2018 NDIA GROUND VEHICLE SYSTEMS ENGINEERING AND TECHNOLOGY SYMPOSIUM
AUTONOMOUS GROUND SYSTEMS (AGS) TECHNICAL SESSION
AUGUST 7-9, 2018 - NOVI, MICHIGAN

DEMONSTRATION OF A COMMON WORLD MODEL

Mark Hinton
Gautam Vallabha
Chris Cooke
Christine Piatko
Michael Zeher
Peter Gayler
Geoffrey Osier
The Johns Hopkins University
Applied Physics Laboratory
Laurel, MD

ABSTRACT

A crucial part of facilitating the cooperation of multi-robot and human-robot teams is a Common World Model – a shared knowledge base with both physical information (e.g., ground map) and semantic information (e.g, locations of threats and goals) – that can be used to provide high-level guidance to heterogeneous robot teams. Past work performed by Johns Hopkins University Applied Physics Laboratory (JHU/APL) has shown that the Advanced Explosive Ordnance Disposal Robotic System (AEODRS) architecture – a Modular Open Systems Approach (MOSA) architecture leveraging the JAUS (Joint Architecture for Unmanned Systems) standard for definition of its logical interfaces – can be effectively used to develop and integrate the subsystems of a teleoperated ground vehicle for use in a complex environment. This demonstration tackles the next challenge, which is to extend the AEODRS architecture to facilitate multi-robot and human-robot teams.

Introduction

To facilitate the formation and operation of multi-robot and human-robot teams, JHU/APL has developed a foundation for a Common World Model. This Common World Model has several components: a distributed World Model Data Store (WMDS), world ontology and autonomy algorithms that consult and update the WMDS, and a means for replicating data in a controlled manner throughout the nodes using the WMDS. Implementation of the WMDS as a distributed data store allows autonomy algorithms to run in the same space as their local world model,

improving efficiency of operation. We refer to such a World Model as a Common World Model when the model, its data sharing mechanisms, and its data representations are shared by all nodes in a system or system of systems.

In this effort, the Common World Model team at JHU/APL has developed a three-tier World Model architecture, comprised of a set of JAUS-based World Model Services, a WMDS interface based on extensions to the JAUS Transport Service [4], and a World Model data sharing mechanism built on a publish-subscribe paradigm. The key contribution of the current effort is the design of the distributed data store and related standardization efforts. These are building blocks for future work in the development of ontology and autonomy algorithm components.

The Common World Model team at JHU/APL is collaborating with a World Model Working Group within the SAE committee responsible for the ongoing development of the JAUS standard (SAE Technical Committee AS-4JAUS), to develop extensions to the JAUS standard that define the interface to the internode data sharing means, and to define the interfaces and protocol behaviors [3] of World Model Services. When complete, these extensions will be submitted to the standards body as proposed additions to the standard. This standardization of both application interfaces and internode interfaces enables data sharing across homogeneous or heterogeneous nodes in a system, or across systems, and also facilitates integration of world model capabilities within a node.

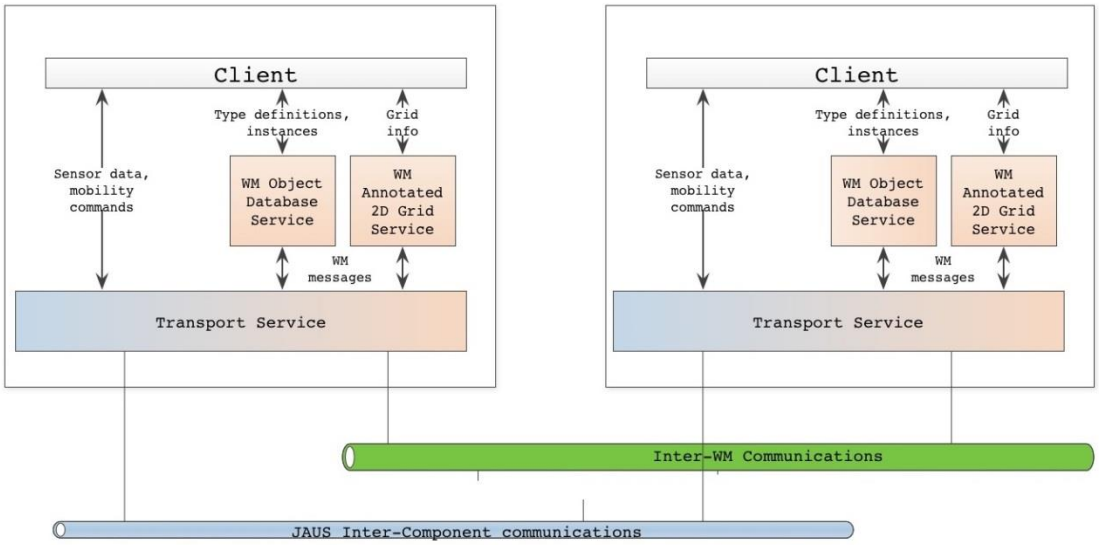

Figure 1 - Common World Model Concept

Figure 1 depicts the three-tier architecture of our World Model solution, and the role of the WMDS and its interfaces within that solution. The three tiers of the architecture are the World Model Services tier, a Transport Service

interface tier, and the selected data transport channel (shown as the "Inter-WM Communications" in Figure 1).

The World Model is exposed to application level clients through a set of World Model Services. These services provide capabilities to their clients; in Figure 1, two such services are identified: *WM Object Database Service*, and *WM Annotated 2D Grid Service* (a 2D occupancy grid map service that provides for annotation labeling of grid cells). Each World Model Service exposes its topics through publication; any other World Model Service may subscribe to any published topic. Publication and subscription are performed via World Model Transport, represented in Figure 1 in the Transport Service and the underlying Inter-WM Communications.

Proposed additions to the standard JAUS Transport Service interface provide the channel-agnostic interface between the local WMDS instance and remote WMDS instances via the inter-WM communications channel. Publication, subscription, and replication of data between WMDS instances uses the Transport Service to access the inter-WM channel. The proposed WMDS additions to the JAUS Transport Service provide a mechanism that is logically distinct from the standard JAUS transport interface and underlying communications channel. Separation of the standard JAUS Transport communications channel and the inter-WM communications channel allows a system designer to specify separate quality of service (QoS) regimes for the two channels.

For this demonstration effort, the team focused on the use of the Common World Model, and the adequacy of the proposed extensions to the JAUS Transport Service to provide the WMDS interface for an AEODRS-compliant [5,6,7] or other JAUS-compliant system.

Previous Work

Several attempts have been made by individual industry players, industry-government consortia [1], and by standards committees [2] to develop a Common World Model that would enable the sharing of map, threat, self, and other information between participants in a human-robot team.

Previous work [8,9] focused primarily on fusion of sensor information, what we are referring to as the autonomy algorithm component of a World Model. However, we are not aware of prior efforts related to the distributed data store, that is, efficiently distributing shared data amongst a set of mobile autonomous vehicles and identifying a candidate standard for interoperability between human and robotic nodes in a human-robot system.

There are a number of reasons that multiple efforts and approaches have failed to result in broadly-accepted, stable standards:

- The ontologies of different platforms' World Models tend to differ, particularly between differing classes of platforms or across platform domains. However, useful sharing of relevant physical and semantic information requires data interoperability across platforms and across sensor and input modalities.

- The processing requirements generally failed to scale down sufficiently for use on small UGVs and tactical UAS.
- Interoperability required the data transmissions be compatible with existing UGV control and data standards, which presented the challenge of integrating existing standards that tolerate data loss with the requirements of consistency implied by a Common World Model.

System Design

Our design has two key aspects: a distributed data store, and the interface it presents to clients.

While there is extensive work on distributed replication, it has mostly focused on server-centric paradigms (e.g., a mobile device communicating with a cluster of email servers, or replication among multiple servers for reliability). Our key requirement for the data store is that it should be suitable for operation in the field, which imposes a unique set of constraints: (a) support for multiple autonomous mobile vehicles, with (b) potentially heterogeneous compute, storage and networking capacities, (c) contested or congested communications, (d) no guarantee of globally ordered system time, and (e) no guarantee of a single coherent network (i.e., systems may unpredictably split into sub-networks that rejoin in an ad-hoc manner).

Our basic design is a peer-to-peer network that uses IP multicast for discovery. The database is a key-value store with a version vector for each item; each node also maintains a knowledge vector of its state [10]. Nodes periodically broadcast updates to each other; if a node discovers that it is too far behind (as determined by the knowledge vector), it makes a unicast request to get "caught up". All transactions are stateless. The design implements "eventual consistency", so nodes gradually converge to the same state, with the rate of convergence depending on network congestion and a node's own ability to receive and process updates.

As noted, the "World Model" is composed of one or more key-value stores. Each store has a unique "topic" name to enable routing of broadcasts and update requests. The keys form a tree-structured namespace (e.g., "/node/location/"), and each value is a JSON object. The clients interact with the World Model through purely local key-value operations: get the list of keys and add/update key-value pairs. Client operation efficiency benefits from interaction with local WMDS data; crucially, the clients do not need to know how the data is replicated.

The above scheme allows a very flexible way for nodes to inspect and update world model data. It is deliberately free-form, with the expectation that future work will impose a structured ontology and more complex interfaces (e.g., the 2-D map interface would be adapted to spatial tiles and related geometric primitives instead of raw key-value pairs).

System Operation

Figure 2, below, presents one possible message sequence illustrating the use of

inter-WM communications in replicating data between the WMDS instances in the system. Replication of data between WMDS instances enables each node's WM clients to interact with data local to the node, increasing client execution efficiency.

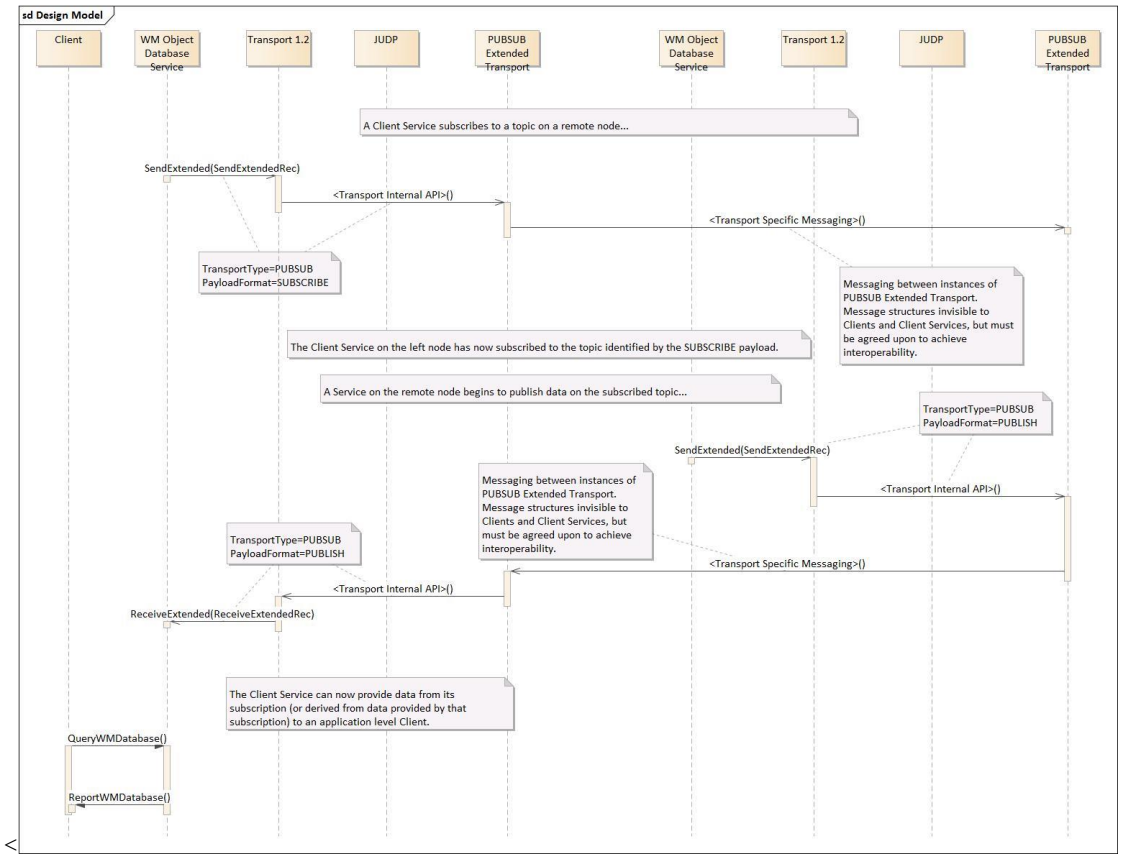

Figure 2 - Notional Message Sequence

The message sequence of Figure 2 begins with a world model service (in this case, the *WM Object Database* service) seeking to subscribe to a topic published by a corresponding service on another WMDS node. This request is sent to the client node's JAUS Transport service (which has been extended to implement the interface into inter-WMDS communications). The Transport service validates the request, determining that the request is for PUBSUB transport, and passes it off to the PUBSUB extended transport (which will be defined in the proposed WM extensions for the JAUS standard).

The PUBSUB transport delivers the subscription request via inter-WMDS communications, and the subscription is entered. The client on the left node has now subscribed to the topic identified by the subscription request in the PUBSUB message payload.

At some later point in time, a client service (in this case, the *WM Object Database* service on the right node) begins publishing data on the topic to which the left node subscribed. The client service requests publication by sending the request to its JAUS Transport service, which validates the publication request, determining that the request is for PUBSUB transport, and passes it to the PUBSUB extended transport.

The subscribing node's PUBSUB extended transport, on receiving a publication matching a subscription, delivers the publication payload by topic through the JAUS Transport service to the subscribed WM service (in this case, the *WM Object Database* service), and the data is replicated into the WMDS.

The WM client service can now provide data from its subscription (or information derived from data provided by that subscription) to a requesting application level client. Note that requests from application level clients are fulfilled with data residing in the WMDS local to the client.

Note that given the distributed nature of the WMDS, clients are always provided with data from the local WMDS, improving the responsiveness of the clients. In addition to execution efficiency, the underlying inter-WMDS transport achieves resilience to intermittent communication through the use of opportunistic replication; further, the decoupling of data replication from application request timing enhances scalability by reducing asynchronous application-level requests for remote data.

Operational Scenario for the Demonstration

To show that the three challenges noted above (data interoperability, scalability, and integration with existing standards) could be overcome within an open, standards-based architecture, the approach taken was to develop and demonstrate a system comprised of two UGVs, each with its own instance of the distributed WMDS with an extended JAUS Transport interface; two human robot operators, each human operator having an operator control unit with its own instance of the WMDS; a remote Analyst equipped with a laptop having an instance of the WMDS; and a System Commander also possessing a laptop having an instance of the WMDS.

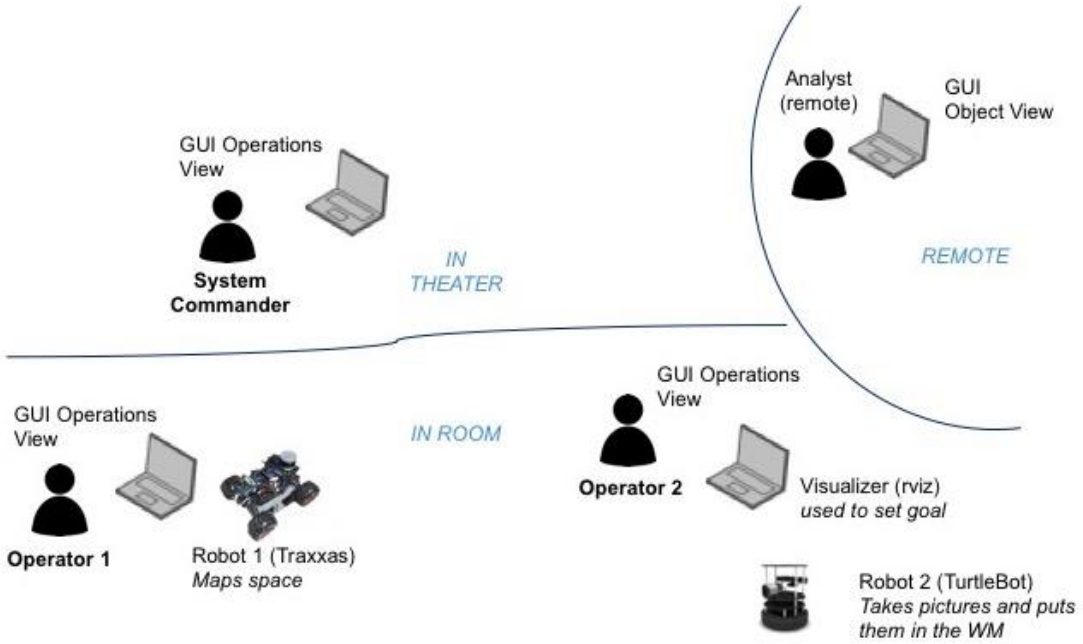

Figure 3 - Demonstration Participants

The roles of each of the demonstration participants (both human and robotic), and the information (both data and command) flowing between them is presented in Table 1.

Table 1 - Demonstration Participants and Roles

Actor	**Role**	**Actor Actions**	**Information Provided to Actor**
Operator 1	Commands the mapping robot	Enter self-location; set alert	World Model
Mapping Robot	Creates maps of area of interest	Self-locate; publish self-location; build map; publish map	World Model
Operator 2	Commands the surveillance robot	Command robot to Point of Interest (POI); command snapshot; teleoperate; enter self-location; set alert	World Model

Actor	Role	Actor Actions	Information Provided to Actor
Surveillance Robot	Capture images of area of interest as commanded	Self-locate; receive map; navigate to commanded POI; send snapshot image; publish self-location	World Model
System Commander	Directs human-robot team	Designate POI, Object of Interest (OOI); enter self-location; set alert; clear alerts; initiate/terminate operation	World Model
Analyst	Provides real-time expert evaluation of hazard of objects	Update OOI annotations and hazard status	World Model

The demonstration illustrated the use of separate systems, each with its own instance of the WMDS, updating and sharing information transparently through the use of the Common World Model.

For the demonstration, the mapping robot function was assigned to a teleoperated Traxxas Summit 1/10 scale RC car with an onboard Velodyne Puck LIDAR that produced an estimated 3D occupancy map and localized position using a particle filtered motion model. The 3D occupancy map produced was then projected onto a 2D occupancy grid for consumption by the surveillance robot. This simultaneous localization and mapping (SLAM) approach allowed the mapping robot to explore unknown environments and add previously occluded features as discovered; the global reference frame of the occupancy map enabled the mapping and surveillance robots to navigate in a shared environment. The surveillance robot function was assigned to a TurtleBot laboratory robot with Hokuyo LIDAR and a Microsoft 1080p HD LifeCam USB camera. The surveillance robot used a Dell quad-core Intel CPU laptop with Ubuntu 16.04 for onboard computation and sensor processing. Compressed raw RGB imagery was transmitted to the World Model for requested snapshots. ROS TurtleBot navigation and localization modules were used to localize and navigate using the map information transmitted by the mapping robot. The surveillance and mapping robots used are shown in Figure 4.

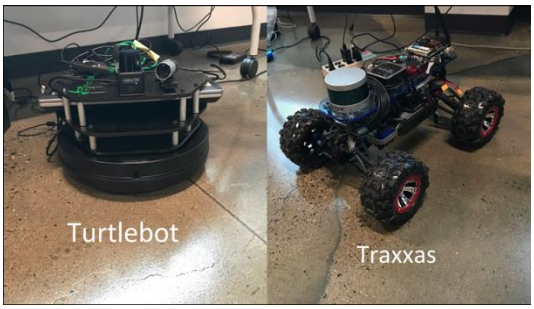

Figure 4 - Surveillance and Mapping Robots

Note that each robot operator was provided direct control of their respective

robot: Operator 1 (O1) had direct teleoperation control of the mapping robot; Operator 2 (O2) had the ability to send navigational and imaging-system commands to the surveillance robot. Aside from this, all communications amongst the team, robot and human, took place by an actor placing information into the World Model, and other actors taking information out of the World Model.

The World Model Transport for all actors was identical. The two robots implemented a JAUS stack that employed our proposed extensions to the JAUS Transport Service for sending information into the World Model Transport. The use of extensions to the JAUS Transport service for interface to World Model capabilities accelerated integration with our robotic assets. The graphical user interfaces (GUIs) used a RESTful interface into the World Model Transport.

Design of Demonstration

The demonstration was based on the concept of operation (CONOP) shown in Figure 5.

Note that the Demonstration CONOP reflects that each node (each robot and human interface station) possesses a local instance of the WMDS, and that the instances of the WMDS propagate data via subscription-based replication (shown in Figure 5 as "Inter-WMDS Communications"). Note that there is no central server; each node references data in its local instance of the WMDS; replication is transparent to the WMDS clients residing on that node.

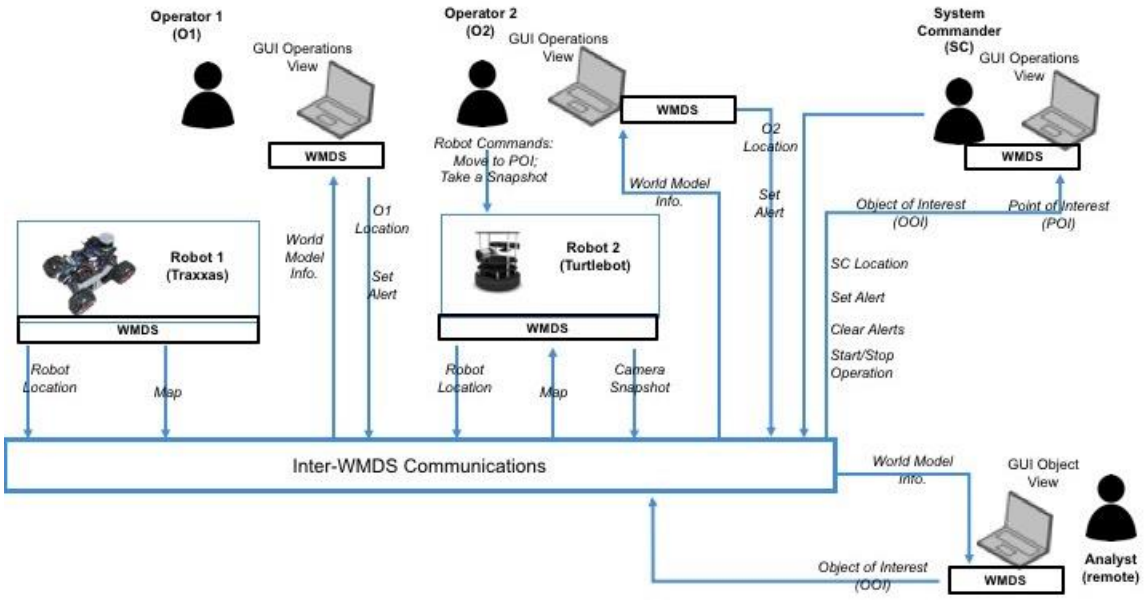

Figure 5 - Demonstration CONOP

The onsite team assembles outside of the building under investigation. All three human team members (both operators and the System Commander) possess separate computers running the World Model GUI, as does the off-site Analyst. The on-site team members are presented an operational view of relevant World Model data by a GUI on their workstations (typically laptops), while the Analyst examines the Stored Object View (a graphical view of World Model information associated with one or more objects) on his workstation.

Throughout the operation, as the on-site humans change their positions, they can update their position on the map by clicking the Update Position button on the GUI then clicking the map with their current position. If either O1 or O2 experiences some condition that keeps them from accomplishing their task, they can click the Start Alert button on their GUI. This will change their alert status in the node view. If they can resolve the issue, they can clear their alert, or the System Commander (SC) can clear all alerts.

The System Commander uses the "Assume Command" button on the GUI to enable system commander functionality in the GUI. When the System Commander is ready to start the operation, he presses the "Start Operation" button on his GUI. The operation indicator on all the other GUIs (including the Analyst) turns from red to green, indicating the operation should start.

O1 teleoperates the mapping robot into the building, which starts creating a map and placing it in the WMDS. The GUIs constantly read the current map from the WMDS and display it in the operations view. The surveillance robot also reads the map, which is then used by standard localization and navigation modules. O2 can assist localization by providing initial pose estimation within the map. As the mapping robot moves, it updates its position in the WMDS. This is reflected in the GUI. A screen capture of the System Commander GUI during map building is shown in Figure 6.

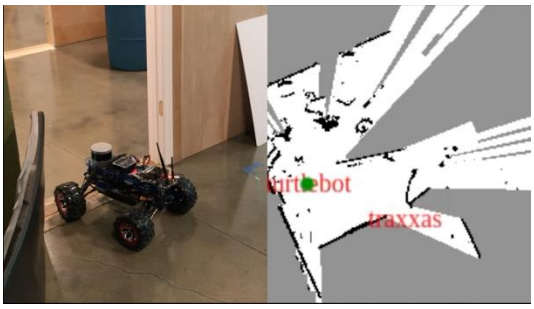

Figure 6 - Mapbuilding - System Commander GUI

As the map fills in, the System Commander identifies places on the map where he would like to acquire snapshots and clicks on the map to create Points of Interest (POIs): a screen capture of the POI designation activity appears in Figure 7. O1 and O2 see these POIs on their maps. O1 drives the mapping robot to fill in the map as needed to provide routes to POIs. O2 selects a POI based on distance to the surveillance robot and map quality and directs the surveillance robot to drive to the selected POI. As the mapping robot moves, it localizes itself in the map and updates its position in the WMDS, which is reflected in the GUI.

Figure 7- POI Designation

Once the surveillance robot reaches a POI, O2 directs the surveillance robot to take a snapshot. This snapshot is stored in the surveillance robot's local world model and is subsequently replicated to the local world models of the operators' laptops, the System Commander's laptop, and the Analyst's laptop. Each laptop displays the snapshot on the map as a thumbnail (Figure 8).

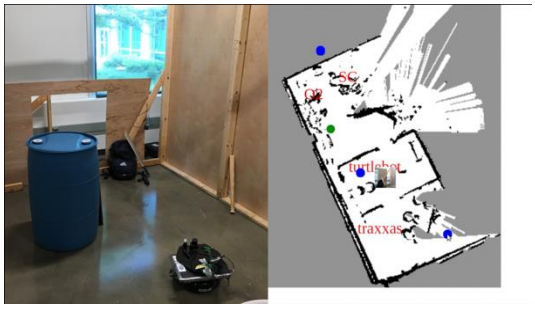

Figure 8 - Snapshot

The System Commander clicks on thumbnails to inspect them. If the System Commander finds the image to contain an Object of Interest (OOI), he can add notes and click a checkbox to indicate the object is a hazard (Figure 9). He can then store the object using a GUI button.

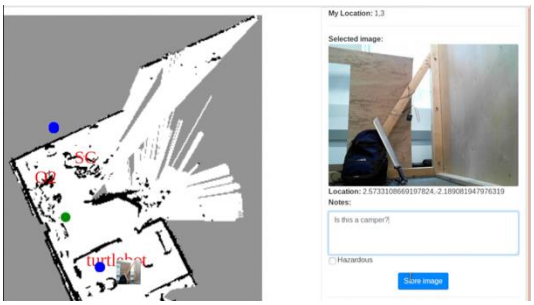

Figure 9 - System Commander Annotation

The Analyst will see objects appear in a table on the Stored Objects screen; this screen presents a list of objects observed and stored in the WMDS (Figure 10). He can click an object to get a bigger view for inspection and can update the notes and save the changes to the object.

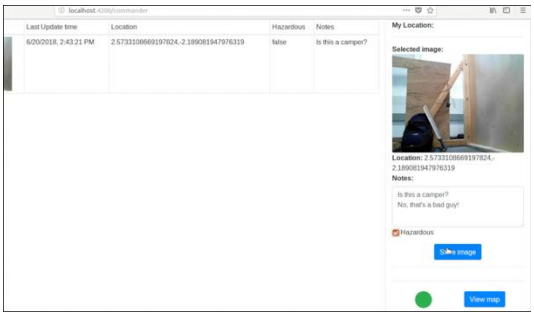

Figure 10 - Analyst Annotation

If the System Commander clicks on the thumbnail again, or goes to the Stored Object view, he will see the updates.

If the System Commander is satisfied the operation is complete or needs to end the operation early for some reason, he clicks the Stop Operation button in the GUI. The operation indicator on all the other GUIs (including the Analyst) turns from green to red, indicating the operation should end. O1 and O2 will withdraw their robots from the building and the entire on-site team will gather outside the building.

Results

The demonstration CONOP was performed at the Intelligent Systems Center of JHU/APL. The CONOP was performed as described above, with no remarkable deviations. Support for heterogeneous sharing of Common World Model information between a variety of robotic and human information producers and consumers was demonstrated. Limited "stress testing" was conducted by simulating additional WMDS instances; initial bandwidth limitations were identified and addressed at the scale of this demonstration.

This initial demonstration system will be extended and optimized in follow-on efforts. Such efforts will seek to gather performance data and assess scalability to large multi-robot / multi-asset systems, and will inform further development of the proposed world model extensions to the JAUS standard.

Conclusion

A Common World Model based on a distributed World Model Data Store (WMDS) can be developed, implemented, and meaningfully deployed on small, simple robotic platforms using a Modular Open Systems Architecture (MOSA) approach, specifically leveraging the Joint Architecture for Unmanned Systems (JAUS) standard and proposing World Model support additions thereto. In addition, it has been demonstrated that the WMDS can maintain a consistent world view, with the semantic labeling of objects of interest for two separate systems in the presence of inputs from both of those local WMDS hosts, and from multiple human-robot interfaces.

References

[1] Dean, Robert Michael S., "Common World Model for Unmanned Systems," in Unmanned Systems Technology XV, Proceedings of SPIE Vol. 8741, Article 87410O, SPIE, 2013.

[2] JAUS Reference Architecture Volume 3, "World Model Vector Knowledge Store", JAUS Working Group, Pittsburgh, PA, 2007.

[3] JAUS Service Interface Definition Language, Aerospace Standard AS5684A, SAE International, Warrendale, PA, 2013.

[4] JAUS Core Service Set, Aerospace Standard AS5710A, SAE International, Warrendale, PA, http://www.sae.org, 2013.

[5] M.V. Kozlowski, M. Hinton and M. Johannes, "Toward a Common Architecture for the Advanced Explosive Ordnance Disposal Robotic Systems (AEODRS) Family of Unmanned Ground Vehicles," 2010 NDIA Ground Vehicle Systems Engineering and Technology Symposium (GVSETS), Vehicle Electronics and Architecture (VEA) Mini-Symposium, Dearborn, MI, August 17-19, 2010.

[6] M. A. Hinton, M. S. Johannes, M. J. Zeher and M. V. Kozlowski, "Implementing a Common Architecture for EOD Robotic Systems," 2011 NDIA Ground Vehicle Systems Engineering and Technology Symposium (GVSETS), Robotic Systems

(RS) Mini-Symposium, Dearborn, MI, August 9-11, 2011.

[7] M. A. Hinton, M. J. Zeher, M. V. Kozlowski, M. S. Johannes, "Advanced Explosive Ordnance Disposal Robotic System (AEODRS): A Common Architecture Revolution," Johns Hopkins APL Technical Digest, Volume 30, Number 3, JHU/APL, Laurel, MD.

[8] P. Koch and S. Lacroix. Managing environment models in multi-robot teams. Proceedings of the International Conference on Intelligent Robots and Systems (IROS), Daejon, Korea, 2017.

[9] M. Tenorth and M. Beetz. Knowledge processing for autonomous robot control. AAAI Technical Report SS-12-02, 2012.

[10] D. B. Terry. Replicated data management for mobile computing. Morgan & Claypool, 2008.