques to: 1) examine plans for contingency upset conditions or breakdowns, 2) identify plan "patches" or repairs that remove those breakdowns or reduce their likelihood, and 3) projects vehicle behavior (even in conditions of comms denial) from a complex suite of initial contingent plans. SCOPE also evaluates and compares alternate plans, providing a structured framework for the human operator to quickly examine tradeoffs.

This paper discusses some of the key design concepts of the SuperC3DE system. We first provide a brief overview the HSI and SCOPE designs, along with an overview of our testbed environment. We then walk through the lifecycle of using the HSI to call, execute, and monitor a play, focusing our attention on a select subset of HSI components. Next, we provide a summary of evaluation results from subject matter experts that reviewed our design. Finally, we conclude with a summary of lessons learned through these evaluation sessions and offer suggestions for future research efforts that may benefit other programs with similar HSI challenges.

II. Overview of the SuperC3DE Design

Figure 1 below illustrates our proposed architecture for both the CODE system overall and for the portions of it we emphasized in our Phase 1 effort (depicted in yellow or gold). The two novel components developed for our SuperC3DE effort include a Playbook-based HSI and SCOPE. The blocks in red and green represent other CODE software components that our system either interacts with directly or that must be modeled in some way to close the loop and drive the simulation. SuperC3DE components interface directly with the Team Autonomy Planner (TAP), which is the component responsible for collaboration and plan generation for the team of UAVs. Plans generated from the TAP are then executed by the autonomy and execution modules in the individual vehicles. To support demonstrations and evaluations of our designs, we needed a simulation environment to emulate vehicle and world telemetry data. This was provided by additional work performed at AFRL in the form of the Fusion/IMPACT Testbed (FIT)[5,6,7]. Fusion[7] is a framework for the development and experimentation of user-interfaces that enable various types of human interaction with autonomous, intelligent agents. Designed and developed by AFRL Wright Patterson AFB, Fusion uses an instrumented, play-oriented user interface (UI) and connects to an extensible Java-based simulation framework that provides a configurable simulation of multiple air and ground vehicles, as well as access to various automation services to govern the behavior of those vehicles. IMPACT[5,6] uses the

Fusion environment and adds a number of unique computational and display components. FIT builds upon some of the Playbook-related HSI elements that are part of IMPACT, including the Play Creator, Playble, and Play Status tiles. SIFT had participated in earlier development efforts for both Fusion and IMPACT and we were granted permission to use both in our development of the SuperC3DE design. Together, the capabilities of FIT enable development and closed-loop demonstration by providing both an extensible platform for implementing new HSI designs, as well as a configurable simulation environment for UAV flight dynamics and control software.

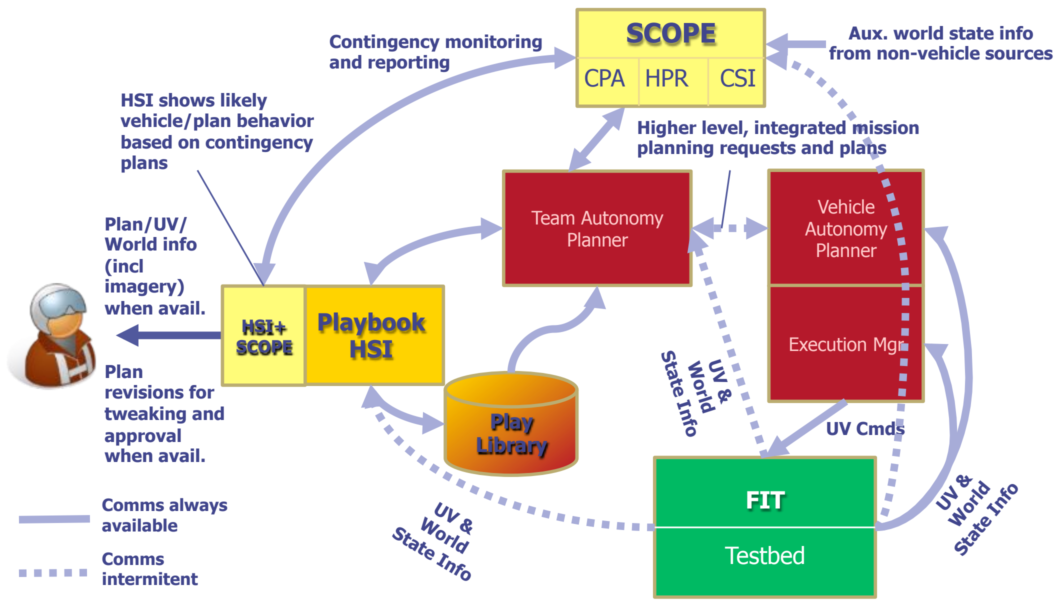

SCOPE
CPA
HPR
CSI
Contingency monitoring and reporting
Aux. world state info from non-vehicle sources
HSI shows likely vehicle/plan behavior based on contingency plans
Higher level, integrated mission planning requests and plans
Team Autonomy Planner
Vehicle Autonomy Planner
Execution Mgr
Plan/UV/ World info (incl imagery) when avail.
HSI+ SCOPE
Playbook HSI
Plan revisions for tweaking and approval when avail.
Play Library
UV & World State Info
UV Cmds
FIT
Testbed
Comms always available
Comms intermitent
UV & World State Info
UV & World State Info

Figure 1: High-Level SuperC3DE Architecture

A. HSI: Human-System Interface

Our primary goal with the HSI design was to provide a comprehensive, integrated approach to human interaction with and supervisory control over the CODE system. The focus was not to develop specific HSI components or widgets, but rather to arrive at an overarching design that integrates all CODE functionality for user awareness, control, and trust. We used a play-centered design to both command and maintain SA of multiple UAVs (up to 30+ in our designs). The final design was configured for a single touchscreen of variable size with mouse-based control and touch interactions, and allowed for voice input and output. The integrated HSI consisted of the following primary components:

1. Navigation Menu Bar/Button
2. **Map Display**[**]
3. Timeline Display[**]
4. Notification Pane
5. Imagery Review Pane
6. **Play Calling/Review Panes (Play Creator)**[**]
7. Play Quality Pane
8. **SCOPE Breakdown/ Repair Table**[**]
9. Active Play Pane (Playble)[**]

Figure 2 shows a preliminary layout of the above-listed HSI components. The layout evolved over the project as

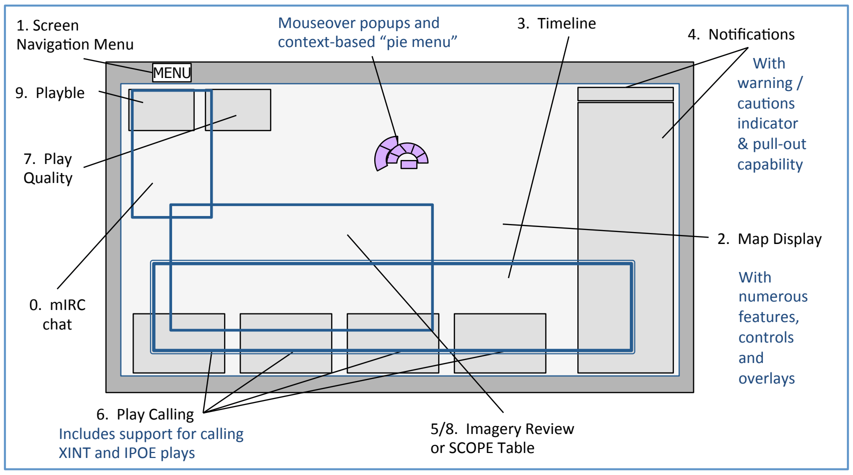

1. Screen Navigation Menu
Mouseover popups and context-based "pie menu"
3. Timeline
4. Notifications
With warning / cautions indicator & pull-out capability
MENU
9. Playble
7. Play Quality
0. mIRC chat
2. Map Display
With numerous features, controls and overlays
6. Play Calling
Includes support for calling XINT and IPOE plays
5/8. Imagery Review or SCOPE Table

Figure 2: Preliminary Layout of the Primary HSI Components

[**] These components evaluated as part of a Formative Review, summarized in Section V. Boldface components also presented in Section IV.

new components were developed, and as we gained a better understanding (through formative reviews and evaluations) of how the operator would prefer to use screen real estate during different modes of operation. The HSI is designed to support both pre-mission planning, as well as monitoring and supervision during mission execution.

In this paper, we present only a selected subset of the above-listed HSI components. Features of each component are presented in the context of how they might be used in a notional scenario (see Section III).

B. Supervisory Contingency Planning Executive (SCOPE)

SCOPE is an auxiliary advisory component which incorporates unique counter-planning and hierarchical plan repair capabilities in order to analyze plans, identify possible breakdown cases, and generate alternative courses of actions to avoid those breakdowns and preserve the original plans as much as possible. As illustrated in Figure 3, SCOPE consists of four primary components: Contingency Plan Analysis (CPA), Hierarchical Plan Repair (HPR), Contingency Status Inferencing (CSI), and Alternate Plan Evaluation (APE). CPA uses counter-planning techniques on plans produced by any automated planning system to identify how disturbances could cause failure. The HPR component performs additional planning and/or re-planning to identify the relaxation or substitution of constraints in the initial plan to make it robust to disturbances identified by CPA. Under comms denial, CSI determines what contingent portions of a plan are likely to be active at any point, in order to better understand what information a user needs to maintain awareness of that plan. The APE module applies a scoring method that takes a set of plans and a set of plan-evaluation dimensions as input, and ranks those input plans based on the score it computes across the input dimensions for the plans and their associated contingencies. Plan evaluation dimensions include metrics such as probability of success, number of assets required, fuel usage, duration, and possibly others.

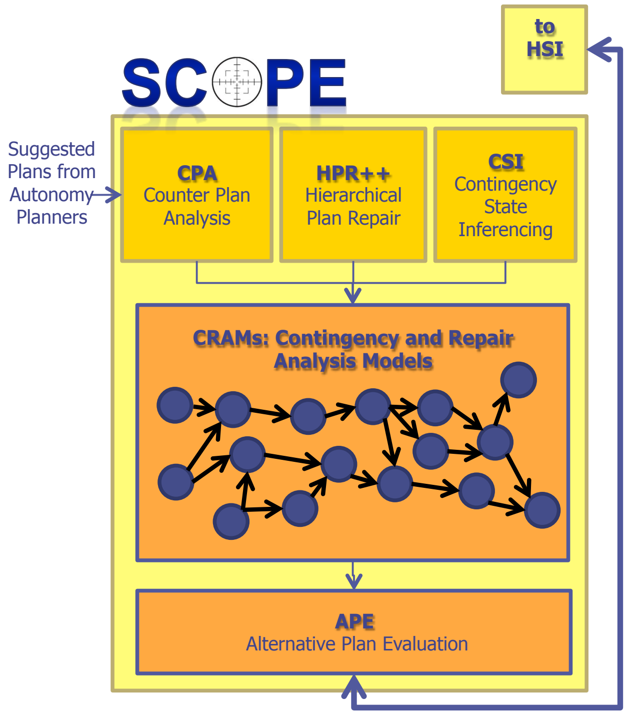

SCOPE
to HSI
Suggested Plans from Autonomy Planners
CPA Counter Plan Analysis
HPR++ Hierarchical Plan Repair
CSI Contingency State Inferencing
CRAMs: Contingency and Repair Analysis Models
APE Alternative Plan Evaluation

Figure 3: High-Level SCOPE Architecture

APE enables a quantitative comparison of plan alternatives, which might be generated by the human operators, CODE planners, SCOPE itself, or some combination, along several possible evaluation dimensions. The plans are ranked according to the aggregate score of the weighted dimensions, and the ranked plan alternatives are displayed for the human operator to visually explore and choose tradeoffs.

A fundamental element of SCOPE's analysis methods are CRAMs – Contingency and Repair Analysis Models. CRAMs are graphical models of plans that specify probabilistic, resource, and causal dependencies between actions. We have developed this formalism by extending the causal models used by the AFRL's Causal Analysis Tool (CAT)[12]. In SCOPE, CRAMs extend CAT's causal models to include resource modeling (e.g., fuel usage, duration, and others). CRAMs allow the causal information in the plans and in the contingencies to be used in a semantically consistent way to estimate other entries, which the plan representations or the human modelers choose not to provide.

FIT: Fusion/IMPACT Testbed

1. *Fusion and IMPACT*

Fusion is a framework for the development and experimentation of user-interfaces that enable various types of human interaction with autonomous, intelligent agents. Designed and developed by AFRL Wright Patterson AFB, Fusion uses an instrumented, play-oriented UI and connects to an extensible Java-based simulation framework that provides a configurable simulation of multiple air and ground vehicles, as well as access to various automation services to govern the behavior of those vehicles.

Fusion can connect to many different simulation environments or external services, using diverse communication protocols from STANAG 4586 to JSON and others. In addition, Fusion's highly modular and open architecture makes it possible to develop custom components for both the HSI and vehicle autonomy purposes.

The graphical layout of the baseline Fusion screen design is a map-based view that fills the screen with a notification bar along the top, a feedback bar along the bottom, and individual tiles that may be dynamically shown or hidden. These elements are shown in Figure 4. The notification bar is used to display messages and alerts, and can be expanded to pull up a Fusion control panel for managing the tile and canvas layout. The feedback bar is used for automation feedback, as well as for speech control and response. For example, a press-and-hold on the feedback bar triggers speech recognition to process voice commands. We used this feature to call plays via voice. Both the map-based canvas and floating tiles are used to implement specific UI controls. All of the HSI components that we developed in Fusion for Phase 1 were implemented either as map-based icons or tiles.

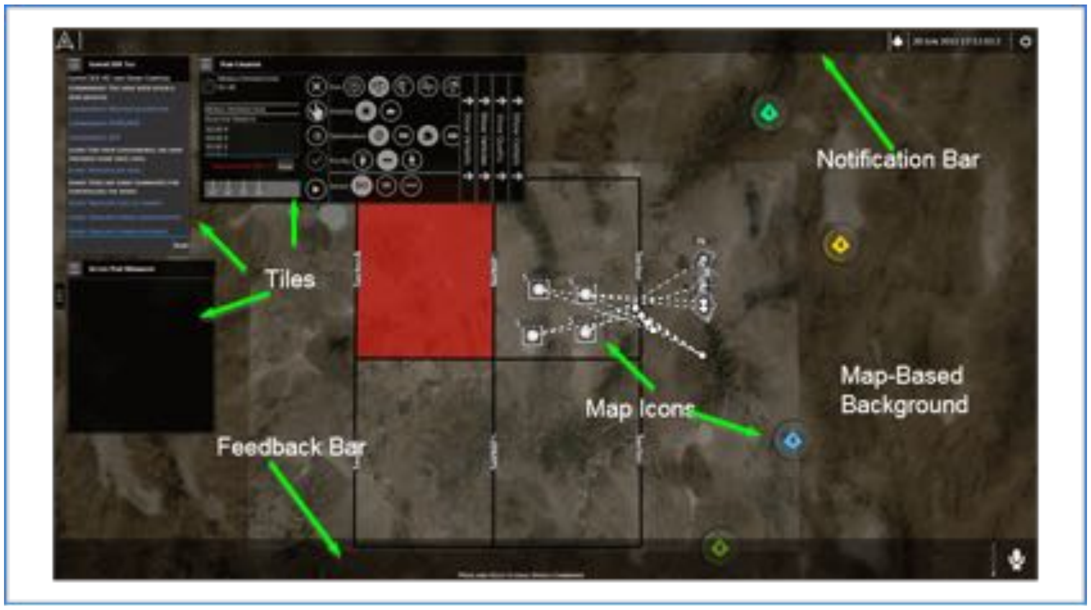


Figure 4. Example of the Fusion Layout

Since 2014, SIFT has been actively involved with the IMPACT program, a multi-agency coordinated ARPI. The focus of our work on IMPACT was to extend the functional capabilities of the HSI concepts we previously developed under our FLEX-IT project in active collaboration with AFRL, and which we implemented in Fusion. The collaborative work on IMPACT has brought together a number of unique autonomy-related components into the Fusion environment.

In designing the SuperC3DE HSI, we built upon some of the Playbook-related HSI elements that were part of IMPACT, including the Play Creator, Playble, and Play Status tiles. For each of these graphical UI widgets, we made extensive modifications for our own HSI designs and added extensions to support CODE concepts. The HSI designs are discussed in detail in Section 6.

2. *AMASE Simulation*

AMASE was developed by the Aerospace Vehicles Technology Assessment and Simulation (AVTAS) laboratory at AFRL Wright-Patterson[8]. The motivation for AFRL's development of AMASE was to provide a common and extensible simulation framework for researchers across several different organizations to use in their development and evaluation of control algorithms for teams of UAVs. The Java-based application is provided with a NetBeans IDE, including a library of tools and a network interface to connect to external applications. The simulation framework organizes vehicles and other entities as separate models, where each model is simulated with a set of distinct modules. The attributes of each vehicle and the initial state values for a simulation run are configured through a scenario EXtensible Markup Language (XML) file. AMASE provides a configurable kinematic flight module to simulate the basic physics of UAV flight, as well several controlling modules to provide a core set of commonly used functions, such as gimbal control, camera control, waypoint navigation and loitering.

Each of the vehicle modules responds to mission commands or vehicle action commands that are issued as messages from other modules in the system.

Data transfer in AMASE includes 1) data read from and written to XML files, 2) data exchanged between software modules within AMASE, and 3) data exchanged between AMASE and other applications over a network connection. For all of these cases, AMASE uses the Lightweight Message Construction Protocol (LMCP) to represent objects. The main purpose of LMCP is that it defines a serialization protocol for data types, allowing a common set of objects to be used between software components implemented in a number of different programming languages. The LMCP package making tool provided with AMASE automatically generates the methods to serialize and de-serialize all built-in and user-defined LMCP objects, which enables us to pass any LMCP-based object between AMASE and its client without incurring any additional work.

LMCP is a generic, domain-independent protocol, and provides the underlying structure for all of the data objects that are passed to/from the AMASE application or between its modules. Building on top of LMCP is the Common Mission Automation Services Interface (CMASI). CMASI is a specific type of Message Data Model (MDM) that uses LMCP. It was developed with the goal of providing a standard of communication between software tools that perform automated or semi-automated command and control of unmanned vehicles. As such, CMASI includes an assortment of specific data structures that define the commands, objectives, and constraints associated with UAV missions and tasking[9].

3. *Plan Representations*

The HSI design is built around a play-calling paradigm that assumes a hierarchical plan structure with branching logic. This type of branching plan representation is consistent with DARPA's open architecture vision for CODE and with the planning approaches being developed by the system integrators. In addition, SCOPE uses two specific artificial intelligence (AI) planning formalisms to internally represent CODE plans and user plays. It uses the Planning Domain Description Language (PDDL) to represent primitive program statements and hierarchical task networks (HTNs) to represent conditionals, loops, methods, and classes in Java programs.

The architecture diagram in Figure 4 illustrates which types of plan representations are used by different components of the SuperC3DE demonstration system. We stubbed the TAP by simply creating a set of our own handwritten plans, assigning a unique ID number to each plan so it could be referenced and loaded in the course of performing demonstrations. For the executive and vehicle simulation in AMASE, these static plans were expressed using a custom extension of the CMASI protocol. SCOPE, on the other hand, stored a separate instance of each plan as an HTN.

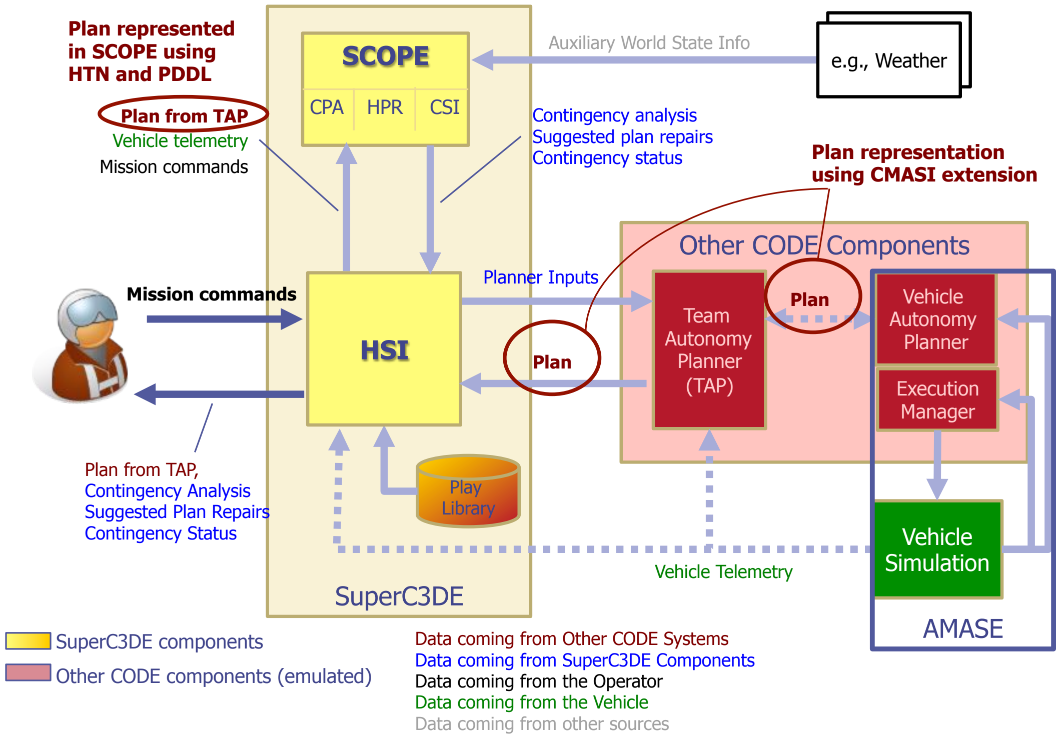


Figure 5: Plan Representations Used by SuperC3DE Components

4. SuperC3DE Extensions to AMASE

We developed several extensions to the core AMASE software to enable automatic execution of the branching plans that were formulated for the various Phase 1 demos. The diagram in Figure **6** illustrates a subset of the core AMASE modules that we interfaced with on the right side of the figure, along with the set of SuperC3DE extensions that we developed on the left. Note that this diagram shows the architecture for a single vehicle, which is applied uniformly to all UAVs in the simulation. The SuperC3DE Executive (hereafter SCExecutive) extends the AMASE EntityModule class, and thus it behaves just like another AMASE module. As the diagram shows, the SCExecutive manages all interactions with other native AMASE simulation modules.

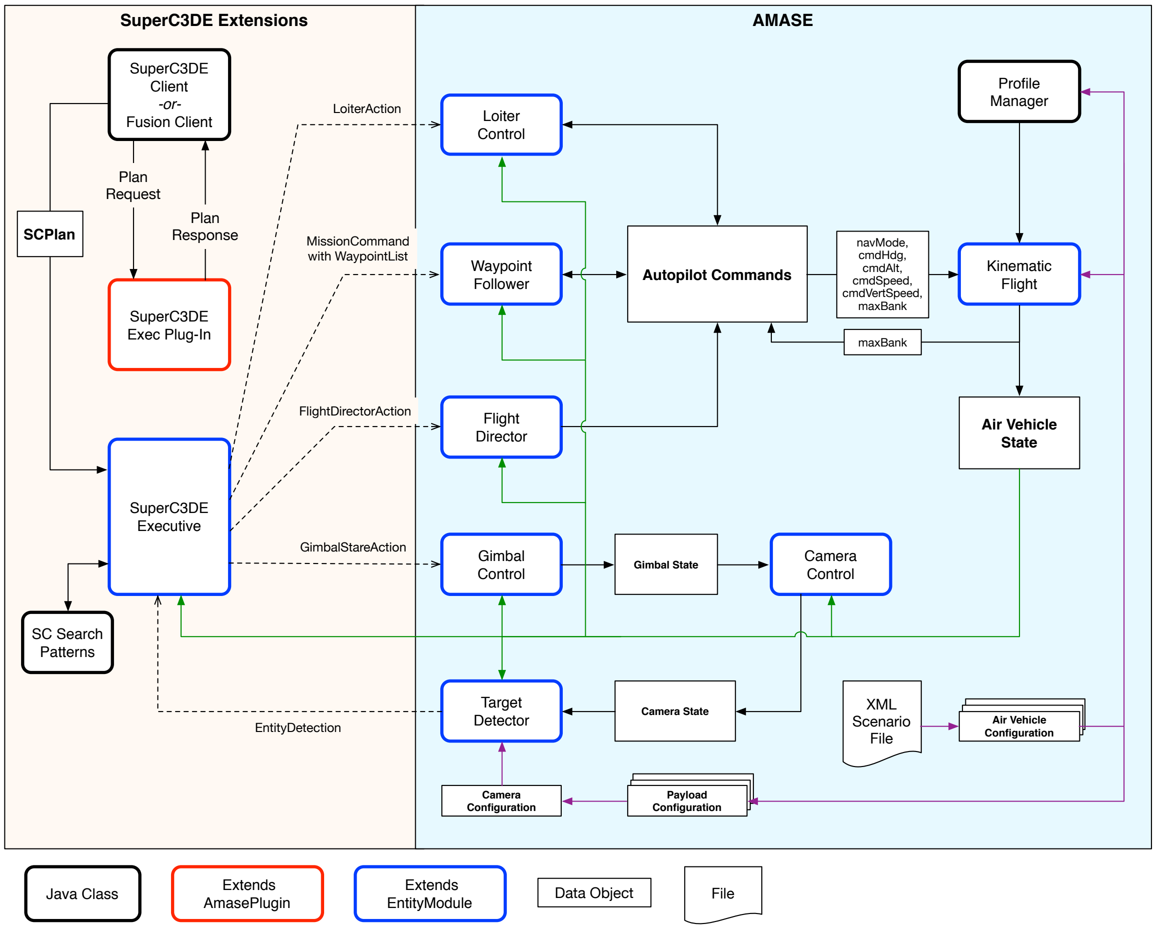


Figure 6. AMASE Simulation Architecture with SuperC3DE Extensions

The SuperC3DE Executive Plugin (hereafter SCExecPlugin) is used to programmatically generate plans for execution. It listens for a Plan Request message that may come either from the Fusion application, if it is connected, or from a SuperC3DE client (hereafter SCClient). The SCClient was developed to expedite development and testing of the simulation and SCExecutive. This allowed us to develop and test the SCExecutive software using only the AMASE simulation, rather than having to continually interface with the full HSI in Fusion during this process. As features in the SCExecutive reached completion, we performed fully integrated tests and demonstrations by 1) launching the HSI in Fusion, 2) issuing commands to the SCExecutive in AMASE, and 3) observing plan execution by viewing telemetry in the Fusion HSI.

When the SCExecPlugin receives a Plan Request message, which includes a plan ID, it programmatically generates the hierarchical branching plan associated with that ID, and then sends the SCPlan object back to the client in a Plan Response message. This plan generation service was used by the Fusion-based SuperC3DE HSI to emulate the type of interaction it would have with a TAP component. At this point, the client now has a SCPlan object that it can send to the SCExecutive for execution.

The SCExecutive class stores one SCPlan at a time, where each SCPlan data structure includes a linked-list of uniquely numbered SCActions, and where the first action to be executed has an ActionID of 0. Each SCAction in a SCPlan is extended from the generic SCAction type to be a specific type of action, such as *FlyWaypoints* or *SearchAreaForTarget*, for example. Every SCAction also has an array of ExitData structures to facilitate branching. Each ExitData block provides the conditions under which the current action should be terminated and identifies the next ActionID to switch to. The executive issues commands to execute the current action and continuously monitors the state of the simulation. When any of the exit conditions prescribed for the current action are satisfied, it switches to the corresponding ActionID for that exit condition.

As Figure **6** indicates, the SCExecutive module controls the UAV actions by sending four different types of commands:

- *LoiterAction*—is sent to the LoiterControl module. The loiter command specifies the latitude, longitude and altitude of a loiter center-point, the type of loiter pattern (circle, figure eight, or race-track), the dimensions of the pattern (radius and baseline if needed), and the airspeed.
- *MissionCommand*—with a waypoint list is sent to the WaypointFollower module. Waypoint following is used to execute several different actions, including ingress, egress, and search patterns.
- *FlightDirectorAction*—is sent to the FlightDirector module, which supersedes the autopilot and all other flight modules. This action specifies the heading and airspeed for the aircraft to fly. The SCExecutive uses a FlightDirectorAction to avoid restricted operating zones (ROZ) and no-fly zones (NFZ) if they are encountered.
- *GimbalStareAction*—is sent to the GimbalControl module. With this action, we specify a latitude/longitude coordinate on the ground for the camera to point at. This action is used to direct the camera during egress, ingress, search patterns, and for imaging targets.

The executive uses the SCSearchPatterns class as a utility for developing search patterns. It computes waypoint sequences for different types of search patterns, including lawn-mower, strip-and-grid, and spiral, and it performs geodetic calculations to convert between east-/north- and latitude-/longitude-based coordinates.

5. *Plan Execution*

The main purpose of the SCExecutive is to perform plan execution. By executing the plan in a representative multi-UAV simulation environment, we can exercise those aspects of the interface that require dynamic vehicle telemetry updates and, consequently, provide an interactive experience to the human operator.

The plan representations for all AMASE scenarios were encoded in MDMs that extend the CMASI message protocol. Specific details of the plan representations are discussed on page 6. In Phase 1, we developed a set of example plans for two main types of play calls: 1) mobile interdiction (XINT), and intelligence preparation of the operational environment (IPOE). The branching structure of these plays is depicted in Figure **7**. The boxes in each diagram represent specific types of configurable actions that the executive is responsible for managing. The diamond branch points represent exit conditions that the executive is responsible for monitoring, and that dictate the next action in the plan to be executed.

In the XINT plan, for example, the vehicles first fly to the area of interest (AOI), performing a scanning ingress to the expected Scud missile location, and then scan at that location. If a Scud is not found initially, the vehicle performs a spiral search (by default, though other alternate search patterns may be stipulated) until the Scud is found or until the prescribed time limit has elapsed. Once the Scud is found, the vehicle loiters, scans and reports from that location until the prescribed time limit has elapsed. Finally, the vehicle performs a scanning egress and returns to base.

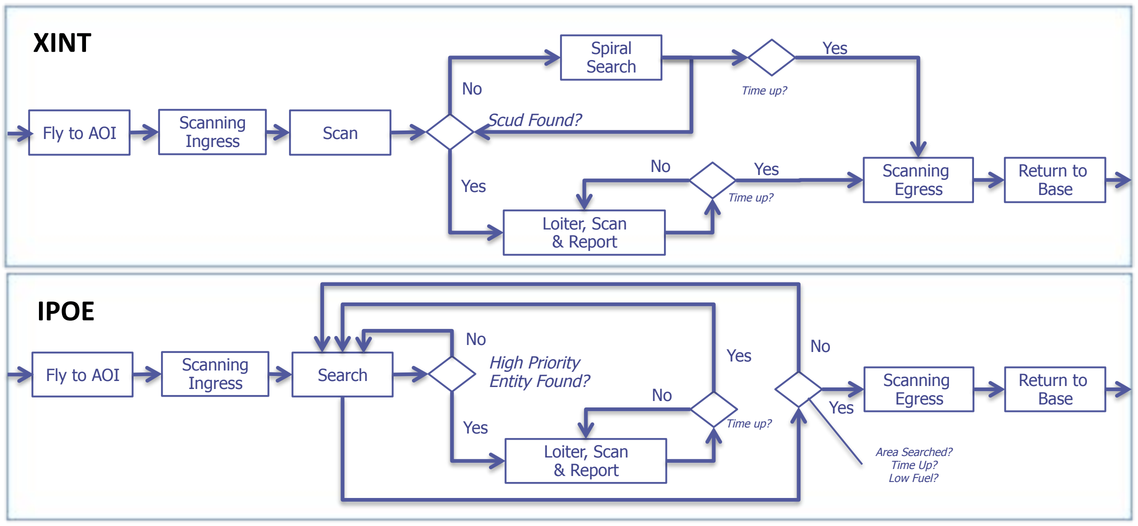


Figure 7. Branching Plan Structure for the XINT and IPOE Plays

The process by which the SCExecutive monitors and carries out the actions in a branching plan is described in Figure 8. Plan information is encoded as a SCPlan object, which itself contains a linked list of SCAction objects; both SCPlan and SCAction are a customized extensions of the CMASI message protocol. The executive, along with all other AMASE modules, updates at default frequency of 10 Hz, though this may be configured in the scenario XML file. At the beginning of the update function, the executive first checks whether a new SCPlan object has been received. If so, the executive sets the current action ID to 0, finds the corresponding SCAction within the SCPlan tree, and begins execution. A list of supported action types is shown in the diagram. Many of these actions represent extensions to CMASI. The executive will perform specific functions and issue different commands to other AMASE modules, depending on the current action type being executed.

The executive also monitors the air vehicle state at the end of each update step to determine whether any of the exit conditions for the current action have been met. If an exit condition is met, it updates the current action ID to the next action ID specified by that exit condition, causing the new action to be executed on the next update cycle.

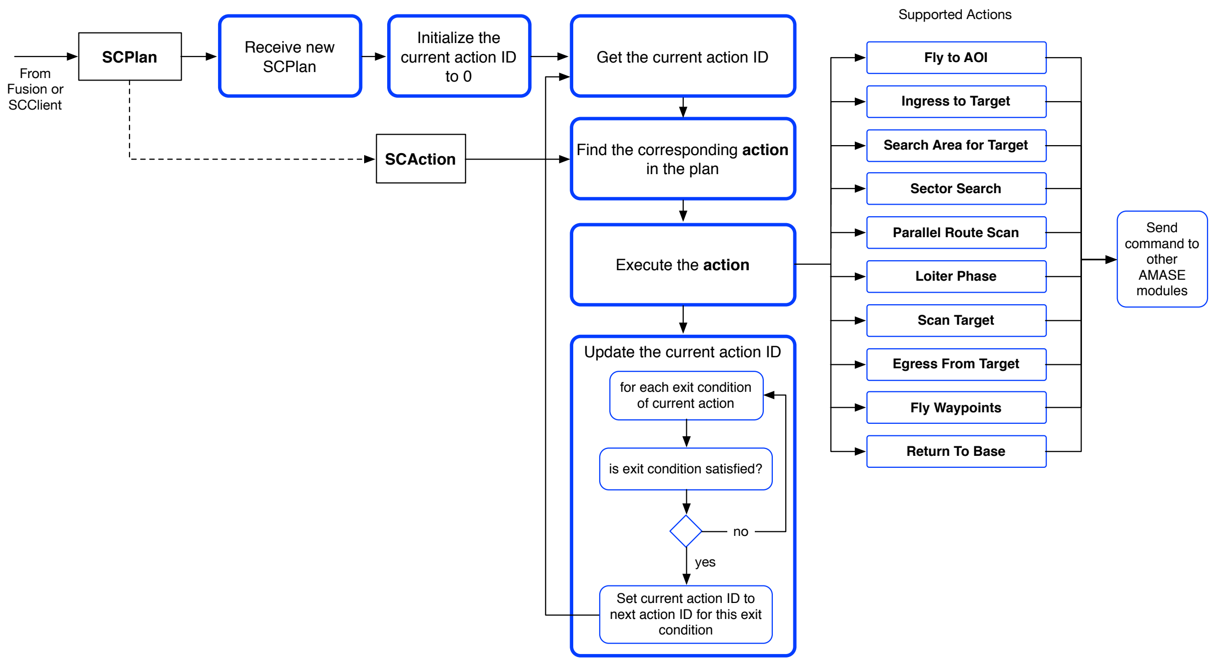

From Fusion or SCClient
SCPlan
Receive new SCPlan
Initialize the current action ID to 0
Get the current action ID
SCAction
Find the corresponding action in the plan
Execute the action
Update the current action ID
for each exit condition of current action
is exit condition satisfied?
no
yes
Set current action ID to next action ID for this exit condition
Supported Actions
Fly to AOI
Ingress to Target
Search Area for Target
Sector Search
Parallel Route Scan
Loiter Phase
Scan Target
Egress From Target
Fly Waypoints
Return To Base
Send command to other AMASE modules

Figure 8. SCExecutive Process to Monitor and Execute Branching Plans

III. Example Use Case

Here we describe a notional scenario to provide an example use case for SCOPE and the HSI. The scenario backdrop is an arbitrary geographic location (in this example, a mountainous area in southern Nevada). We also do not define or specify any particular UAV platform, though our models assume a medium-altitude, medium-endurance UAV with heterogeneous sensor packages. The scope of the scenario is illustrated in Figure 9. There are three areas to search, as well as a no-fly zone that must be avoided. Within the search areas, we have expected locations for both short-range and long-range surface-to-air missile (SAM) sites. There also exists another pop-up SAM site (not initially known). To the east of the search areas, we have three sorties, each composed of four UAVs. The mission has two parallel objectives: 1) XINT, and 2) IPOE. The goal of IPOE is to search the area and build intelligence by identifying and cataloging all items of interest, while the goal of the XINT is to search, identify and report Scud missiles specifically.

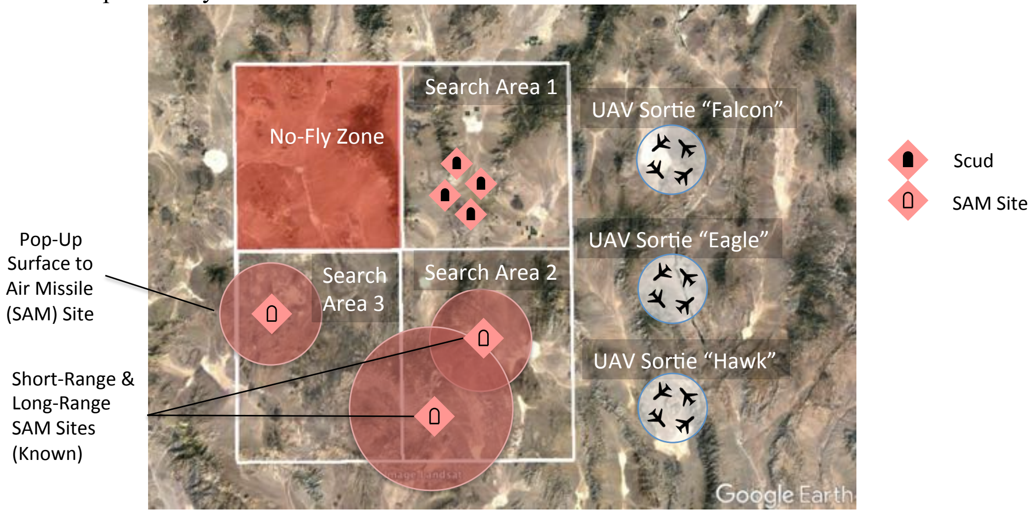


Figure 9: Scenario for XINT and IPOE with 3 UAV Sorties

We used this notional scenario as a means of demonstrating how the SuperC3DE HSI can be used to accomplish two primary tasks: 1) develop and visualize one or more plans for executing the mission, and 2) monitor the status of the mission during execution. Using our Playbook approach, the user first calls the play by selecting XINT and IPOE as objectives, and then continues to specify more detailed priorities and constraints for the mission. With the play fully defined, the user sends the play call to the planning software, which computes one or more plans to execute the play. In order to demonstrate interaction with the HSI, we stubbed the planning software and used SHOP2 to develop a representative set of hierarchical plans that carry out the dual XINT/IPOE mission. We then use the AMASE simulation with an extended set of onboard control software to execute the plan. Simulated telemetry from AMASE is provided to the Fusion application, enabling us to dynamically update the map display and provide user interaction during plan execution. In the next section, we focus on a few specific instances of this overall process, and discuss the corresponding components of the HSI.

IV. Play-Calling and Play Execution Monitoring

In this section, we provide an overview of selected HSI components used in the process of first calling a play, and then later monitoring the play's execution. The notional scenario described in Section III is used to motivate the examples.

A. Play-Calling Interfaces

The Play Calling component of SuperC3DE is used for creating new Plays and reviewing or editing previously created Plays. In order to create a new Play, the user either uses the navigation bar or may tap on the microphone icon on the bottom right of the Map Display to begin speaking. The Play Creator is always available via the navigation bar or by using voice input. Figure 10 shows an example of the user tapping the microphone icon and speaking "IPOE on quadrants B, C, and D." The left side of the picture shows a new Play has been created with the Sector Search selector set to IPOE. Tooltips appear when the user hovers over each icon on the Play Creator as a reminder of each option's functionality. There is a compositional formalism, or visual semantics, to these play icons, developed originally at AFRL, which provides a reminder of the available plays. For example, all target surveillance plays have a basic circle with centered "+" sign, all route surveillance plays have a line through the center of the icon, Sector Search play icons are built around a square box, etc.

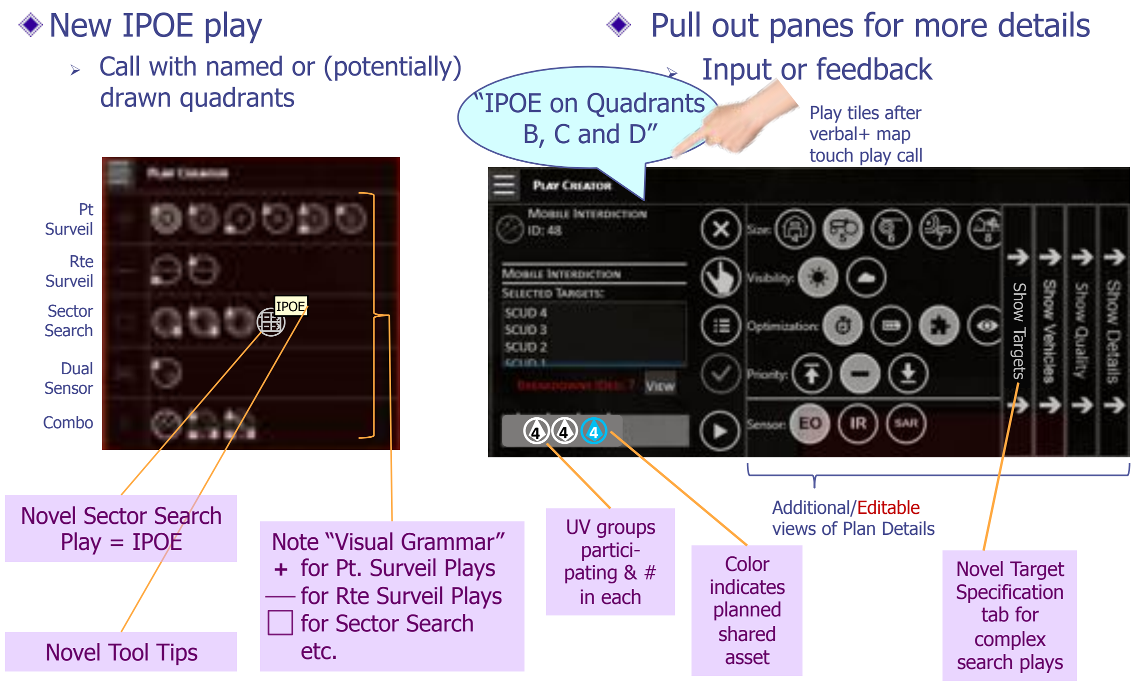

New IPOE play
Call with named or (potentially) drawn quadrants
Pull out panes for more details
Input or feedback
"IPOE on Quadrants B, C and D"
Play tiles after verbal+ map touch play call
Pt Surveil
Rte Surveil
Sector Search
Dual Sensor
Combo
IPOE
Play Creator
Mobile Interdiction ID: 48
Mobile Interdiction
Selected Targets:
SCUD 4
SCUD 3
SCUD 2
View
Size:
Visibility:
Optimization:
Priority:
Sensor:
EO
IR
SAR
Show Targets
Show Vehicles
Show Quality
Show Details
Additional/Editable views of Plan Details
Novel Sector Search Play = IPOE
Novel Tool Tips
Note "Visual Grammar" + for Pt. Surveil Plays — for Rte Surveil Plays ☐ for Sector Search etc.
UV groups partici-pating & # in each
Color indicates planned shared asset
Novel Target Specification tab for complex search plays

Figure 10. Voice Input to Play Calling Component

More detail on our approach, philosophy and prior work in creating play calling interactions may be found in various resources including[1,4,10]. Plays are intended not as pre-specified missions, nor as general control modes, but rather as partially instantiated templates of behavior. Just as in a sport such as American football, or more continuously, soccer or basketball, a play has a goal and a constrained set of methods that constitute instances of that play. Other means of accomplishing the goal may exist, but they would not be instances of the play. The methods are also not exhaustively planned; they are shaped and constrained, but some authority is left to the agents/players at execution time to implement the plan as best they see fit. Plays are also hierarchically defined and can be composed and decomposed (as illustrated in Figure 10 above). Plays can be composed of smaller elements to create new plays or longer missions. While much of this should not be done during mission execution if time and workload constraints prohibit it, it does provide a bridge between mission planning and dynamic aspects of execution. We envisioned the possibility of planning multiple plays to meet anticipated needs or contingencies during a mission with the ability to call them and even to briefly edit them (in the form of modifying pre-set and/or default parameters) either immediately before launching a UAV salvo or dynamically (assuming comms availability) during their mission performance.

The view for reviewing or editing previously created Plays is shown on the right of Figure 10 above. On the bottom left of the review and editing pane of the Play Calling interface is a list of UAV groups and number of groups participating in the selected Play. The section to the right of that is the main review and edit interface. This section allows the user to see and set each of the Play's details. Our general philosophy in play calling is to give the user the *capability* to set a wide range of parameters to specialize the play to the current need and intent, but to *require* as few as possible of these parameters to be set. Thus, all parameters come with intelligent default settings, customized to the current mission and expected needs. Furthermore, parameters can adapt as the user stipulates one or more of them (for example, dictating expected cloudy weather may influence either standoff distance or type of sensor used). Many of these parameter settings for an XINT play are described below.

The Size section is indicated on the top by several icons depicting relative sizes of the object to be targeted based on the National Image Interpretability Rating Scales[11]; a barn is the biggest, and a crumb being eaten by a rodent is the smallest. Under that is the Visibility section. This section indicated the weather condition (clear or cloudy in this example). Below the Visibility section is the Optimizations section. This section lists how the play should be optimized. Time is one of the selected optimizations in this example; other available optimizations include fuel/power, area coverage and image quality. The Priority section sets the priority of the play, which can be above normal, normal, and below normal. Under that lies the Sensor section. This section allows the user to select which sensors the UAV will be using for the play. In this example, the electro-optical (EO) sensor is selected.

Figure 11 and Figure 12 show the contents of each vertical tab on the right of the Play Calling interface. The Subplay Variants tab shows the user the defaults for the play, but it is also editable. This tab allows the user to select different search types, search patterns, its report priority, and what each UAV should do after the play is completed. There are two cases for what the UAV should do after the play is completed. Next+ indicates what the UAV should do after a successful play (in this case, finding the desired target), and Next- indicates what the UAV should do after an unsuccessful play.

The tab to the right of Subplay Variants is the Vehicle Assignments tab, which illustrates (and provides control over, if desired) the vehicles assigned to the play. The next tab shows the Play Quality Glyph, a pie chart developed at AFRL that indicates the play's current expected or actual quality against several play-specific parameters. We explained this chart in more depth in the subsequent section.

The last tab is the Details Pane. The Details Pane allows the user to view and edit any other miscellaneous information in more detail. This list is scrollable, so the figure illustrates only the top of a potentially long list of available details which can be edited. While this is not a particularly efficient means of viewing or editing such details, a successful design should make details at this level of granularity almost never needed during mission execution. It is our assumption that it is better than not to have them available for those rare cases where they are needed.

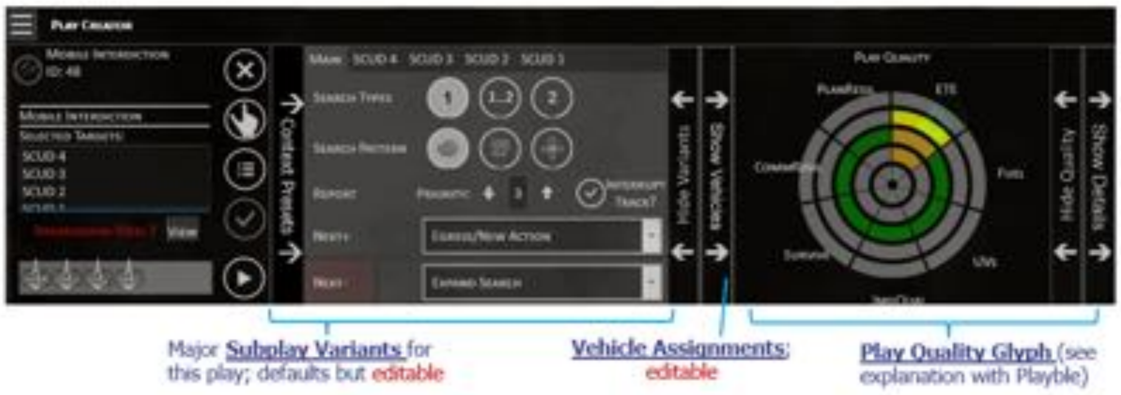


Figure 11. Play Calling Interface: Sub-Variants, Vehicle Assignments, and Play Quality Glyph

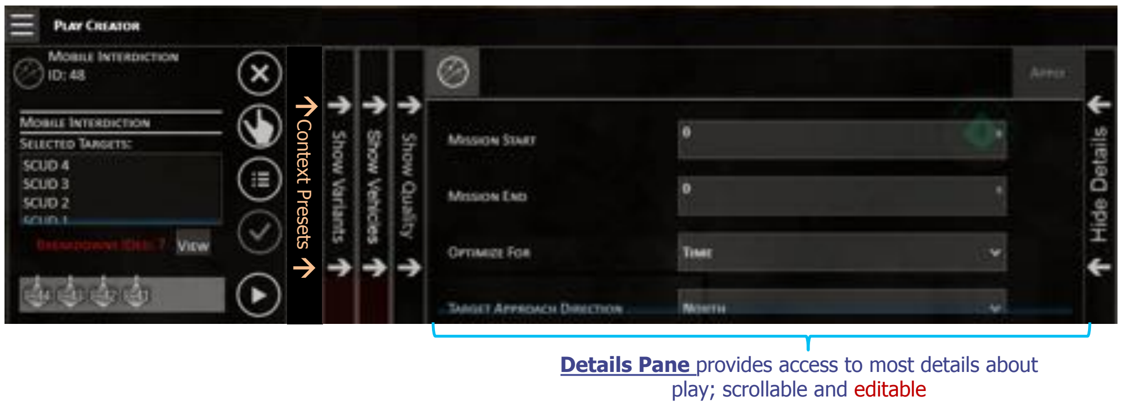


Figure 12. Play Calling Interface: Details Pane

Note that all of the above examples were depicted for an XINT play. The set of options available on the Play Calling Pane will be different depending on the play to be called, though they will adhere to the same general layout. For example, Figure 13 illustrates the design for a new icon for an IPOE play we added as a type of multi-sector search (illustrated using the graphic convention of multiple rectangles).

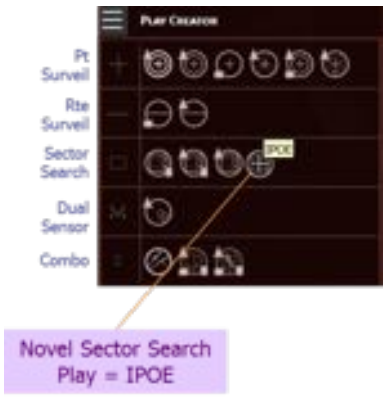


Figure 13. IPOE Play Icon Design

The IPOE play's expanded, additional tabs are similar to those for XINT illustrated above, but include a substitution for XINT's Context Variants tab in the form of a novel Targets Control. This was because the XINT play involved searching for specific targets in an area, and thus, specifying elements of search patterns, etc. for those targets was an appropriate level of detail given the assumed context of the XINT play. The IPOE play, however, involves searching a larger area for, essentially, anything of interest. Hence, in the context of this play, an appropriate level of intent specification detail involves specifying the types of targets of interest, their relative priorities and what to do about them if found.

B. Contingency Analysis and Plan Evaluation

Once the user has specified the priorities and constraints for the play within the Play Creator tabs, she can send the play-call to the planning software by pressing the Play icon, ▶. The planning software then attempts to generate one or more plans for the group of vehicles to execute the play, subject to those detailed specifications. Once a plan exists for executing the play, we use the counter-planning capabilities of our SCOPE software to attempt to identify *breakdown cases* – conditions in the environment, or perhaps actions taken by other actors in the environment, that have the potential to break our plan. SCOPE then attempts to generate repairs for the breakdown cases by creating new branches within the original plan. Note that this approach is valid whether SCOPE resides only at the ground control station, or if it is implemented onboard the vehicles as part of the autonomous planning software. In general, because contingency planning can be computationally intensive, and because it has the potential to generate a large number of branching repairs, it seems more practical to perform this analysis at the ground control station where there ought to be more computational resources, and where a human operator can judiciously prune the space of alternative plans.

Because each breakdown case is identified as potentially (probabilistically) causing the failure of the plan, we initially chose to present these as a series of events or outcome effects, each occupying one row of a table. Events could be sorted or filtered by probability under the user's control as one means of managing a potential flood of SCOPE outputs. Both event name and the Trigger/Cause column entry were derivable from the knowledge base that SCOPE uses to perform counter-planning. SCOPE's Hierarchical Plan Repair component generates minimal repair plans to (again probabilistically) provide a means of avoiding plan breakdowns stemming from the initially detected event. From these repairs, we derived both a repair suggestion name and an expected value (in terms of increased probability of success) of the repair. Other concepts were also explored for showing, in more detail, the tradeoff space of various elements, since complex planning for multiple UAVs in unpredictable environments frequently becomes a process of trading one kind of risk for another, or of trading risk for plan outcome quality. While many approaches are possible, we were of the opinion that they were generally far too complex for use during mission execution in the CODE context – and probably too complex for even immediate pre-mission planning.

The SCOPE Table display described above was reviewed by subject matter experts (SMEs) at a formative review (see Section V below). Feedback from the SMEs confirmed that it was both too complex and too abstract for operators to be completely comfortable. They were particularly critical of a portion of the table display in which we presented an expected value computation of each potential plan repair which they neither understood nor trusted. Their primary request was that we replace them with concrete dimensions that could project actual usage or risk dimensions for operator consideration and risk awareness.

Using these critiques and requests, we developed the Alternate Plan Evaluator (APE) as a more compact way of visualizing and comparing plan quality. APE projects the effects of a plan along dimensions that are previously defined as relevant for the play (and that can be tuned for different plays, either a priori or during analysis). A basic view of the APE Widget is shown in Figure 14 below.

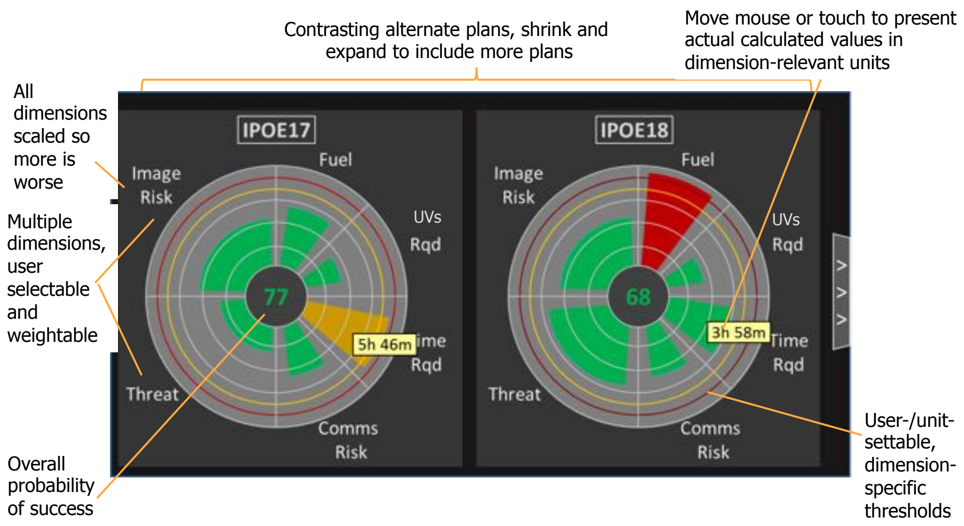
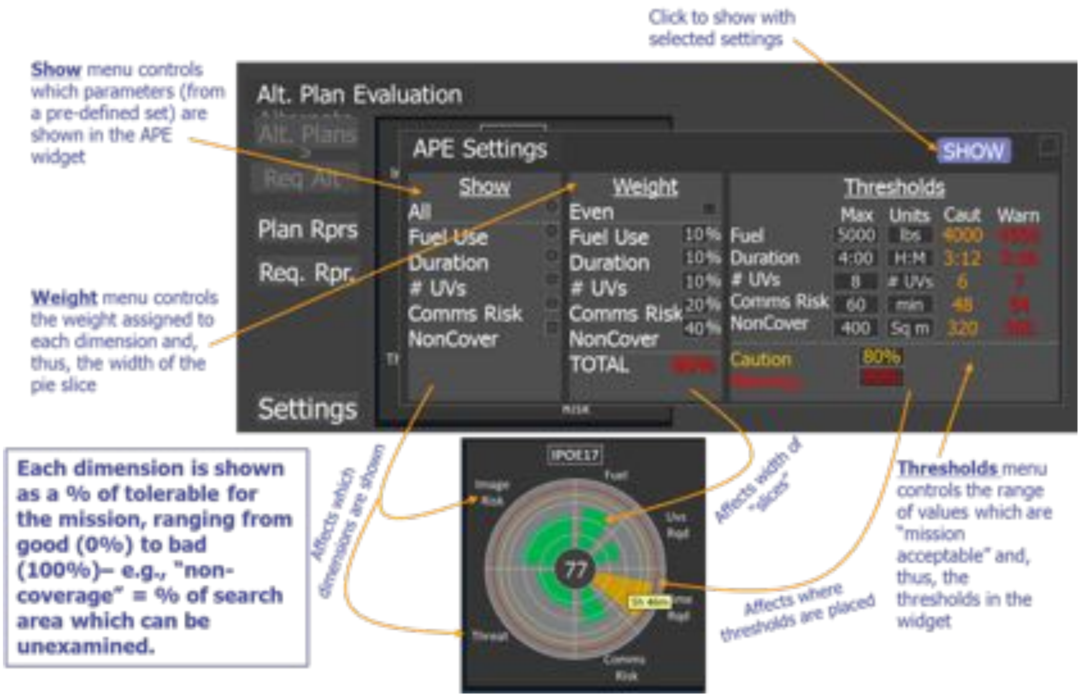

Contrasting alternate plans, shrink and expand to include more plans
Move mouse or touch to present actual calculated values in dimension-relevant units
All dimensions scaled so more is worse
Multiple dimensions, user selectable and weightable
Overall probability of success
User-/unit-settable, dimension-specific thresholds
IPOE17
IPOE18
Fuel
Image Risk
UVs Rqd
Time Rqd
Comms Risk
Threat
77
68
5h 46m
3h 58m

Figure 14. APE Widget Basic View

The APE Widget is a sprocket display similar to the initial AFRL Play Quality Pane, and is, in fact, built on the same basic software. The center of the display conveys the overall estimated probability of success of the plan. Each slice or sprocket conveys how the plan is projected to do along a dimension of interest within caution and warning thresholds. Slices are normalized in such a way that expanding slices always convey more of the dimension and relatively worse outcomes. The thresholds (caution in yellow and warning in red) can be set, but the dimensional score is meant to be conveyed as a percentage of some maximal acceptable value. This means that the dimensions cannot be arbitrarily chosen, but must sometimes be re-defined to be conveyed by the slice dynamics. For example, instead of conveying simple pounds of fuel expected to be used, a slice for Fuel Use is configured to be presented as the percentage of an allowable maximum range. Similarly, instead of the degree of coverage of an area during a search play, we would use a Non-Coverage percent. Note, though, that the actual value for a dimension can be presented via a Tool Tip pop-up at any time. While we acknowledge that there may be a need to revise this display to convey minimal thresholds and/or to convey positive increases in a dimension, we have not yet found a compelling need to do so. We also note that recent feedback has critiqued the use of *larger* green colored slices to convey relatively *worse* outcomes—a violation of most cockpit and MIL STD conventions. We believe this is a valid concern and can be easily corrected by either replacing the green with a translucent white or light gray color, or by omitting the full fill-in of slices that don't reach warning or caution thresholds.

Controls for setting up dimensions and their thresholds in the APE Widget are shown in Figure 15. In essence, the user (probably pre-mission) can set up the dimensions to be shown and can adjust the thresholds for them, perhaps on a mission-specific basis. Again, this is another area where our core philosophy of allowing but not requiring or expecting user input is used. Dimensions and thresholds appropriate to each play are expected to be predefined well before use and specific mission planning and then to be rarely changed, but the opportunity exists for the user to alter these values if desired.


Show menu controls which parameters (from a pre-defined set) are shown in the APE widget

Weight menu controls the weight assigned to each dimension and, thus, the width of the pie slice

Click to show with selected settings

Each dimension is shown as a % of tolerable for the mission, ranging from good (0%) to bad (100%)– e.g., "non-coverage" = % of search area which can be unexamined.

Affects which dimensions are shown

Affects width of "slices"

Affects where thresholds are placed

Thresholds menu controls the range of values which are "mission acceptable" and, thus, the thresholds in the widget

Weight		Thresholds	Max	Units	Caut	Warn
Fuel Use	10%	Fuel	5000	lbs	4000	[illegible]
Duration	10%	Duration	4:00	H:M	3:12	[illegible]
# UVs	10%	# UVs	8	# UVs	6	7
Comms Risk	20%	Comms Risk	60	min	48	[illegible]
NonCover	40%	NonCover	400	Sq m	320	[illegible]
TOTAL	[illegible]	Caution		80%		

Figure 15. APE Widget Controls

In Figure 14, we show two APE Widgets side by side. We expect this to be a common usage when comparing multiple plans or plan repairs. In addition, we have designed controls to review and select from among multiple plans or plan repairs, potentially tens of alternatives. In that context, a single sprocket view is also used to show the range of values across a large number of plans and the user can drag sliders along each slice to specify the range of values that are acceptable for each dimension – thereby pruning the set of acceptable plan repairs.

C. Play Execution Monitoring

Various components of the Map Display are used to visualize both the planned (pre-mission) and actual (in-mission) execution of a play. Here, we provide a brief discussion of just a few of these elements. It is important to point out that the CODE concept presents many challenges to the HSI design, such as:

a. *How to de-clutter the display with a large number of vehicles?*
b. *How to compactly describe teams of heterogeneous vehicles?*
c. *How to associate vehicles and teams with targets or tasks?*
d. *How to zoom in on smaller areas without losing context?*
e. *How to visualize and understand when and how tasks will be completed?*
f. *How to best notify the user of significant events, that have either already occurred or are about to occur?*
g. *How to properly account for uncertainty of plan execution when comms are denied?*
h. *How to convey to the user the communications topology (current and/or expected) among all vehicles?*

While we have developed in-depth HSI designs to address each of the above challenges, in this paper we provide only a brief overview of a subset of elements, focused on items *a*. through *e*.

1. Aggregate Icons

Aggregate icons were created as a means of showing groups of platforms using less screen space (see). Aggregate icons are shown on the map display when multiple units, which are close to each other, share the same task. They are meant to de-clutter the screen, so the user can focus on other things. Aggregation of units is toggle-able, by clicking on or touching the icon.

In the innermost part of the icon, smaller unit icons are used to describe the aggregate's number of units of that type (e.g., letters to distinguish platforms). Alternatively, a single number may be used to simply indicate the number of platforms in the team, which is the method shown in subsequent map views. The direction these icons face is the average directional movement of the aggregate. The color around these smaller icons indicate the unit's secondary task, if any, white being used to convey a planned but not-yet-executing play. The color of the outer circle describes the aggregate unit's primary task. If the outer circle is dashed, one or more of the member UAVs is currently out of communication with the ground station. If there is a rotating arc around the edge of the main icon, it indicates that the aggregate unit is carrying out the core part of its primary task. This may be defined on a task by task basis, but generally refers those portions of the UAV's mission *other than* ingress and egress. On core is meant as a cue to the operator that this vehicle or aggregate is worth paying more attention to. There is an optional dashed white circle beyond the On Core symbol, which conveys that a new play is planned (but not yet executing) which will replace this aggregate's previous primary task. Finally, the small but wide arrow around the edge of the main icon points at the direction of the target area which may, of course, not be the direction the aggregate is currently heading.

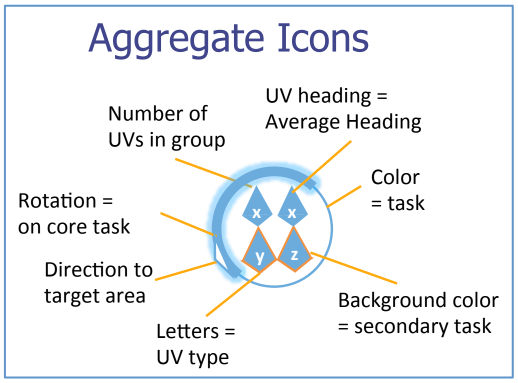


Figure 16: Aggregate Icon Design

2. *Planned Routes*

As a way of showing the planned routes for individual UAVs, SuperC3DE made use of a baseline Fusion functionality with minor modifications which we came to call thin routes. These lines can be both solid and dashed. If solid, they were shown in the vehicle/task color. If dashed, they were either colored (indicating that the UAV had an assigned route that it was expected to be executing but lack of communications made verification currently impossible – and perhaps we were projecting that route via SCOPE's CSI function (see Section 7.3.3). A dashed white line meant that the path was proposed and not yet accepted and active. An example of thin line, individual *proposed* routes for a group of UAVs is shown in Figure 17 below.

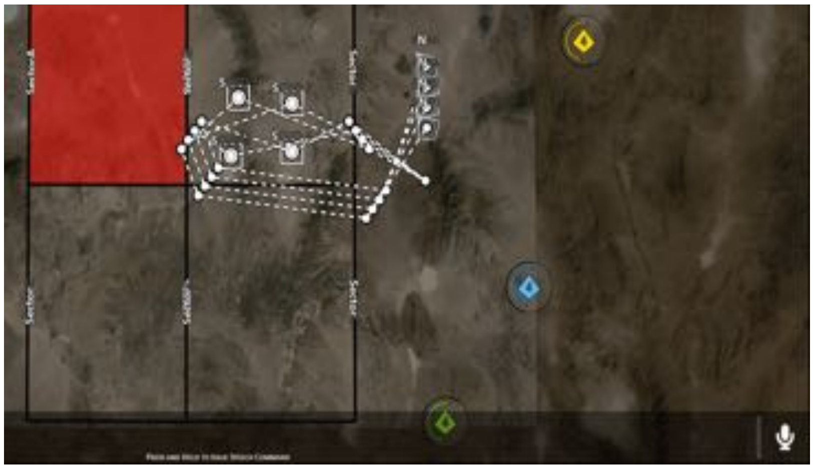


Figure 17. Proposed Routes for a Group of UAVs

3. *Fat Routes and Target Icons*

An interesting expansion on the unit aggregation icon is a larger arrow that we've termed the Fat Route. Showing individual thin routes for each vehicle quickly produces a "spaghetti" effect when more than a few UAVs are involved (as is already becoming apparent in Figure 17 with only four UAV routes expanded). Thus, the Fat Route was developed to show the route and travel behavior of an aggregate set of UAVs. Each Fat Route indicates the aggregate's current direction and route, with hashes representing five minutes of travel each. Figure 18 below gives an example from the notional IPOE/XINT scenario. Note that each aggregate icon points to a target symbol (circle) that exists within a task-appropriate region outlined using same color as the related play. In this case, rectangles show the region in which the IPOE task is to be carried out. This figure also provides an added example of both Aggregate Icons and their target symbols, since the orange colored aggregate icon represents a group of UAVs performing two concurrent tasks – both one sector of the IPOE search (the blue rectangle) and an XINT task involving searching for Scuds in the region of the orange oval. The fact that these vehicles are multi-tasked is conveyed by the blue outline in addition to the orange fill of the individual vehicles in the aggregate icon.

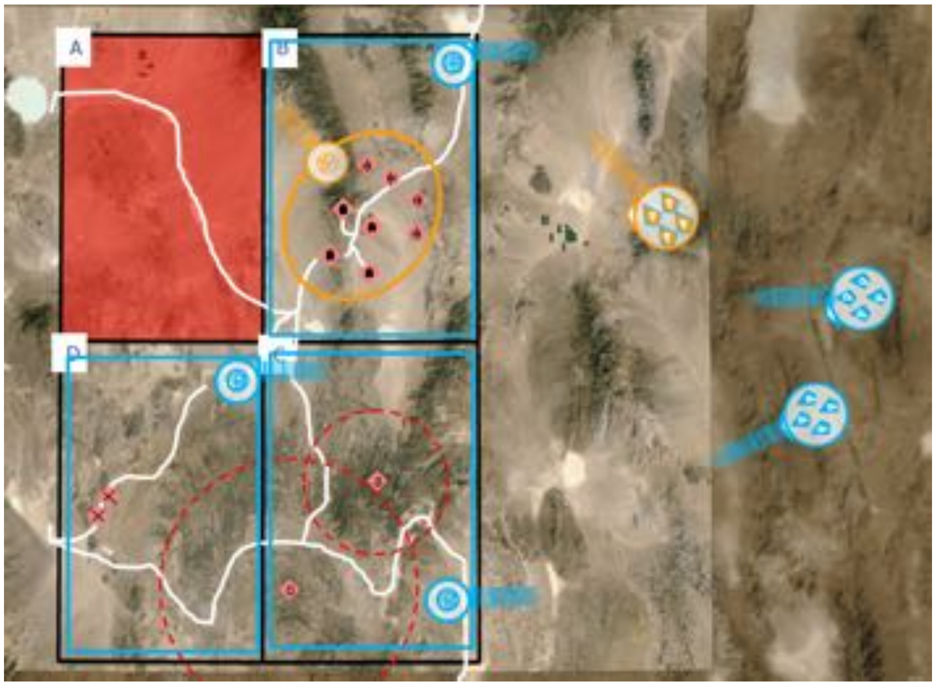
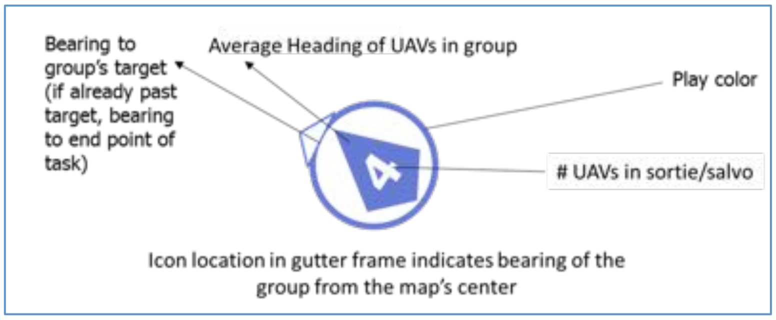


Figure 18. Fat Routes and Target/Task Icons Example

4. Gutter Functionality

Figure 19 below shows another feature designed to enhance the map display: a peripheral edge known as the gutter designed to improve SA and provide easy access to UAVs and task groups that might be outside the current pan and zoom settings for the map. The gutter is displayed as the outer side of the dashed grey line around the edges of the map display. It uses a further miniaturized version of the UAV or aggregate icon to indicate the direction and direction of travel of UAVs that are not currently on-screen. Clicking on an icon in the gutter immediately pans the screen to center on that group (and results in other UAVs, likely including those formerly on screen to now appear as gutter icons). Gutter icons appear at the bearing of the vehicles in that cluster relative to the center of the screen and, thus, in the direction one would have to look or travel from the screen's center to encounter those vehicles. Gutter icons will move around the screen's edge if the screen center moves or if the vehicles in the cluster change their bearing to the screen center. Once a UAV or aggregate UAV group is in the gutter, it does not move laterally within the gutter and no indication of its total distance from the current map center is provided, though it may rotate as it shifts direction of travel.



Bearing to group's target (if already past target, bearing to end point of task)
Average Heading of UAVs in group
Play color
# UAVs in sortie/salvo
4
Icon location in gutter frame indicates bearing of the group from the map's center

Figure 19. Map Display Gutter

Figure **20** illustrates the appearance and several of the behavioral features of the Gutter. An alternate implementation of the gutter transition behavior was suggested during our interim reviews. This would have a click or touch on one of the icons in the gutter shift the map not directly to that group, but rather to a zoomed-out view that included the UAVs currently on screen as well as those tapped by the operator. A subsequent user action could then pan and zoom the map on the new group if desired. We have not implemented or tested this option, largely due to time constraints, but suspect that the need to perform two actions to view a new UAV group would be disfavored by operators. On the other hand, a behavior that rapidly animates the transition from the current screen focus to the new requested one—combining pan and zoom in one animated motion which leaps out from the current view to an all-encompassing view and then immediately zooms in to the new view—might well be a desirable improvement. Such an implementation would combine the single click access of our current implementation with some of the global map-based SA of the two-step mode described above.

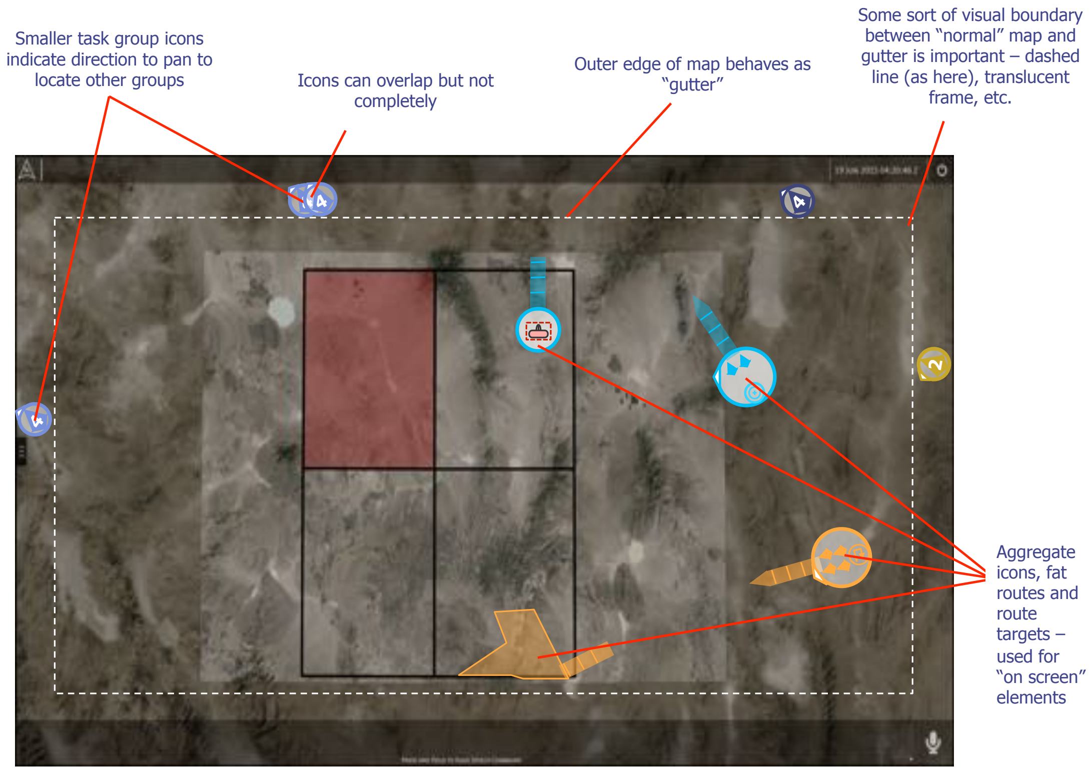

Smaller task group icons indicate direction to pan to locate other groups
Icons can overlap but not completely
Outer edge of map behaves as "gutter"
Some sort of visual boundary between "normal" map and gutter is important – dashed line (as here), translucent frame, etc.
Aggregate icons, fat routes and route targets – used for "on screen" elements

Figure 20**. Gutter Behavioral Features**

5. *Sensor Coverage Overlay*

As a commandable overlay the user can call up (and turn off) when desired, we created a Projected Sensor Coverage Overlay as a view overlaid on the map using a translucent variation of the play color to convey the region inspected by the UAVs in that play over time. The degree of translucency corresponds to the latency of coverage (more color saturation conveys more recent inspection) and time is controlled via a slider bar – or could be allowed to run at a multiple of real time as a movie. Figure 21 illustrates these features with multiple vehicles inspecting different quadrants (and leaving alternate sensor trails; the time slider bar is included in the lower left).

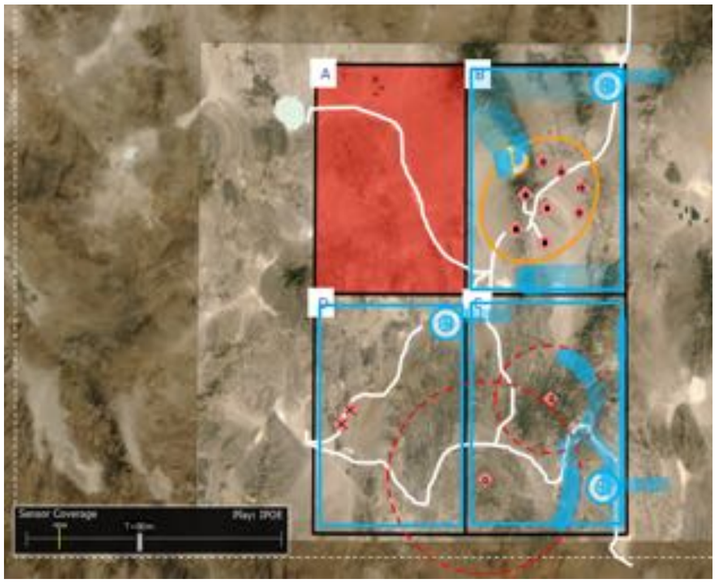


Figure 21. Projected Sensor Coverage Overlay

V. Evaluations

We performed a series of evaluations on both the HSI and SCOPE components. In this section, we describe the objectives and overall approach of the HSI evaluations, and provide a summary of the results.

A. Formative Review Approach

We held a formative review of the in-progress design of the HSI and SCOPE in July 2015. Formative reviews are so-called because they help to inform the design during its formative stages, and thus, are done on a partial design still somewhat in flux. Indeed, some textbooks recommend that they be done using pencil and paper sketches of the designed HSI elements so as not to convey the image that designs are fixed and will be difficult to modify.

Our formative review was held with four subject matter experts as described below. Three had experience in current UAV operations – two as pilots, one as a mission intelligence officer. The fourth was an F-16 pilot familiar with the CODE program and CONOPS.

- **SME1**. Former F-16 pilot. Familiar with the CODE CONOPS and with future tactics for onboard control of multiple UAVs. Volunteer.
- **SME2.** Six years' experience supporting UAV operations as an intelligence operations supervisor and mission intelligence coordinator. Paid participant.
- **SME3**. Five years' experience as a UAV pilot. Paid participant.
- **SME4**. More than 2,000 hours as a UAV pilot. Paid participant.

The materials reviewed in this exercise were the slides, videos, and live demonstration created for and presented at one of our technical interchange meetings. At this time, we had a well-articulated vision and multiple examples, but the designs were not yet fixed. This allowed sufficient time to react to feedback from SMEs and to revise our designs accordingly. Note that the designs described previously in this paper were completed after this review (which was performed on earlier versions of these designs). The designs presented above take advantage of inputs from the SMEs who participated and are thus, expected improved versions of the core concepts.

The review was held in a large conference room at SIFT's Minneapolis headquarters with the four SMEs, as well as some SIFT employees. The format, as explained to the SMEs, involved an initial presentation of the goals of CODE and our designs, followed by a presentation of the overall system via the live demonstration of its behavior in a representative mission. SME participants were then asked to complete the first block of four questions in the questionnaire below. We then presented each major component of the HSI design in detail, showing slides, videos and/or live demonstrations of the component behaviors. After an initial presentation of each element, we opened the floor for discussion and clarification of the design, with SIFT employees taking notes. After discussion, SMEs completed the associated block of questions for that component. Following this extended series of demonstrations and discussions at the component level, we again asked the original set of overview questions to see if any changes in opinion had occurred. This process took approximately eight hours.

The following sections detail the questionnaire the SMEs responded to and summarize both their quantitative responses and their comments.

B. Questionnaire and Response Summaries

For the questionnaire, we used a 1-10 rating scale for participants to rate their impression of different aspects of design. The scale was defined such that: 1 = Potentially Harmful, 5 = Neutral, and 10 = Extremely Helpful. We posed a unique set of questions for each of the following aspects/components of the SuperC3DE HSI design:

- Overview/Summary (scores for the overall HSI design)
- Map Display and Comms Status
- Timeline Display
- Play Calling and Review (Quality) Panes
- Active Play Summary Display (Playble)
- SCOPE Breakdown and Repair Table
- Overview/Summary (repeated)

While the full questionnaire is too long to include in this paper, we do, however, provide the set of Summary/Overview questions and responses in Appendix A. These questions were asked to the SMEs participating in our formative review. Each question is listed on a row of the table, with the individual SME response scores and transcriptions of their written comments provided in subsequent columns in that row.

C. Summary of Results

Overall, we collected 151 ratings from the four SMEs during the day-long Formative Review. The average rating was 7.8 and the standard deviation was 1.8 on a 10-point scale ranging (as shown above) from 1 = Potentially Harmful through 10 = Extremely Helpful. The minimum score given for any HSI feature was a 3 and the maximum was a 10. This (combined with both written and verbal comments) can be taken as general support for our designs with a few specific critiques and requests for revisions. The ratings of our different design elements appear in Figure 22.

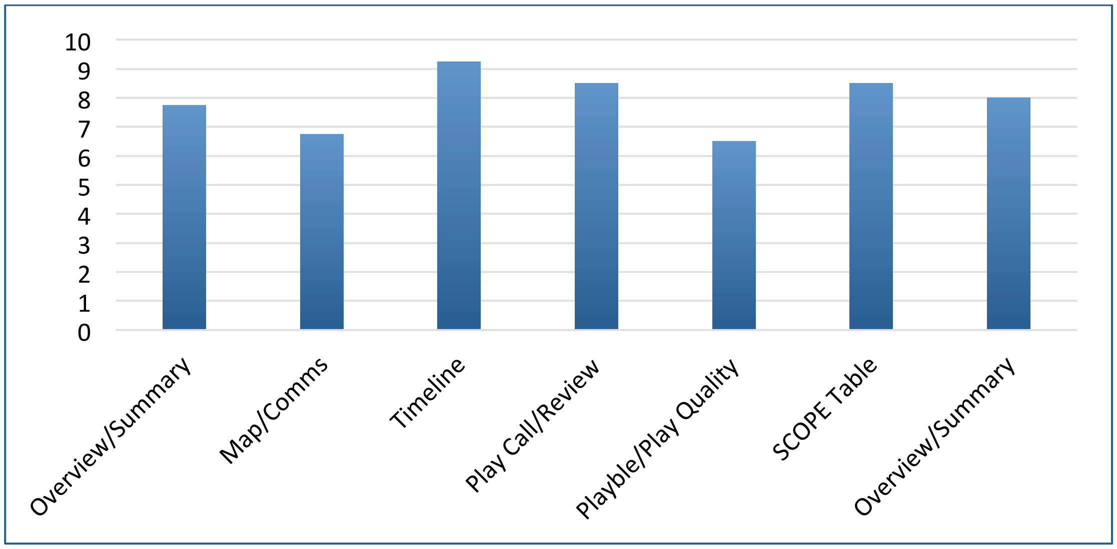
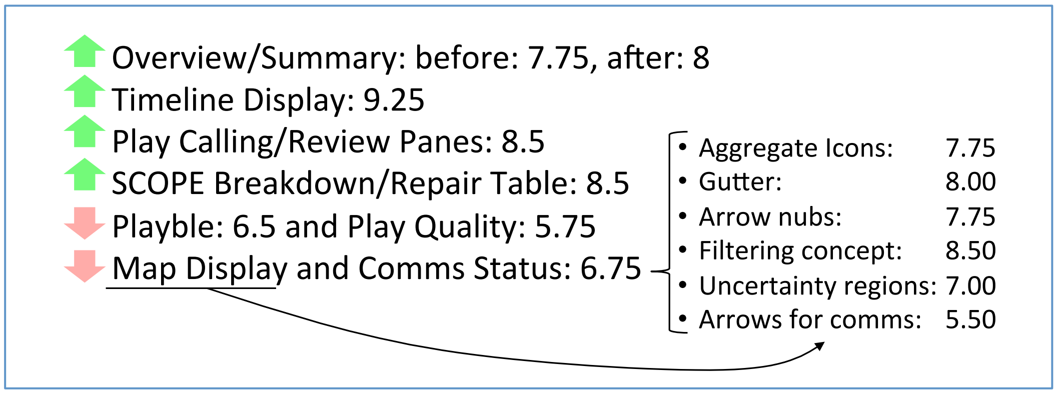


Figure 22. HSI Design Element Ratings

General impressions from this review are summarized in Figure 23 with the green arrows indicating elements that were favorably reviewed and the red ones indicating elements that needed redesign, improvement or replacement.



- ⬆ Overview/Summary: before: 7.75, after: 8
- ⬆ Timeline Display: 9.25
- ⬆ Play Calling/Review Panes: 8.5
- ⬆ SCOPE Breakdown/Repair Table: 8.5
- ⬇ Playble: 6.5 and Play Quality: 5.75
- ⬇ Map Display and Comms Status: 6.75

• Aggregate Icons:	7.75
• Gutter:	8.00
• Arrow nubs:	7.75
• Filtering concept:	8.50
• Uncertainty regions:	7.00
• Arrows for comms:	5.50

Figure 23. Formative Review General Impressions

Numerical values are the average of responses to sub-questions asked within each broad HSI element category. Note, that (as illustrated for the map display), the sub-questions sometimes pointed to a specific element that needed to be redesigned rather than overall satisfaction or dissatisfaction with the HSI element as a whole.

General take away themes from the discussions held during the formative review are included below along with specific actions we took in subsequent design phases (as described above) to address the SMEs' concerns.

1. De-clutter (or better, don't clutter in the first place)
   - Providing information on demand was seen as a better approach instead of offering de-cluttering options or controls.
   - *Action:* Substantial redesign to reduce general clutter, especially through the introduction of callouts and popups and use of the pie menu to access related content.
2. Play Calling "coincides with real-world operations" and "feels intuitive," but SMEs wanted to understand the consistency in play symbols.
   - *Action:* Some symbols were redesigned, but the consistency which was there was better explained as well. The introduction of Tool Tip mouse-over labels for play-calling icons also helped to address this issue.
3. Aggregate icons and gutter features were very well regarded – but should be simplified.
   - *Action:* Aggregate icons were greatly simplified in later designs – both in larger form on the map and in smaller form as a part of the gutter.
4. Timeline display "has enormous SA building potential;" notifications should record actual events.
   - *Action:* Considering for future implementation.
5. Map display needs play-specific filtering options and combinations.
   - *Action:* Organizational tools were added to the display to enable custom filtering of displayed entities.
6. SCOPE outputs were "very useful" as a "double check" on planning, but expected value numbers were hard to interpret.
   - *Action:* Overall expected value numbers were abandoned and were replaced using more concrete numbers (e.g., fuel remaining, image quality, etc.) with mission-specific thresholds in later APE design.

We also asked the SMEs, as a take-home exercise, to complete a comparative timeline analysis in which they were to think through the tasks required to complete a scenario we described (planning and executing an XINT task with four UAVs) as well as a rough estimate of the time required to complete each task both with and without CODE/SuperC3DE. This analysis was completed by three of the SMEs. The resulting data, though admittedly subject to a number of different biases, provides intriguing, if preliminary, evidence for the benefits of the CODE system overall and of our HSI in particular. Across the three data sets prepared for a four UAV XINT mission against four Scuds, the average "fully occupied" minutes during mission execution (not counting flying time) for a human operator was expected to be 23 minutes for a baseline system (without CODE/SuperC3DE) but only 5-8 minutes for the same mission using CODE/SuperC3DE. Thus, we can (with appropriate caveats for the preliminary and subjective nature of this data and process) say that CODE/SuperC3DE has the potential to reduce human workload and time required by 65-78 percent.

VI. Conclusions

We set out to achieve several goals in our development of SuperC3DE:

- Improve human-system performance and trust while minimizing human workload and system bandwidth requirements.
- Enable mission commanders to rapidly and flexibly convey their intent to and maintain awareness of multiple UAVs, even when they are distracted or harried and, most importantly, when communications are restricted.
- Extend and improve performance under Vehicle and Team Autonomy by integrating them more closely with HSI using counter-planning, contingency delegation, and contingency status inferencing.
- Provide novel situation awareness tools, integrated with the HSI, to make commanders more aware of and involved in UAV operations, even under comms denial.

To meet these objectives, we designed an architecture for SuperC3DE comprised of a holistic HSI, a plan-critiquing tool (SCOPE), and a FIT testbed for user-interactive demonstrations. We used AFRL's Fusion environment as a springboard to rapidly prototype new HSI concepts. The FIT testbed enabled interactive testing with multiple simulated vehicles via its connection to a configurable AMASE simulation. We leveraged prior designs from the FLEX-IT and IMPACT work with which we had been involved at AFRL to serve as the basis of our integrated, task-centered architecture for commanding and maintain awareness of many UAVs. Our design included several novel display concepts for maintaining situation awareness in the challenging environment of CODE missions, which involved multiple diverse vehicles with uncertain threat and target information in a denied environment. Recognizing the inherent uncertainty that the commander and planning system will face before mission execution, we developed SCOPE as a planning and plan review aid. Given a plan and a model of our actions and the world, SCOPE will find contingencies within that uncertainty space that will cause the plan to fail, and it will generate repairs in the form of new branches to make the plan resilient to those contingencies. The primary interaction between the HSI and SCOPE is performed through SCOPE's APE, whose design was in response to suggestions from SMEs at a review of an earlier, more complex version of a SCOPE display, which evaluates a set of alternate plans and scores them along multiple dimensions to estimate the plan's performance in various areas, such as mission completion time, threat exposure, or total number of vehicles required.

Moving forward, we plan to build on the above accomplishments in ways that would both further the CODE program objectives, as well as extend our research in the general areas of decision support and planning for semi-autonomous systems. By integrating as fully as possible with Track A architectures, we intend to advance the Technology Readiness Level of selected SuperC3DE components through live flight demonstrations.

Appendix

Table 1: Overview/Summary Questionnaire with Responses – Initial

		Rating	Comments
1	**Overall impression** of SuperC3DE HSI design:	7,7,8,9	SME1—Good initial design, some interface seems cumbersome SME2—Straightforward map and timeline views; overall seems optimized for GCS/OPS[5] center use SME3—High volume of data available in finite space, high level of organization. Some icons are very similar. SME4—Looks very nice. Might be a little busy on a small display.
2	Use of **tasks** as an organizing, controlling & reporting structure:	7,5,8,7-8	SME1—I like the playbooks, but I think they'd need to be more of a mission planning function and less of an execution function SME2—Play creator is fairly straightforward requiring only brief guide or legend for identifying various play options – speed and proficiency will come over time. SME3—Structure seems well thought out. Ability to control groups or individual RPAs feels intuitive SME4—I like the ability to build scan/flight plan functions. Map and CSI are good. Some of the contingency variable predictions might be a little busy, but ok to include.
3	Use of **maps AND timelines** as core organizing views	8,9,9,9	SME1—Easy to follow along as the msn/cc[6] might want to combine routes into 1 with offshoots to de-clutter screen. SME2—Maps and timeline views very reminiscent of real-world tools, thus facilitating easier implementation and use SME3—Division of attention will likely be one of the biggest limiting factors, this overview has a very good situational awareness increasing potential SME4—Nice way to organize the map and various input functions
4	Does the SuperC3DE HSI hold promise to enable controlling **many unmanned air systems (UASs)?**	8,7,9,10	SME1—With some reductions on required inputs from msn/cc during execution, I think it'd be more feasible. SME2—Definitely SME3— Yes, with some familiarity with the system I see a lot of potential here. SME4—Yes. Looks similar if not much better to software we use today.

[5] "GCS" refers to the Ground Control Station, while "OPS" refers to Operations.
[6] "msn/cc" refers to mission command and control.

Table 2: Overview/Summary Questionnaire Responses – Repeated after full description of each component

		Rating	Comments
1	**Overall impression** of SuperC3DE HSI design:	7,8,9,8	SME1— We gave you a lot of inputs, but I think you have a solid base and the ability to expand on it to have an awesome product. SME2—Very good initial approach. Main recommendation focuses on cleaning up display. SME3—I really like the way information is displayed to the user. With some mods, this will be very useful. SME4—Some details need work but overall great idea and concept
2	Use of **tasks** as an organizing, controlling & reporting structure:	--,10,--,7	SME1—No change SME2—Coincides with real-world operations—both planning and execution SME3— SME4—Need some work on play menu, but well organized
3	Use of **maps AND timelines** as core organizing views	--,9,10,9	SME1—Maps are key, with data on demand SME2—Very effective at conveying mission essential information only requiring some map display icon clean up/simplification SME3—Using primarily the map and timeline as SA tools the display is de-cluttered and efficient. SME4—Love the timeline, map is good too
4	Does the SuperC3DE HSI hold promise to enable controlling **many UASs?**	--,10,--,9	SME1—Definitely SME2—Definitely SME3—The most important thing is having the right information when you need it, but only the information you need at that time. I think this is a big step in the right direction. SME4—Very useful for planning/controlling a large number of vehicles

Acknowledgments

This work was developed with funding from the Defense Advanced Research Projects Agency (DARPA) through AFRL contract FA8650-15-C-7508.

In addition to our funding sponsors, the authors would also like to thank our colleagues at AFRL's Supervisory Control & Cognition Branch, 711th Human Performance Wing: Gloria Calhoun, Mark Draper, Heath Ruff, and Allen Rowe. Their prior work in this field, including research under the IMPACT and FLEX-IT programs and the Fusion software, was instrumental to our work on this contract.

References

[1]Miller, C. and Parasuraman, R. "Designing for Flexible Interaction between Humans and Automation: Delegation Interfaces for Supervisory Control." Human Factors, 49(1), 57–75, 2007.

[2]Parasuraman, R., Galster, S., Squire, P., Furukawa, H. and Miller, C. "A Flexible Delegation-Type Interface Enhances System Performance in Human Supervision of Multiple Robots: Empirical Studies with RoboFlag." IEEE Systems, Man and Cybernetics—Part A, Special Issue on Human-Robot Interactions, 35(4), 481–493, 2005.

[3]Kidwell, B., Calhoun, G., Ruff, H., Parasuraman, R. "Adaptable and Adaptive Automation for Supervisory Control of Multiple Autonomous Vehicles." In 56th Annual Human Factors and Ergonomics Society Meeting, pp. 428–432. HFES Press, Santa Monica, 2012.

[4]Miller, C., Draper, D, Hamell, J., Calhoun, G., Barry, T., and Ruff, H. (2013). Enabling Dynamic Delegation Interactions with Multiple Unmanned Vehicles; Flexibility from Top to Bottom. Lecture Notes in Computer Science Volume 8020, pp 282-291.

[5]Behymer, K., Rothwell, C., Ruff, H., & Patzek, M. (in preparation). Initial Evaluation of the Intelligent Multi-UxV Planner with Adaptive Collaborative/Control Technologies (IMPACT). Technical Report AFRL-RH-TR-2017-TBD. Air Force Research Laboratory.

[6]Calhoun, G.L., Ruff, H.A., Behymer, K.J., & Mersch, E.M. 2017. Operator-Autonomy Teaming Interfaces to Support Multi-unmanned Vehicle Missions. Chapter in Advances in Human Factors in Robots and Unmanned Systems: Advances in Intelligent Systems and Computing 499, edited by P. Savage-Knepshield & J. Chen (pp. 113-126). Springer International Publishing.

[7]Rowe, A.J., Spriggs, A., & Hopper, D. (2015). FUSION: A Framework for Human Interaction with Flexible-Adaptive Automation across Multiple Unmanned Systems. In Proceedings of the 18th International Symposium on Aviation Psychology.

[8]Duquette, M. "Effects-Level Models for UAV Simulation." In proceedings of AIAA Modeling and Simulation Technologies Conference, Chicago, IL, Aug 10–13, 2009.

[9]Duquette, M. "The Common Mission Automation Services Interface." In proceedings of Infotech@Aerospace, St. Louis, MO, March 29–31, 2011.

[10] Miller, C., Shaw, T., Musliner, D., Hamell, J., Emfield, A. and Parasurman, R., (2011). Delegation to Automation: Performance and Implications in Off-Nominal Situations. In *Proceedings of the 14th International Conference on Human Computer Interaction* Orlando, FL. 9-14 July 2011.

[11] Maver, L. A., Erdman, C. D., Riehl, K. "National Image Interpretability Rating Scales." URL: http://fas.org/irp/imint/niirs.htm, updated on January 16, 1998. Accessed January 31, 2016.

[12] Lemmer, J. F. and Gossink, D. "Recursive noisy-or: A rule for estimating complex probabilistic causal interactions." *IEEE Transactions on Systems, Man, and Cybernetics*. Reference Number: SMCB-E-10152003-0493.R1, 2004.