REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From - To)*
14-04-2011	Technical Paper	MAR 2011 - APR 2011

4. TITLE AND SUBTITLE

CAUSE Multi-UAV Simulation Demonstration

5a. CONTRACT NUMBER
FA8720-05-C-0002

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Herbert E. M. Viggh, Christopher Weed, Michael T. Chan, and Daniel J. Van Hook

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

MIT Lincoln Laboratory
244 Wood Street
Lexington, MA 02420

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFLCMC/PZE
20 Schilling Circle, Bldg 1305
Hanscom AFB, MA 01731

10. SPONSOR/MONITOR'S ACRONYM(S)

AFLCMC/PZE

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A. Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Intelligence Surveillance and Reconnaissance (ISR) systems have the potential to operate in a net-centric environment in which tasking control, resulting sensor data, and intelligence feeds can be made available via web services under a service oriented architecture. Rapid composition of ISR applications from such services would enable analysts and warfighters to quickly adapt to changing missions and ISR asset availability. A preceding paper described the development of a new composition approach in which plug-ins that encapsulate service interactions are used to rapidly compose applications using a Graphical User Interface (GUI). The composition GUI allows the user to discover needed service plug-ins, connect outputs and inputs between services, and generate a consolidated application GUI. This paper describes a demonstration of this composition approach utilizing simulated ISR assets. Service plug-ins were developed for accessing a simulated Signals Intelligence (SIGINT) data feed, a simulated Electronic Order of Battle (EOB) data feed, and a SIGINT to EOB correlation service. Additional ping-ins were developed for tasking multiple unmanned air vehicle (UAV) simulations supporting full motion video (FMV) and for accessing a resource brokering service. The composition Gill was then used to compose various applications that enabled FMV data collection by the UAVs on SIGINT and EOB locations of interest, utilizing both manual tasking and automated tasking via the resource brokering service.

15. SUBJECT TERMS

component; service composition; SOA; service oriented architecture

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
					Zach Sweet
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**	SAR	6	**19b. TELEPHONE NUMBER** *(include area code)*
U	U	U			781-981-5997

MS-57041

CAUSE Multi-UAV Simulation Demonstration

Dr. Herbert E.M. Viggh, Christopher Weed, Dr. Michael T. Chan, and Daniel J. Van Hook
MIT Lincoln Laboratory
Lexington, MA USA
{Viggh, CWeed, MChan, DVanHook}@LL.MIT.EDU

Abstract—**Intelligence Surveillance and Reconnaissance (ISR) systems have the potential to operate in a net-centric environment in which tasking control, resulting sensor data, and intelligence feeds can be made available via web services under a service oriented architecture. Rapid composition of ISR applications from such services would enable analysts and warfighters to quickly adapt to changing missions and ISR asset availability. A preceding paper described the development of a new composition approach in which plug-ins that encapsulate service interactions are used to rapidly compose applications using a Graphical User Interface (GUI). The composition GUI allows the user to discover needed service plug-ins, connect outputs and inputs between services, and generate a consolidated application GUI. This paper describes a demonstration of this composition approach utilizing simulated ISR assets. Service plug-ins were developed for accessing a simulated Signals Intelligence (SIGINT) data feed, a simulated Electronic Order of Battle (EOB) data feed, and a SIGINT to EOB correlation service. Additional plug-ins were developed for tasking multiple unmanned air vehicle (UAV) simulations supporting full motion video (FMV) and for accessing a resource brokering service. The composition GUI was then used to compose various applications that enabled FMV data collection by the UAVs on SIGINT and EOB locations of interest, utilizing both manual tasking and automated tasking via the resource brokering service.**

Keywords-component; service composition; SOA; service oriented architecture

I. INTRODUCTION

This paper describes the second year results of a two-year research project into rapid composition of applications under a service oriented architecture (SOA), with the goal of allowing non-software developers to compose applications in days or hours. The general approach involves interconnecting plug-ins that hide the details of service interactions, and is referred to as Composable Applications Using Service Encapsulation, or CAUSE.

The first year development and demonstration results were presented in [1]. That paper described an approach to SOA application composition that uses service plug-ins that encapsulate the details of the service client. The first year demonstration involved a simple use-case in which static image collections by a UAV were cued from Signal Intelligence (SIGINT) reports, and was demonstrated using very simple SIGINT and image collection simulations. For the convenience of the reader, this paper summarizes the problem description and approach from the first paper, and briefly describes the development and demonstration environment used. The second year development and demonstration activities are then described in detail. These demonstrations involved a more complex use-case, two high-fidelity UAV full motion video (FMV) simulations, and several new services including a SIGINT-to-EOB correlation service and a task brokering service.

This work is sponsored by the United States Department of Defense under Air Force Contract # FA8721-05-C-0002. Opinions, interpretations, recommendations and conclusions are those of the authors and are not necessarily endorsed by the United States Government.

II. PROBLEM

Intelligence Surveillance and Reconnaissance (ISR) systems have the potential to operate in a net-centric environment in which tasking control, collected sensor data, and intelligence feeds can be made available via web services under a service oriented architecture (SOA). ISR software applications used by analysts, decision makers, and warfighters will need to connect to these services and send and receive.

Problems in the ISR domain often involve rapidly evolving threats, opponent tactics, and ISR sensors. Rapid composition and modification of ISR applications would enable quick reaction and adaptation to changing missions and ISR system availability. Enabling ISR application users to compose and modify applications themselves will generate the fastest adaptation possible.

It is currently very difficult for a non-software developer to compose a SOA application from existing services. This typically involves a team of software developers who discover and connect to the existing services, learn enough domain knowledge to effectively use the services and the data they provide, and then write or modify the application itself to connect to the service and use it. This process can take weeks to months.

III. APPROACH

A. *Service Plug-ins*

In the CAUSE approach, a plug-in concept is used, similar to a web-browser plug-in for a movie player. With such plug-ins, a non-software developer can compose a new application that combines the movie player with the browser, without needing to understand the details of the underlying service interactions.

Generalizing this approach, the authors developed plug-ins that have the following characteristics:

- Each plug-in contains the service client code to connect to one or more services, encapsulating and hiding the details of the service connections from the user.

- Each plug-in contains a graphical user interface (GUI) for interacting with the service, which allows the user to input information and action requests needed by the service, and to display resulting data and status from the service interaction. This allows each plug-in to operate stand alone as a mini-application.
- Each plug-in has standardized input and output ports that can be connected to other plug-ins, and mirror the inputs and outputs of the GUI.
- Both the service plug-ins themselves and the I/O ports are annotated semantically.
- Each plug-in also contains help files and other domain specific information to allow the user to come up to speed on the use of the service and data without interacting with the service provider.
- Plug-ins are stored in a registry similar to a service registry that can be searched by the user based on semantic descriptions.

Figure 1 depicts the architecture of the plug-ins, and how they fit into an overall architecture for composing and executing applications using the CAUSE approach.

Plug-ins are composed of two types of components. The User Interface Components are needed for the user to interact with the plug-in. These include a GUI and domain specific information for educating the user on the correct use of the service. The Service Interface Components encapsulate the service client code, include standard Input/Output (I/O) ports for communicating with the GUI components and other plug-ins, and contain semantic specification information describing the service and the I/O ports. Note that the encapsulation of the service client code allows for the use of heterogeneous service technologies within the same application. A notional Resource Brokering service is depicted as the third service, which performs automated service composition using only the Service Interface Components.

B. *Compostion and Execution Architecture*

Figure 1 also depicts the Composition GUI, which allows the user to discover needed plug-ins in the Plug-in Registry and graphically connect I/O ports among plug-ins to compose applications.

The Composition GUI also has a toolbox of local processing blocks that facilitate the interconnection of service plug-ins. These include pre-built blocks such as data filters and consolidated GUIs to replace multiple individual plug-in GUIs.

The Execution Framework shown in Figure 1 takes the composed application description and runs the resulting application.

C. *CAUSE Benefits for Software Developers*

While the CAUSE approach is aimed at non-software developers, software developers will find it useful as well. For service developers, plug-in templates could provide a quick way to develop a stand-alone mini-application to test one's service, and a test application could be composed to test planned interactions with other services. Likewise, end-user application developers could use CAUSE to rapidly prototype applications. In an open-source environment, the plug-ins can serve as example code to help the developers create optimized applications.

IV. CAUSE Demonstration Environment

Development and demonstration of the CAUSE concept was done at MIT Lincoln Laboratory as part of a research project on service composition.

A. *KEPLER Composition GUI and Execution Framework*

The open-source KEPLER version 1.2 [2] scientific workflow application was used to demonstrate the CAUSE approach and is described in detail in [1]. KEPLER actors, which represent java code blocks with I/O ports, were used to implement service plug-ins. KEPLER provides a composition GUI that allows non-software developers to drag and drop connections between actors' ports to create workflows. KEPLER also provides a Plug-in Registry with a search (discovery) capability and an execution framework for running an application composed as an actor workflow.

B. *ISR Use-Case*

The following use-case was selected to drive the development of the second year CAUSE demonstrations described in this paper. An analyst needs to gather full motion video (FMV) of locations of select SIGINT reports. Of particular interest are those SIGINT reports that come from known emitters stored in an Electronic Order of Battle (EOB). Initially, only a single UAV is available to be tasked for FMV. Later, a second UAV becomes available, and the user needs to make use of both UAVs to collect FMV.

To task a single UAV, the analyst needs to:

- Discover a SIGINT data feed service plug-in
- Discover an EOB service plug-in
- Discover a SIGINT-to-EOB correlation service plug-in
- Pass the SIGINT reports and EOB info to the correlation service plug-in
- Discover a UAV service plug-in that provides platform position, sensor pointing information (SPOI), a FMV tasking interface, and FMV data
- Pass the SIGINT reports, EOB locations, SIGINT-EOB correlations, and UAV position and SPOI to a consolidated display in which the user can click on correlations of interest for FMV tasking
- Display the resulting FMV

To task a second UAV, the analyst needs to:

- Discover a second UAV plug-in

- Discover a resource brokering service that optimally tasks each UAV
- Pass the user selected FMV tasking to the resource brokering service
- Display the resulting FMV from both UAVs

C. *Services and Plug-ins*

For this demonstration, several types of service plug-ins were developed. The SIGINT and EOB Plug-ins access Simple Object Access Protocol (SOAP) [3] / Web Service Description Language (WSDL) [4] services that provide the endpoints of User Datagram Protocol (UDP) [5] feeds for each type of data. The SIGINT UDP feed sends simulated SIGINT detections in eXtensible Markup Language (XML) [6] schema. Similarly, the EOB UDP feed provides a-prior locations of simulated known emitters. The Correlator Plug-in connects to a correlation service that has a SOAP/WSDL interface for both setting and querying for the UDP endpoints for input and output. The correlation service takes the EOB and SIGINT reports inputs and outputs an alert when a SIGINT report is determined to be correlated with an emitter in the EOB. Location, location uncertainty, and emitter type are used in the correlation algorithm.

The UAV SPOI-Tasking Plug-in interfaces to two services. First, it accesses a SOAP/WSDL services that provides the endpoint of the UDP feed that continuously reports the UAV's position and the FMV sensor pointing information (SPOI). The SPOI includes the center and ground footprint of the sensor field-of-view (FOV). Second, the plug-in queries a data feed registry using a SPARQL [7] query over SOAP to obtain the endpoint for a UDP feed of streaming FMV.

Two independent UAV Plug-ins were implemented to connect to two independent UAV FMV sensor simulations. These simulations were developed by MIT Lincoln Laboratory under a prior program and use the commercial VRSG Meta-VR environment [8].

Finally, an RB Tasking plug-in was developed to access a resource brokering service [9] that provides capabilities for discovering and tasking information resources such as sensors, processors, mediators, archived data, networks, and communication systems. This RB Tasking plug-in uses a SOAP/WSDL service to obtain the endpoint of a UDP channel to the resource broker that accepts a latitude-longitude tasking request in Cursor on Target (CoT) [10] format, along with a requested sensor type in CoT. The UAV sensor simulations support both electro-Optical (EO) and Infra-Red (IR) video sensors. The resource broker interfaces to both UAV simulations, monitors their positions, and sends the tasking request to the UAV closest to the tasked point of interest.

V. Development and Demonstration

In this section we step through the use-case and discuss the applications composed to demonstrate the CAUSE approach.

A. *Correlated SIGINT Cueing of FMV from a Single UAV*

The first part of the use-case involves tasking FMV from a single UAV and was completed through the composition of the application embodied in the workflow depicted in Figure 2. This workflow tasks a specific UAV to collect FMV on a SIGINT-EOB correlation selected by the user.

At the top left and center of Figure 2 are located the EOB, SIGINT, and Correlator Plug-ins, which were discovered by the user in the Plug-in Registry and loaded into the composition GUI. Note that each plug-in has three output ports: a 'data' output (emitter type and location, SIGINT reports, and correlation locations for the three plug-ins, respectively), a 'display' output of the same data but in a format compatible with toolkit displays, and 'status' information. Using the composition GUI, the user connected the EOB and SIGINT data outputs to the proper inputs of the Correlator Plug-in.

Note that the EOB and SIGINT Plug-ins each come with their own independent data displays. In order to view the EOB locations, SIGINT reports, and locations of any correlations on a single display, the user first selected a Nondeterministic Merge local processing block from the composition GUI toolkit and then connected the 'display' outputs of the EOB, SIGINT, and Correlator Plug-ins to the input of the merge block. Note that connecting the 'display' outputs of the SIGINT and EOB Plug-ins to another actor suppresses their independent data displays. Next, the user selected a User Interactive Display component from the composition GUI toolkit and connected the merged display output to the User Interactive Display input port. This display is a Graphical Information System (GIS) type display that plots icons at latitude-longitudes (lat-lons) on a map, and was implemented using ESRI's ArcGIS Engine Java API (Ver 9.3) [11]. The display is interactive in the sense that the user can click on the displayed icon of a correlation, EOB, or SIGINT report and the lat-lon position of the icon will be sent to the display's 'selection' output port as Graphics Markup Language (GML).

Now that the user can select correlations of interest, they must be sent to a UAV as FMV tasking. In the lower left of Figure 2 is depicted the UAV SPOI-Tasking 1 Plug-in for the first UAV FMV simulator, which was discovered by the user and downloaded. The user connected the 'selection' output of the User Interactive Display to the 'GML Tasking' input, one of two formats supported by the UAV plug-in. This tasking is in a Geography Markup Language (GML) format used by the ArcGIS display. This tasking causes the UAV FMV sensor to slew to and stare at the tasked lat-lon. The 'display' output of the UAV plug-in was then connected to the input of the Nondeterministic Merge block so that the UAV position and SPOI are also displayed on the User Interactive Display for situational awareness. Note that the collected FMV is displayed in the UAV plug-in's independent FMV display, since a non-GIS video display is required.

Figure 3 shows the run-time GUI generated by the composed application in Figure 2. The GIS User Interactive Display is on the right and the UAV plug-in's FMV display is on the left. In the GIS display, EOB locations are displayed as red diamonds. SIGINT reports are shown as yellow clovers with question marks, plus a red error ellipse. The UAV's

position is indicated by a light blue Mil-Std-2525b 'Pac-Man' icon with a single chevron. The UAV SPOI is represented by a a green outline of the FOV footprint on the ground with a yellow clover indicating the center of the FOV. Correlations between SIGINT and EOB are displayed as green squares, and one is present just above of the center of the map. Clicking on the green correlation icon would task the UAV's FMV sensor to slew to location. The resulting FMV of that location would then be shown in the video display.

B. Tasking Multiple UAVs

The second part of the use-case involves tasking FMV collections from two available UAVs. One option for the user would be to modify the workflow in Figure 2 by adding a second UAV plug-in connected via a second merge block to a second User Interactive Display, and connect the outputs of the SIGINT, EOB, and Correlator Plug-in to that display as well. The user would then click in the GIS display of the UAV that the user decides to task. Such an application was successfully demonstrated under. However, adding more UAV's causes a corresponding increase in the number of GIS displays, which quickly uses up the available display area of the computer monitor. In addition, the task of deciding which UAV to request FMV from increases the users workload and will often not result in an optimal tasking sequence.

A better approach would be to have a single GIS display that shows the EOB, SIGINT, correlations, and both UAV positions and SPOI, with user selected correlations sent to a resource brokering service that automatically determines which UAV to task. Such an application was also successfully demonstrated.

Figure 4 depicts a similar application that utilizes resource brokering, but has been simplified for clarity by removing the EOB and SIGINT-EOB correlation plug-ins. One can think of this as solving a simpler use-case where the user tasks two UAVs to look at selected SIGINT reports. To compose this workflow, the user created a similar workflow as in Figure 2 (minus the EOB and Correlator Plug-ins), but added the UAV SPOI-Tasking 2 Plug-in for the second UAV. The user then connected its 'display' output, which carries position and SPOI data, to the Nondeterministic Merge block so that these data will also be displayed in the single GIS User Interactive Display. Note that the PN-Director block is a part of the KEPLER execution framework and is required for any workflow as described in [1].

Finally, the user discovered and downloaded the RB Tasking plug-in that sends the tasking request to the resource brokering service. This service selects the best UAV to task based on a simple optimization algorithm that chooses the UAV closest to the SIGINT report of interest. The UAV selection could also take into account other conditions, such as whether or not a UAV is busy collecting a minimum amount of FMV on their previously tasked lat-lon.

Figure 5 shows the run-time GUI generated by the multi-UAV application in Figure 4. Note the single GIS User Interactive display, which can support an arbitrary number of UAVs while minimizing the screen area used. There is, however, a separate FMV display from each UAV plug-in.

VI. CONCLUSION

The CAUSE approach was developed to enable non-software developers to rapidly compose service-based applications under a service oriented architecture. This is accomplished via a composition GUI that allows the user to discover and download service plug-ins and interconnect their I/O ports, and adding other application components such as consolidated GUI displays.

In the first year of this research effort, the CAUSE architecture was developed, the KEPLER development environment selected, and initial demonstrations done using a few services driven by simple sensor simulations. In the second and final year of the project, high fidelity UAV FMV sensor simulators were used, and correlation and resource brokering services were added to demonstrate a more complex use-case.

Several areas of future work are worth pursuing. While Kepler provided an existing environment for demonstrating the CAUSE approach, composition and execution architectures are needed that support both analysts on enterprise level networks and mobile users on disadvantaged networks. Finally, while the plug-ins are assumed to handle security, authentication, and identification, further work on plug-in architectures that support these features is needed.

REFERENCES

[1] Viggh, H. E. M., Weed, C., Chan, M. T., Van Hook, D. J., "Composable Applications Using Service Encapsulation (CAUSE)", *Proceedings - IEEE Military Communications Conference MILCOM*, p 243-248, 2010, *2010 IEEE Military Communications Conference, MILCOM 2010*

[2] Kepler 2.0 Getting Started Guide, https://kepler-project.org/users/documentation

[3] http://www.w3.org/TR/soap

[4] http://www.w3.org/TR/wsdl

[5] http://en.wikipedia.org/wiki/User_Datagram_Protocol

[6] http://www.w3.org/TR/xml

[7] http://www.w3.org/2009/sparql/wiki/Main_Page

[8] http://www.metavr.com/products/vrsg/vrsgoverview.html

[9] Van Hook, D. J., Ljungberg, M., Shaw, R., Ford, M., Aubin, E., Konieczny, E., Lee, D. H., Brown, S.T., "Resource Brokering: Timely and Efficient Resource Allocation," SPIE Defense Security and Sensing Conference, April 5-9, 2010. Kirstan, Michael J., et.al., Cursor-on-Target Message Router User's Guide, MITRE Product MP090284, November 2009

[10] Kirstan, Michael J., et.al., Cursor-on-Target Message Router User's Guide, MITRE Product MP090284, November 2009

[11] ArcGIS Engine 9.3 Java API. http://www.esri.com/software/arcgis/engine

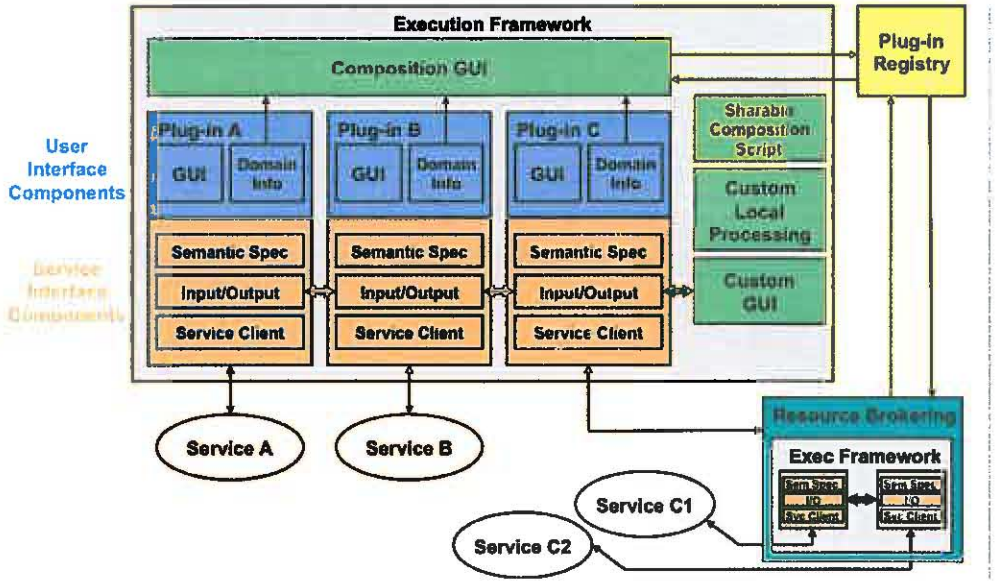

Figure 1. The interactions between Plug-ins, Composition GUI, Plug-in Registry, and Execution Framework.

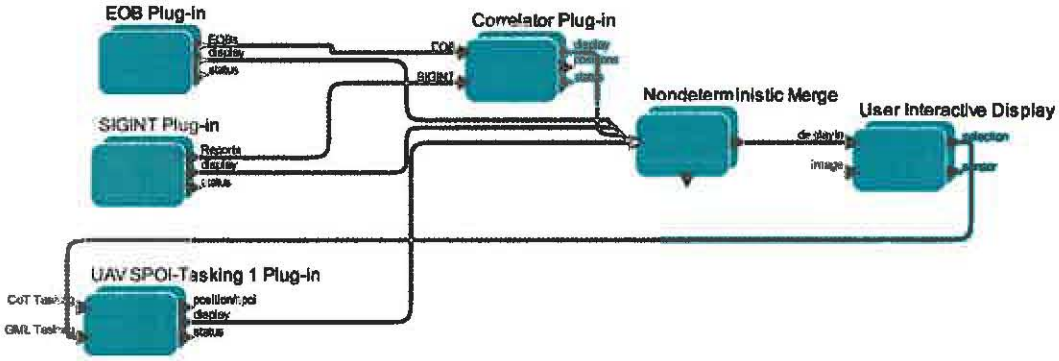

Figure 2. Composed application for tasking a single UAV to collect FMV on a SIGINT-EOB correlation of interest.

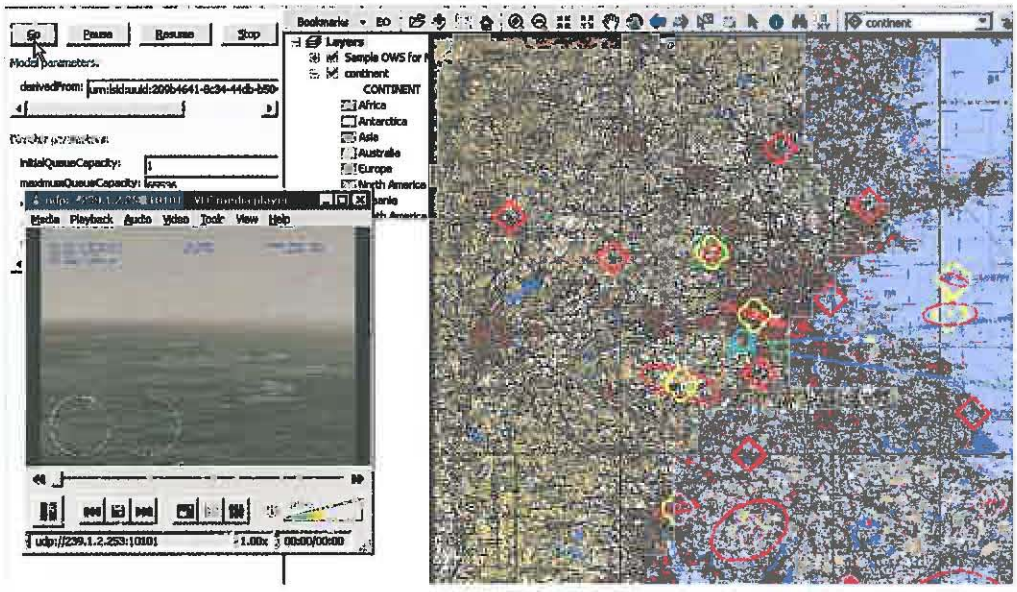

Figure 3. GUI Display of composed application for tasking a single UAV to collect FMV on a SIGINT-EOB correlation of interest.

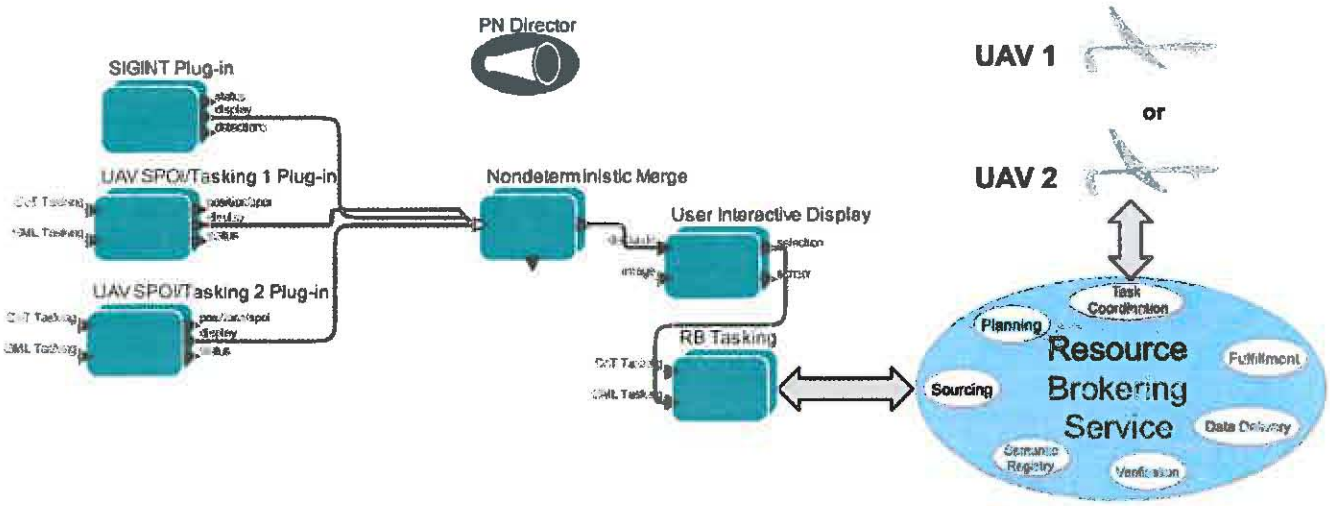

Figure 4. Composed application for tasking multiple UAVs via a resource brokering service.

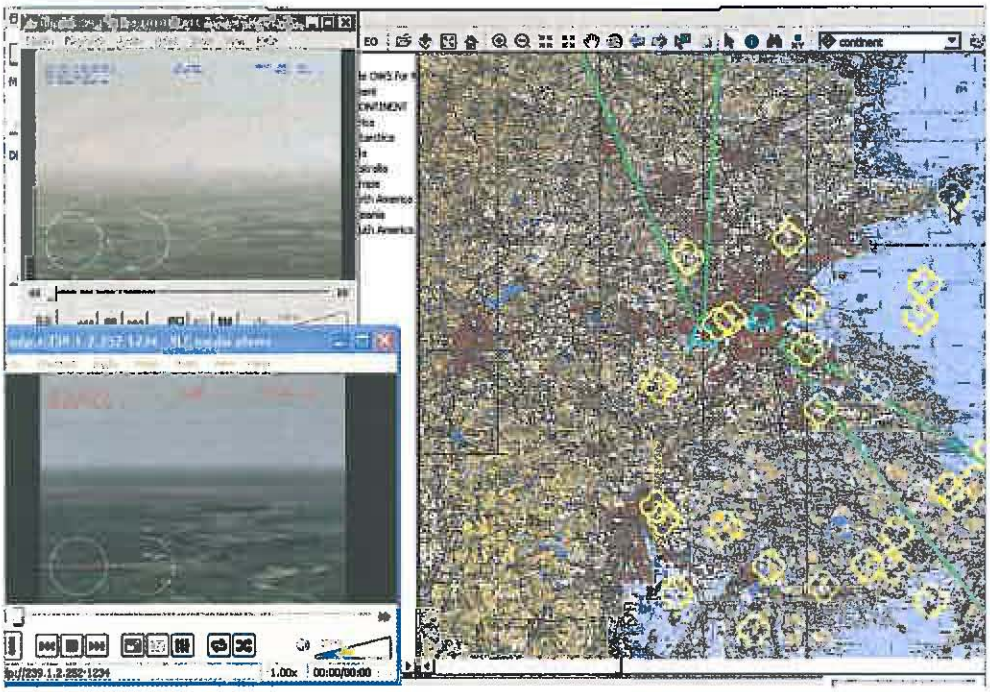

Figure 5. GUI display of a composed application for tasking multiple UAVs via a resource brokering service.