Agile Assignment of Sensing Assets to Mission Tasks in a Coalition Context

Alun Preece, *Cardiff University*
Tim Norman, *University of Aberdeen*
Geeth de Mel, *IBM US / Army Research Laboratory*
Diego Pizzocaro, *Cardiff University*
Murat Sensoy, *Ozyegin University* / *University of Aberdeen*
Tien Pham, *Army Research Laboratory*

Abstract

A key problem in managing intelligence, surveillance and reconnaissance (ISR) operations in a coalition context is assigning available sensing assets – of which there are increasingly many – to mission tasks. High demands for information and relative scarcity of available assets implies that assignments must be made taking into account all possible ways of achieving an ISR task by different kinds of sensing. Moreover, the dynamic nature of most ISR situations means that asset assignment must be done in a highly agile manner. The problem is exacerbated in a coalition context because it is harder for users to have an overview of all suitable assets across multiple coalition partners. In this paper, we describe a knowledge-driven approach to ISR asset assignment using ontologies, allocation algorithms, and a service-oriented architecture. An illustration of the use of the system from a mobile device is presented.

Keywords: sensor assignment; sensor sharing; intelligence, surveillance, reconnaissance; ontology; coalition

Introduction

In a coalition context, making the most effective use of intelligence, surveillance, and reconnaissance (ISR) assets is a challenging problem[1]. There are typically multiple ways to achieve an ISR task using sensor-provided data. For example, the NIIRS (National Image Interpretability Rating Scales) framework characterises various kinds of ISR tasks that can be achieved using visual sensing data of different types (visible, radar, infrared and multispectral)[2]. An end-user, for example an ISR analyst, cannot be expected to have specialist sensing knowledge: they need to be able to state their information needs in terms of *what* they want (for example, tracking high value targets in an area) rather than *how* those needs may be satisfied by sensor data. Assets are owned by different coalition partners, who need to control how their assets are shared with other partners[1]. Therefore the problem of identifying suitable ISR assets is difficult, without a great deal of knowledge about sensing capabilities and availability of coalition assets. Because the situation evolves rapidly, the asset-provisioning infrastructure that supports ISR

operations must be agile, being responsive to changes in users' needs and the availability of relevant assets.

A solution to the coalition ISR asset-task assignment problem needs a common representation of tasks and assets, extensible to new kinds of task or asset. Tasks need to be expressed at a high level, in terms of what the user wants. Efficient mechanisms are needed for matching tasks to available assets, where all possible means of satisfying a task are considered. As part of a solution to this problem, several works have proposed the use of some form of knowledge base or mapping that relates sensor capabilities to task requirements, to aid either automatic or semi-automatic identification of suitable assets for tasks[3,4,5]. In our previous work[6] we developed an approach to automatic asset-task assignment founded on the Military Missions and Means Framework (MMF)[7]. We created ontologies of task and asset types, and an automatic procedure for matching one to the other, through the capabilities required on one and provided by the other. In this paper, we describe the current status of this approach and its implementation using a service-oriented architecture with mobile apps to serve users of the system.

An Ontological Approach to Asset-Task Matching

To derive our asset-task matching ontology, we formalized concepts and relationships from the MMF documentation, tailored to the ISR domain, as shown in Figure 1. *Missions* are comprised of *operations* which are in turn comprised of *tasks*. *Tasks* require *capabilities*, which are provided by *assets*. *Assets* include *platforms* and *systems*; *systems* – including *sensors* – are mounted on *platforms*. The relationship *allocatedTo* captures that an *asset* is assigned to resource a particular *task*. The ontology is implemented in the Web Ontology Language, OWL DL.

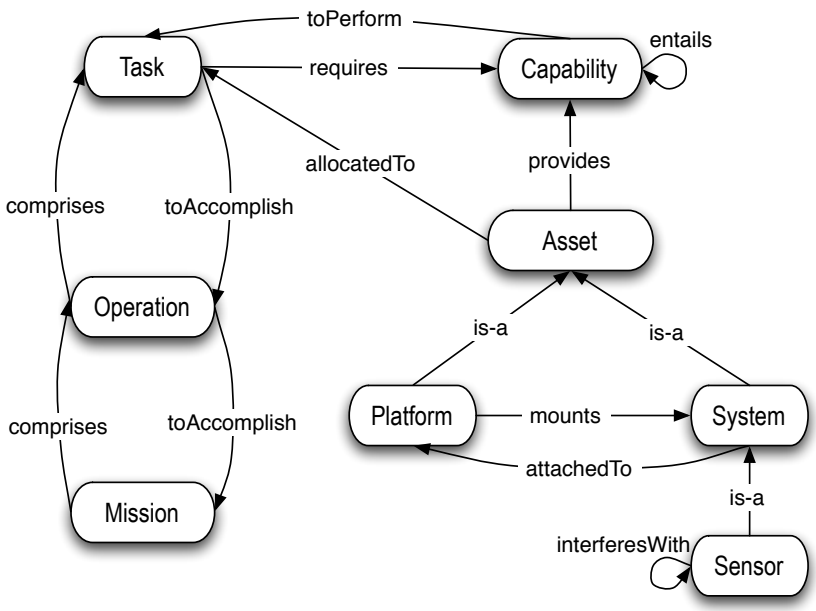

Figure 1: Missions and Means Framework ISR ontology

The current matching procedure using this ontology is based on the NIIRS framework. NIIRS associates various kinds of ISR tasks with ratings for the various kinds of visual sensing that can collect data sufficient to achieve the tasks. We formalized a collection of statements derived from NIIRS and related literature in the form of a knowledge base which, in abstract form, contains a set of *intelligence clause* tuples with six elements:

- an *intelligence capability* which is one of the three capabilities in NIIRS – *detect*, *distinguish*, *identify* – or *localize*;
- a set of *detectable things* drawn from the NIIRS framework (for example, kinds of vehicle or building);
- a set of more specific *features* of the detectable entities (for example, the roads or guard posts of a base, or the runways of an airport);
- a *context*, defining the preconditions that must hold for the intelligence clause to apply (for example, detection of a ship in the context of open water);
- the *type* of sensor-provided data which includes the NIIRS types – *visible, radar, infrared, multispectral* – plus other types including *acoustic* and *seismic*; and
- a *NIIRS rating* on a scale of 0 to 9 (for example, visible-6 or radar-4).

A user expresses their task in terms of required intelligence capability and detectable as above; for example, *detect {tank}* or *distinguish {tank, jeep}*. Required tasks feature either a single detectable (for detect, identify and localize tasks) or a pair of detectables (for distinguish tasks). The user also specifies the area of interest and other parameters shown later in the paper in the context of our mobile app.

The capabilities required by a task are determined by matching the task's intelligence capability and detectable to the KB of intelligence clauses. Two kinds of inference are

used here: firstly, some types of task imply others (for example, the ability to *identify* something implies also an ability to *detect* it); secondly, a hierarchy of detectables meant that, for example, any clause involving *detect* and a kind of detectable is considered to cover all more specialised kinds of that detectable also (so, for example, the task *detect* { *car* } covers specialised kinds of car: jeep, SUV, saloon, etc).

Sensors are allocated to tasks in terms of *bundles* (a task may require more than one sensor, for example a pair for 2D-localisation of an object); a *bundle type* is a combination of a platform type and a set of sensor types that can be mounted on that platform type. The ontology contains the additional relationship *interferesWith* to cover cases where types of sensor are incompatible. Bundle types are derived from *deployable configurations* as discussed in the next section, which also take into account restrictions on the number of sensors that can be mounted simultaneously on a platform. In terms of our MMF ontology, a bundle type *provides* a set of capabilities that is the union of the capabilities provided by either the platform type or a sensor type in the bundle type.

A bundle type *matches* a task type if the set of capabilities provided by the bundle type contains the set of capabilities required by the task type. We say that a bundle type *minimally matches* a task type when no sensor type can be removed from the bundle type such that the *matches* relationship still holds.

Using these definitions, the matching procedure operates as follows:

1. A user creates a task, from which the system derives the corresponding required intelligence capability and detectables.
2. The system determines all bundle types that minimally match the task.
3. The system determines all possible *bundle instances* that conform to each bundle type in the required area of interest. A bundle instance is a deployable, configured set of assets of the platform and sensor types defined by the bundle type. These are determined by querying the coalition's asset catalogue, governed by sharing and deployment policies as described later. For example, a UAV would not be selected for an area under a no-fly zone.
4. The system uses an allocation mechanism to choose the most appropriate bundle instance to assign to the user's task. Depending on the context of use, there will be different ways to do this; one method is to use a distributed allocation protocol[8] that attempts to maximise overall utility of assigned assets in the face of multiple competing tasks. The protocol aims at maximising the utility provided to the most important tasks, allowing for sensor assets to be reallocated to more important newly created tasks (we refer to this as *preemption*). For this, it is necessary to have an appropriate utility function to determine the "goodness" of assigning a particular bundle instance to a given task. We give an example in the sidebar "**End-to-end asset-task allocation and information delivery"**.

Our knowledge-based approach is intended to be extensible and maintainable. The core sensor, platform and task ontologies are based on pre-existing models, chosen for their relatively stable and accepted nature[6]. The ontologies are open to new asset, task, and capability types – in fact, the NIIRS framework was added as an extension to the original

ontologies. Some types of platform are not yet incorporated into our system, for example satellites, but they are covered by the ontologies on which our approach builds (e.g. Ontosensor[5]), so adding them would be straightforward. In general, we would expect new types of assets, tasks, and capabilities will appear only rarely. To make their assets available to our system, a new coalition partner needs to create representations of their asset instances in a shared asset catalogue; how this is done will depend on the underlying middleware: we give an example of an approach using a service-oriented architecture later in this paper. One area where future extension is planned is the inclusion of humans as sensors.

Sidebar: End-to-end asset-task allocation and information delivery

Figure 2 shows an overview of our approach, with an example. The process operates clockwise from the top left of the figure. The user specifies a task to *localize SUVs* in an area of interest. The matching procedure uses the MMF ISR ontology and knowledge base clauses to determine that this task can be achieved by assets with a *Visible NIIRS rating of 4* (or higher) or an *Acoustic NIIRS rating of 6* (or higher). These ratings are provided by the following bundle types respectively: a *UAV* platform with *video camera* sensor, or an *unattended ground system* platform with an *acoustic array* sensor. (This example is simplified: many more possibilities and more specific kinds of asset actually exist for this task.) Bundle instances are now generated and ranked with a utility function for the localization task. A bundle instance may contain more than one instantiation of the bundle type, where more than one set of deployed assets is needed to achieve the task. In our example, this results in the generation of three pairs of acoustic sensing bundles, each containing two instances of the bundle type <*UGS,AcousticArray*> (at least two assets are required for triangulation). For the bundle type <*UAV,VideoCamera*>, the bundle generation process results in two visual sensing bundle instances composed of a single UAV mounting a camera. In total, there are five candidate bundles. As a result of the ranking process, one of the acoustic bundles is chosen and assigned to the task. We envisage our approach being deployed in the context of a service-oriented architecture, where sensors and processing services are "wrapped" as Web services, allowing highly agile configuration of services to deliver information to a user – potentially to a mobile device if the user is in the field – once assets are assigned.

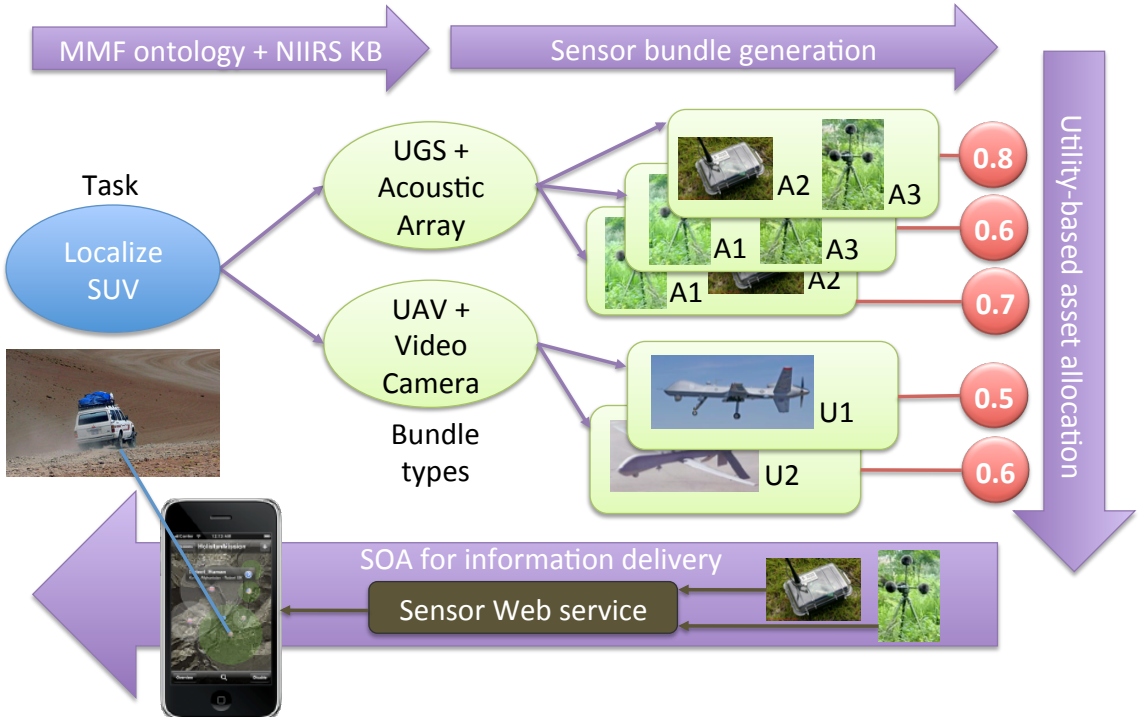

Figure 2: Overview of the our ISR asset-task assignment approach

---

Extensible Matching Framework

To be useful in a coalition context, a asset-task assignment system needs to be extensible and flexible, not only in terms of new ontological and KB elements (for example, to incorporate new kinds of sensing assets and alternative formulations of ISR tasks), but also in terms of extended matching schemes (including alternative allocation procedures). This extensibility and flexibility is enabled by Ontological Logic Programming (OLP)[9], which combines Prolog with DL-based reasoning. An OLP program can dynamically import various ontologies and use the terms (i.e., classes, properties, and individuals) in these ontologies directly within an OLP program. The interpretation of the ontological terms is delegated to an OWL DL reasoner during interpretation of the OLP program, supporting operations including subsumption, satisfiability, consistency, class equivalence, and class instance checking.

A fragment of our OLP implementation to compute deployable configurations (for use as bundle types) is shown in Figure 3. The OLP program is a Prolog program, where concepts and properties from the underlying ontologies are referenced directly. The MMF ISR ontology (http://homepages.abdn.ac.uk/c.emele/pages/ita/index.php?page=resources) is imported on the first line. The *getConfigurations* predicate computes deployable configurations (bundle types) for a specific task. Each sensor must be carried by a

deployable platform that provides all of the operational requirements of the task (e.g., constant surveillance). If a sensor cannot be carried by a deployable platform, there is no point in considering deployable configurations with that sensor type. Using this knowledge, a tailored and efficient matchmaker can be employed. This matchmaker first identifies the deployable platforms that meet the requirements of the task. Once many possibilities are narrowed down by determining deployable platforms, the sensor types that provide the intelligence capabilities required by the task are determined incrementally so that those sensors can be mounted on the deployable platforms. In the code, terms from the MMF ISR ontology have the prefix "istar:". Most of these are shown in Figure 1 (*requireOperationalCapability* can be considered a specialization of *requiresCapability*).

```
%import http://…/ISTAR.owl
getConfigurations(T,[P|S]):-
    deployablePlatform(T,P),
    extendSolution(T,P,[],S).
deployablePlatform(T,P):-
    istar:`Platform'(P),
    not((istar:`requireOperationalCapability'(T,C),
         not(istar:`provideCapability'(P,C)))).
extendSolution(T,P,Prev,Next):-
    requireSensor(T,P,Prev,X),
    istar:`mounts'(P,X),
    A=[X|Prev],
    extendSolution(T,P,A,Next).
extendSolution(T,P,S,S):-
    not(requireCapability(T,P,S,_)).
requireSensor(T,P,S,X):-
    requireCapability(T,P,S,C),
    istar:`provideCapability'(X,C).
requireCapability(T,P,S,C):-
    istar:`requireCapability'(T,C),
    not(provideCapability(S,C)),
    not(provideCapability([P],C)).
provideCapability([Y|Tail],C):-
    istar:`provideCapability'(Y,C),!;
    provideCapability(Tail,C).
```

Figure 3: OLP program to compute deployable configurations

This method for computing deployable configurations is based on the idea that the search space can be significantly reduced using domain knowledge. For this purpose, dependencies between sensors and platforms are exploited. Using this principle, at each iteration, the matchmaking algorithm rules out many combinations and significantly reduces the time required to compute deployable configurations. We compared the computational performance of the OLP program in Figure 3 with a set-covering algorithm that exhaustively searches for deployable configurations. As the size of deployable configurations increases, our experiments have shown the OLP-based approach outperforms the exhaustive search approach significantly: time consumption of

the exhaustive search increases exponentially while that of the proposed approach is linear[9].

Asset-Task Matching in a Service-Oriented Architecture

To allow our approach to operate within a service-oriented architecture for sensor information processing and delivery, we have deployed the asset-task matching mechanism as a service within the ITA Fabric[10]. The Fabric provides a distributed stream-oriented middleware layer that mitigates the complexity of managing message flows between coalition partners and across disparate networks, while also encapsulating network and security management for resource-constrained networks. The Fabric implements a two-way messaging bus and a set of middleware services providing connectivity between all network resources, to each other and to users. A typical Fabric node consists of a message broker, an instance of the Fabric Manager, and an instance of the Fabric Registry. The Fabric Manager manages all the communication channels between nodes, the routing of messages between nodes, sensing resources and users. Information about all nodes, routes and ISR resources is recorded in the Fabric Registry: a database distributed across each of the Fabric nodes, recording resource types, physical locations, operational characteristics, task commitments, and current operational status/availability. The Registry implements the asset catalogue – one of the immediate benefits of embedding our approach in the Fabric is the matching process can use the Fabric Registry to limit the generation of bundle type to only those deployable configurations where asset instances are available for assignment.

Integration of our approach in the ITA Fabric is shown in Figure 4. **R1**…**R5** represent sensing resources, and **N1**...**N4** are fabric nodes. Because Fabric nodes are intended to be highly lightweight, deployable on low-powered sensor platforms, it is infeasible to run the computationally-expensive DL-based reasoning operations on a Fabric node. Instead, the functionalities of our approach are divided into two separate components: the OLP implementation of the reasoning procedure is deployed on a server, accessible via an API defined a set of RESTful Web services, separate from the user-facing application that allows users to submit tasks and receive recommendations. We refer to the reasoning API as RaaS: Reasoning-as-a-Service.

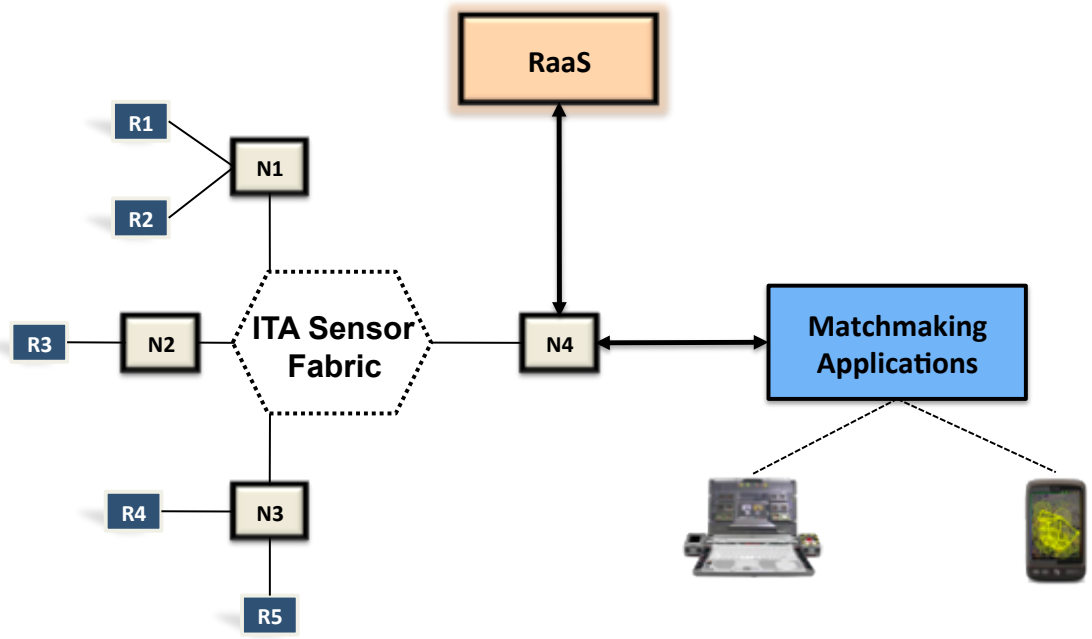

Figure 4: Integration of our approach in the ITA Fabric

Asset-Task Matching via a Mobile Device

As a concept illustration of the use of our approach, we have created mobile apps for both smartphone and tablet platforms. The most recent version is implemented as an iPad app. The main features of this app are:

- Allow a user to create an ISR task in an area-of-interest, by means of a convenient user interface, and submit the task for asset assignment.
- Allow a user to view all tasks with assigned assets in an area of interest (subject to access policies).
- Allow the sharing of tasks among users (again, subject to access policies).

The motivation for task sharing was to reduce competition among tasks for resources, by making it relatively easy for a user to share an existing task rather than to create a new request for resources. The app is intended to give a user an overview of how well-covered an area is in ISR terms, not just in terms of what tasks are currently being resourced, but also the likely permanence of these tasks (by allowing the user to view details of the tasks such as their ownership and priority). This is very different from simply showing the current location of sensor assets deployed on the field; in fact, displaying such information may be unfeasible due to visibility policies; for example, a coalition member might not be allowed to see other partners' sensor locations or exact resource types, though they might have the rights to access the data collected. Moreover, displaying locations of sensors on the field could also be misleading for a commander, who might plan a mission in a certain area because "better covered" by sensors, while instead those sensors might be busy or preempted to serve other more important tasks.

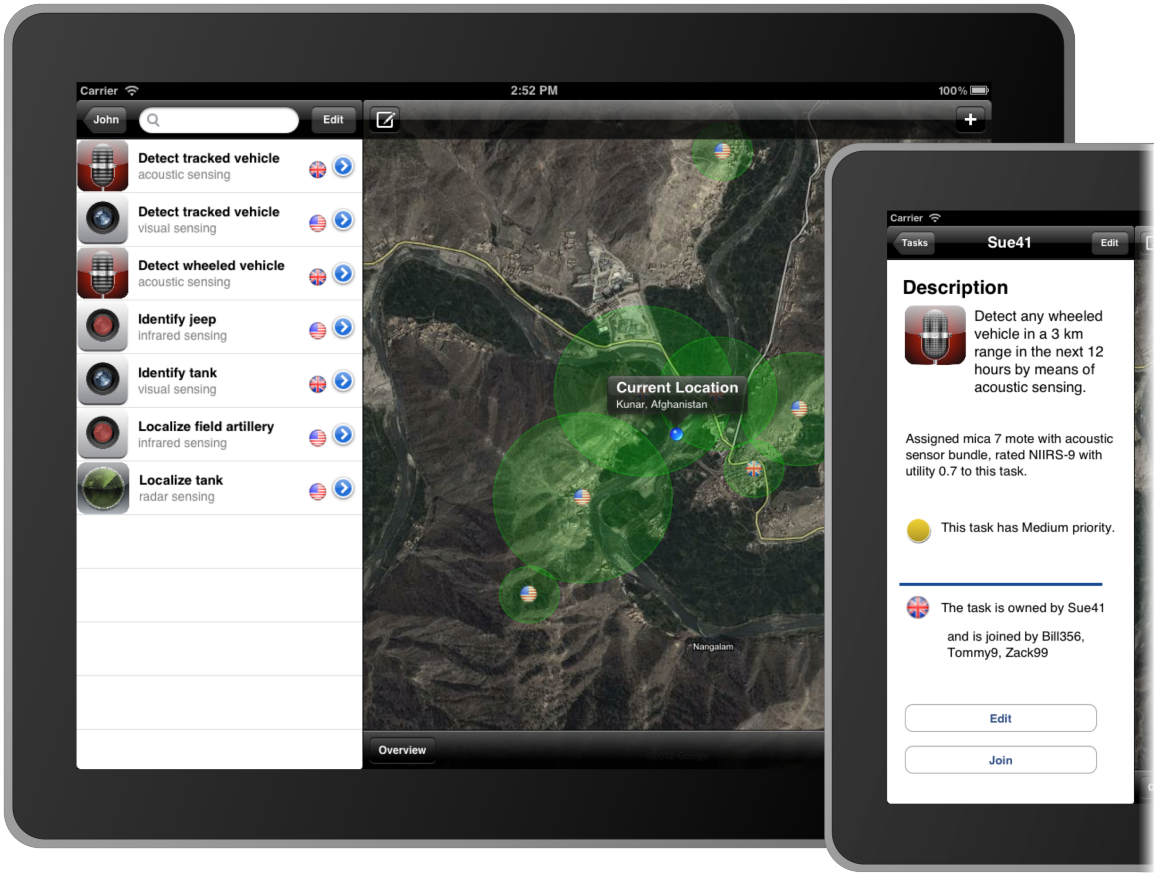

Figure 5: Sensor Assignment to Missions iPad app

Figure 5 shows two screens from the iPad app: the task list is on the left, and the locations of tasks are shown as circular regions on the map. The list of tasks is obtained by querying the Registry on a local Fabric node, and contextualised by the area of interest, which is by default determined by the iPad's location (via GPS) but can be set manually also. After defining the radius of the area of interest of the task, the user is then allowed to move the task to the desired location on the map by dragging the center of the area of interest defined. Defining areas as circular is a limitation, but it would be straightforward to implement more complex shapes. The interface also allows the user to set the task type, task priority and expected duration of the task. The user interface is designed for a logged-in user who belongs to a single coalition partner, and is able to see a list of tasks (and their details) belonging to other coalition partners according to pre-defined access policies. Access policies[1] are rule-based, and can take account of factors including the user's coalition partner membership, their rank, membership of a particular group within the coalition, and also the partner ownership, rank, and group associated with the task.

The task list is searchable using the box on the top-left allowing users to filter the displayed list by intelligence capability or type of detectable thing. More detail can be obtained by selecting an individual task, which results in display like the one shown on

the right of the figure. Assuming the task has assigned sensors, the display presents a summary of the task and the assigned asset bundle. With this amount of detail, the user can understand how the task is currently being resourced. Inclusion of the NIIRS rating is intended to give an indication of the quality of the data the asset bundle can collect. We are considering other ways to convey quality information, as it may not be reasonable to assume users are familiar with the NIIRS scale. The task priority is an indication of how "stable" the task is, as lower priority tasks are more prone to being pre-empted if assets are too scarce to cover all tasks. Information on the task owner and other users who have joined the task is intended to have a "social" effect, as the logged-in user is likely to know other users in the region.

Based on this detailed task assignment information, the user may use the buttons on the bottom left to choose to join or edit the existing task. This entails a more efficient use of network resources: avoiding the creation of a new task reduces the competition among them for sensing resources (where it might not be possible to support every task in the case of limited resources). In some cases, a user's information requirements may be satisfiable by pre-existing tasks which are already being served by allocated bundles and therefore there is no need to preempt resources from other tasks.

The iPad app was demonstrated to ISR specialists from the US Department of Defense, UK Ministry of Defence, and NATO communities in September and October 2011. Users were appreciative of how the app achieves separation between what information the user requires, and how this information can be obtained by various kinds of sensing asset. The demos also elicited positive comments regarding the usability and simplicity of the mobile version in terms of how one can request complex information through a simple task creation form. Moreover it was appreciated that low-level details of both sensors and tasks (e.g. particular sensor capabilities required to satisfy a task) can remain hidden to a user therefore making the system accessible also to non-expert users.

Conclusion & Future Work

In this paper, we described a knowledge-driven approach to ISR asset assignment using ontologies, allocation algorithms, and a service-oriented architecture, accessible from mobile devices. Going forward, we are experimenting with richer ways of interacting with users by means of natural language interfaces, and using the approach to assign pre-collected information sources and streams in addition to sensing assets. Other related ongoing work addresses consideration of network (e.g. bandwidth) constraints in allocating asset bundles, and trust and information obfuscation issues in relation to ISR asset sharing among members of heterogeneous coalition teams.

Acknowledgment: This research was sponsored by the US Army Research Laboratory and the UK Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either

References

1. T. Pham, G. Cirincione, D. Verma, and G. Pearson, “Intelligence, Surveillance, and Reconnaisance Fusion for Coalition Operations,” in *Proc 11th Int Conf on Information Fusion*, 2008.
2. J.M. Irvine, “National Imagery Interpretability Rating Scales (NIIRS),” in *Encyclopedia of Optical Engineering*, Marcel Dekker, 2003, pp. 1442–1456.
3. L. Lefort, C. Henson, and K. Taylor, *Semantic Sensor Network XG Final Report*, W3C, 2011; http://www.w3.org/2005/Incubator/ssn/XGR-ssn-20110628/.
4. T. Mullen, V. Avasarala, and D. L. Hall, “Customer-Driven Sensor Management,” *IEEE Intelligent Systems*, vol. 21, no. 2, Mar/April 2006, pp. 41–49.
5. D. Russomanno, C. Kothari, and O. Thomas, “Building a Sensor Ontology: A Practical Approach Leveraging ISO and OGC Models,” in *Proc Int Conf on Artificial Intelligence*, 2005, pp. 637–643.
6. M. Gomez, A. Preece, M. Johnson, G. de Mel, W. Vasconcelos, C. Gibson, A. Bar-Noy, K. Borowiecki, T. La Porta, D. Pizzocaro, H. Rowaihy, G. Pearson, and T.~Pham, “An Ontology-Centric Approach to Sensor-Mission Assignment,” in *Proc 16th Int Conf on Knowledge Engineering and Knowledge Management (EKAW*), 2008, pp. 347–363.
7. J.H. Sheehan, P.H. Deitz, B.E. Bray, B.A. Harris, and A.B.H. Wong, “The Military Missions and Means Framework,” in *Proc Interservice/Industry Training and Simulation and Education Conference*, 2003, pp. 655–663.
8. D. Pizzocaro, A. Preece, F. Chen, T. La Porta, and A. Bar-Noy, “A Distributed Architecture for Heterogeneous Multi Sensor-Task Allocation,” in *Proc 7th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS),* 2011.
9. M. Sensoy, G. de Mel, W. Vasconcelos, and T.J. Norman, “Ontological Logic Programming,” in *Proc Int Conf on Web Intelligence, Mining and Semantics (WIMS),* 2011.
10. J. Wright, C. Gibson, F. Bergamaschi, K. Marcus, R.Pressley, G.Verma, and G.Whipps, “A Dynamic Infrastructure for Interconnecting Disparate ISR/ISTAR Assets (the ITA Sensor Fabric)," in *Proc 12th Int Conf on Information Fusion*, 2009.