Matching Sensors to Missions using a Knowledge-Based Approach

Alun Preece[a], Mario Gomez[b], Geeth de Mel[b], Wamberto Vasconcelos[b], Derek Sleeman[b], Stuart Colley[c], Gavin Pearson[c], Tien Pham[d] and Tom La Porta[e]

[a]School of Computer Science, Cardiff University, UK
[b]Computing Science Department, University of Aberdeen, UK
[c]Defence Science and Technology Laboratory (Dstl), UK
[d]Army Research Laboratory (ARL), USA
[e]Department of Computer Science and Engineering, Pennsylvania State University, USA

ABSTRACT

Making decisions on how best to utilise limited intelligence, surveillance and reconnaisance (ISR) resources is a key issue in mission planning. This requires judgements about which kinds of available sensors are more or less appropriate for specific ISR tasks in a mission. A methodological approach to addressing this kind of decision problem in the military context is the Missions and Means Framework (MMF), which provides a structured way to analyse a mission in terms of tasks, and assess the effectiveness of various means for accomplishing those tasks. Moreover, the problem can be defined as knowledge-based matchmaking: matching the ISR requirements of tasks to the ISR-providing capabilities of available sensors. In this paper we show how the MMF can be represented formally as an ontology (that is, a specification of a conceptualisation); we also represent knowledge about ISR requirements and sensors, and then use automated reasoning to solve the matchmaking problem. We adopt the Semantic Web approach and the Web Ontology Language (OWL), allowing us to import elements of existing sensor knowledge bases. Our core ontologies use the description logic subset of OWL, providing efficient reasoning. We describe a prototype tool as a proof-of-concept for our approach. We discuss the various kinds of possible sensor-mission matches, both exact and inexact, and how the tool helps mission planners consider alternative choices of sensors.

Keywords: sensor assignment; mission planning; intelligence, surveillance, reconnaissance; ontologies; reasoning

1. INTRODUCTION

Making the most effective use of limited intelligence, surveillance, and reconnaissance (ISR) resources is seen as a key problem in modern, network-centric coalition operations. The problem has been stated in the following terms:[1] "ISR resources are typically in high demand and requirements usually exceed platform capabilities and inventory. ... The foremost challenge of collection management is to maximize the effectiveness of limited collection resources within the time constraints imposed by operational requirements." The problem is exacerbated in the coalition context because the full inventory of ISR assets potentially available — including platforms such as unmanned aerial and ground vehicles, as well as the various types of sensor available for deployment on the platforms — is not easy to obtain at-a-glance, and so a commander engaged in mission planning will not have ready access to the full range of options. Moreover, the operational environment is highly dynamic: ISR requirements change in response to the emerging situation, and the availability of assets needs constant updating due to factors such as technical failures, vulnerabilities and, more positively, provision of newly-available solutions.

In this context, we identify the sub-problem of *sensor-mission assignment* as that of allocating a collection of ISR assets (including sensors and sensor platforms) to one or more missions in an attempt to satisfy the ISR needs of those missions. During the process of mission planning, a commander will identify a number of tasks that require ISR capabilities.[2] For example, the task of locating snipers in woodland may be specified as

Send correspondence to A.Preece, E-mail: A.D.Preece@cs.cf.ac.uk, Tel: +44 (0) 29 2087 4653

requiring day/night visual surveillance. Logistically, the coalition in which the commander is operating will have a set of available ISR assets (platforms and sensors), characterised in terms of their types (for example UAVs and UGVs), locations, readiness status, and so forth. One element of the sensor-mission assignment problem is to match available assets to tasks, to give the commander an at-a-glance view of potential solutions.

In this paper we propose a solution to this element of the sensor-mission assignment problem, which follows a knowledge-based systems approach. Our goal is to provide mission commanders with a tool that allows them to consider all options at their disposal to meet their ISR needs. We view this sensor-mission matching tool as being usable in a highly dynamic mission planning context, both at the early stages of assigning assets to tasks, and also during re-planning to cope with the emerging situation and contingencies. Our solution is designed to serve as input to the subsequent stages of resource allocation, sensor and platform deployment and tasking, and operational monitoring of the ISR network. Both the ongoing monitoring, and the appearance of new tasks and ISR requirements, can force the commander to reassess the sensor-mission assignment solution. This whole process is illustrated in Figure 1.

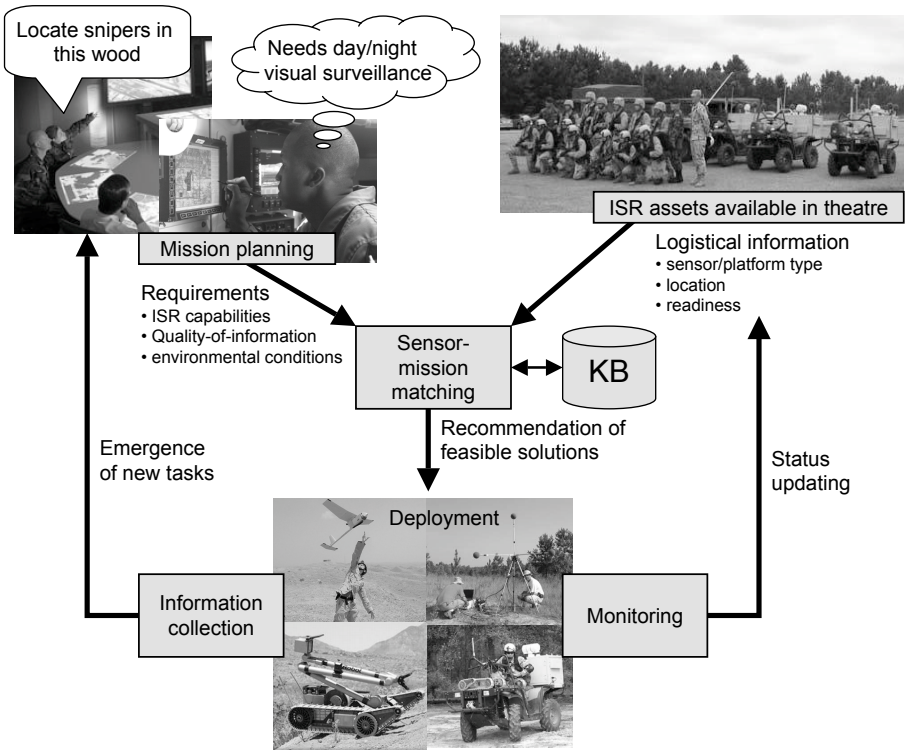

Figure 1. Overall view of the sensor-mission assignment process

Our knowledge base for the sensor-mission matching problem is structured as a set of interrelated ontologies.[3] We adhere to the Semantic Web architecture[4] and employ the Web Ontology Language (OWL)* so as to get maximum benefit from off-the-shelf tools (for system development and for reasoning over the knowledge base), and also to be able to exploit pre-existing ontologies where they exist for parts of our problem. In the next section, we introduce the fundamental structures in an ontology, using an example to motivate their use in sensor-mission matching.

2. ONTOLOGIES FOR MATCHMAKING

Since the early 1990s, it has been widely agreed that knowledge bases are generally founded on ontologies, where widely-cited definitions of *ontology* in this context are: "a specification of a conceptualization"[3] and "a set of

*http://www.w3.org/TR/owl-features/

logical axioms designed to account for the intended meaning of a vocabulary".[5] So, while a knowledge base will often contain statements such as rules, facts, and constraints, the ontology defines formally the semantics of the terms used in these statements.[4] This modern view of ontologies arose largely to address the problem of knowledge sharing and reuse: if two different knowledge bases use the same underlying ontology, then it should be possible to combine the knowledge from both in a meaningful way. In our domain, for example, a knowledge base covering sensors, sensor platforms, and mission tasks would in principle be able to reuse pre-existing knowledge on each of these, provided that common ontologies are used. As noted above, the Semantic Web approach offers an architecture and suite of ontology representation languages aimed at supporting information sharing and interoperability in the Web context. This has resulted in the provision of a range of tools for building and processing Semantic Web ontologies, and the availability of a large number of reusable ontologies[†].

An ontology is typically structured as a set of definitions of *concepts* (also known as *classes*) and *relations* between those concepts (also known as *properties*). A fundamental pre-defined property is the *subclass* relation, denoting that one concept is a specialisation of another. For example, the concept "unmanned aerial vehicle" is a subclass of "aircraft", which in turn is a subclass of "vehicle". Ontologies are expressed using some meta-ontology language, the most commonly-used of which is now the Web Ontology Language (OWL). Strictly speaking, OWL is a suite of interrelated languages, with various levels of expressive power.[4] OWL provides ontology developers with the apparatus to define new concepts and properties based on existing ones; OWL-supporting tools allow ontologies to be checked for consistency and other forms of well-formedness, and a number of reasoning engines exist to process OWL ontologies to draw inferences and answer queries.

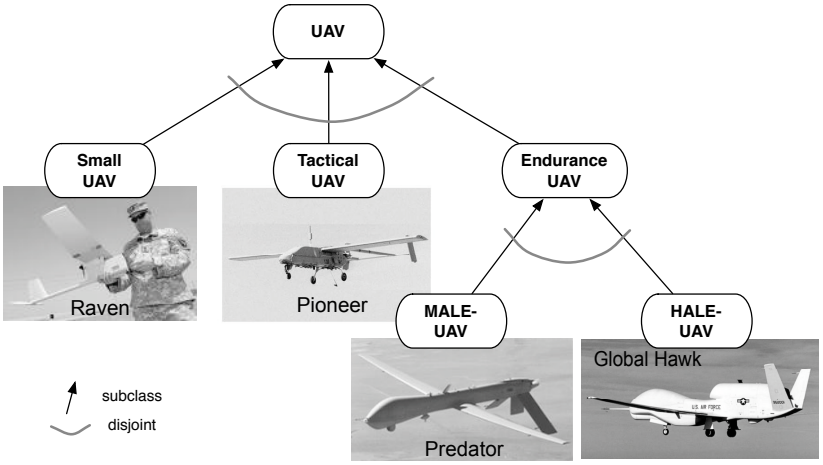

Figure 2. Fragment of an ontology for unmanned aerial vehicles (UAVs)

As a simple motivating example for the use of ontologies in the sensor-mission matching problem, Figure 2 depicts a fragment of an ontology for unmanned aerial vehicles (UAVs). Six concepts are shown, together with subclass relationships between them.[‡] The most general class is UAV, which is subclassed into: Small UAV, designed to perform "over-the-hill" and "around-the-corner" reconnaissance; Tactical UAV, which focuses on the close battle, providing targeting, situation development and battle damage assessment; and Endurance UAV, aimed at the deep battle, typically ranging from 150 to 300 kilometres. The Endurance UAV class is specialised into MALE-UAV (Medium Altitude Long Endurance UAV) and HALE-UAV (High Altitude Long Endurance UAV); the former is designed to operate at altitudes between 5000 and 25000 feet, while the latter are designed to function as low earth orbit satellites. The arcs between subclass relationships indicate a disjoint relationship among subclasses; a disjoint relation among a set of classes entails that an individual cannot belong to more

[†]For example, see http://swoogle.umbc.edu/

[‡]We have kept this example simple for illustrative purposes; a realistic ontology of UAVs will have many more classes and more complex relationships between them.[6]

than one of those classes; for example, a UAV that is classified as a Small UAV cannot be classified as being a Tactical UAV.

Based on this simple ontology fragment, we can illustrate some basic examples of reasoning. Let us suppose that we have the following instances of UAVs available for a mission: (1) a Pioneer, which is a Tactical UAV; (2) a Predator, which is a MALE-UAV; and (3) a Global Hawk, which is a HALE-UAV. Now suppose that a mission commander needs to carry out a task involving the ISR requirement Constant Surveillance over a wide area, in order to detect any suspicious movement. This kind of ISR requirement is best met by an Endurance UAV, since it is able to fly for long periods of time. From just the concept definitions we know that: (1) the Pioneer is not an Endurance UAV (because of the disjoint relationship among the *Endurance UAV* and Tactical UAV classes), and (2) both the Predator and the Global Hawk are Endurance UAVs (because of the subclass relationships). Note that we only state minimum explicit information about the UAVs (e.g. Pioneer is a Tactical-UAV); everything else is inferred from the concept definitions (e.g. the Pioneer is *not* a HALE-UAV). In a sensor-mission matchmaking process using this ontology, a reasoning engine could select both the Predator and the Global Hawk as platform assets satisfying the specified mission requirements.

Taking this example further, suppose that according to the weather forecast, storms are very likely to occur in the area of operations during the surveillance period. High Altitude Long Endurance UAVs have the capability of flying "above the weather", so the best option would now be to recommend the Global Hawk as the only asset satisfying the ISR requirements, factoring-in the environmental conditions.

Ontology designers using OWL are able to define their own descriptive properties, and use these to define both concepts and individual instances. OWL allows the definition of two kinds of property: a *datatype property* relates an individual to a data literal, and an *object property* relates two individuals. Normally, the ontology includes definitions stating that all instances of a particular class have some restriction on their properties and values. For example, we can define an object property providesCapability that relates an individual sensor or platform to an ISR capability, and then define all instances of Endurance UAV such that they have the providesCapability property with value Constant Surveillance. Datatype properties are useful for assigning values to properties that are integers, real numbers, or strings (but not instances of a defined class). We could define properties range and speed for all vehicles (including all instances of UAV) and then restrict instances of a particular model of UAV to specific values for these. For example, a PredatorB UAV could be defined as having range 5500km and speed 400kph. Recent versions of OWL allow definition of concepts in terms of restrictions on values of datatype properties. So, for example, an Endurance UAV could be defined as one that has a range greater than some particular value.

As already noted, an advantage of using the Web Ontology Language is that there exists a reasonably well-developed toolset for OWL, including editors such as Protégé and Swoop and reasoners such as Pellet[§]. Using an editor linked to a reasoner allows a developer to build an ontology (possibly by importing existing ontologies and extending them) and check its consistency in an incremental way. Once the ontology has stabilised, the reasoner can then be used to answer queries using the ontology. In our domain, such queries involve matching platforms and sensors to mission requirements, as described in the following section.

3. FRAMEWORK FOR SENSOR-MISSION MATCHING

As introduced above, the main elements in our sensor-mission matching problem are the mission tasks, the ISR requirements of these, the available assets (sensor and platforms) and the ISR-providing capabilities of the assets. A well-known structured approach to mission planning used by the US military is the Missions and Means Framework (MMF).[7] MMF provides a model for explicitly specifying a military mission and quantitatively evaluating the mission utility of alternative warfighting solutions (the *means*). While there has been work in formalising MMF using set theory,[8] ours is the first work to define MMF as an ontological framework.

Figure 3 shows a high level picture of how missions map to ISR-priviging assets, based on MMF. Starting from the top left the diagram breaks a mission down into a collection of *operations* (e.g. search-and-rescue), each

[§]A list of Semantic Web tools, including OWL editors and reasoners, is maintained by the W3C at http://esw.w3.org/topic/SemanticWebTools

of which is broken down further into a collection of distinct *tasks*. Each task has specific capability requirements (e.g. wide-area surveillance). On the right hand side, the diagram shows the analysis of capabilities as a bottom-up process that builds from elementary components (e.g. EO/IR camera) into systems (e.g. camera turret), and from systems up into platforms equipped with or carrying those systems (e.g. a UAV).

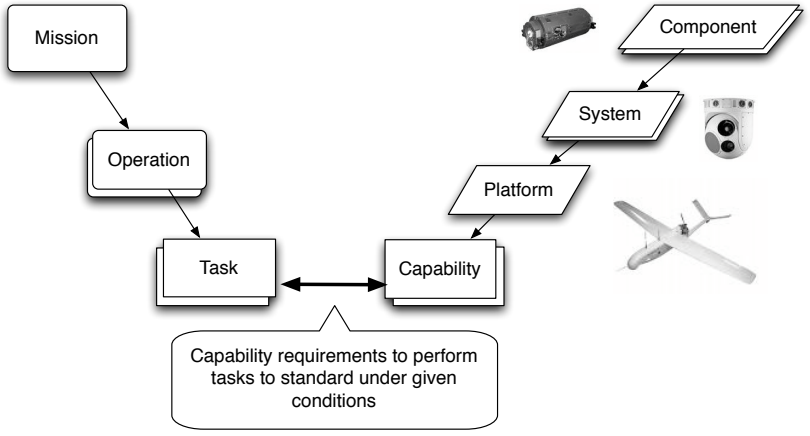

Figure 3. Mission and Means Framework

MMF thus provides us with an existing framework into which we can fit a solution to the sensor-mission matching problem, and which has the highly desirable property of already being familiar and proven in the military context. It is worth noting that, while MMF is not the only such approach used in the military domain, others such as *effects-based planning*, share many common features with MMF. It should therefore be possible to reuse much of a solution structured around MMF in the context of an alternative mission planning approach.

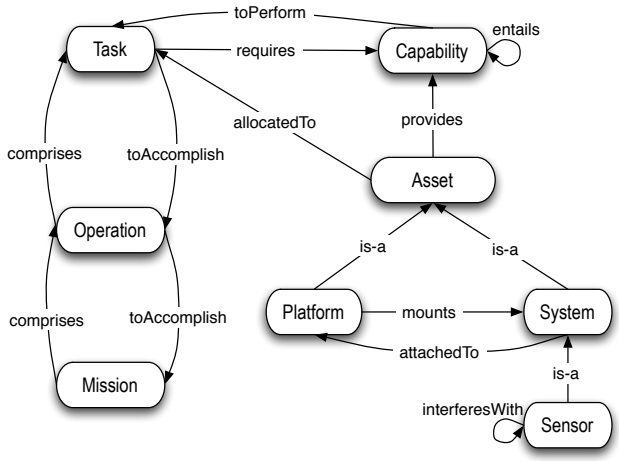

Figure 4. Main concepts and relations in the MMF ontology

Figure 4 sketches the main concepts of our ontology based on MMF. On the left hand side, we have the concepts related to the mission: a Mission comprises one or more Operations to be carried out, and each operation breaks down into a number of Tasks that must be accomplished. The important feature of a Task is that it is defined as requiring some capabilities; in our domain, we focus on ISR capabilities, but the approach generalises to other forms of capability too.[8] We define the OWL object property requires to relate an individual Task to the individual Capability instances that it requires. Because the definition of mission, operation and task is somewhat

subjective in practice, we adopt a simple model of hierarchical trees of tasks, where:

- a Task can be broken down into sub-tasks (which are also Tasks);
- a Mission is a Task with no super-task (it is the root of a tree of tasks);
- an Operation is a Task whose super-task is a Mission (operations comprise the second level of a tree of tasks, imediately below the root).

These definitions allow us greater flexibility in specifying the required capabilities of a mission, catering for relatively simple specifications where, for example, we associate collections of required capabilities at a high level with operations or missions, and also covering very detailed specifications down to fine-rained tasks, sub-tasks, sub-sub-tasks, etc, each with specific capability requirements. We thus hope to be broadly compatible with a range of mission-planning approaches. The set of capabilities required by a task T (at any level) is the aggregation of all capabilities required by T itself or by any subclass of T:

$$\mathsf{requires}(T', C) \wedge \mathsf{subTaskOf}(T', T) \rightarrow \mathsf{requires}(T, C)$$

where subTaskOf is a transitive object property.

On the right hand side of Figure 4 we have concepts related to capability-provision ("means" in MMF terms), with a focus on the sensor-mission matching problem: Platform and Sensor are two kinds of Asset; a Sensor is a kind of System that can be attached to a Platform; inversely, a Platform can mount one or more Systems. Some sensors can interfere with other sensors, so they cannot be used simultaneously. Assets provide capabilities, which provides the link to Tasks as discussed above. Moreover, a Capability can entail a number of more elementary capabilities[¶]. At some point, assets will be allocated to specific tasks that require the capabilities provided by them: the object property allocatedTo allows us to construct solutions to the sensor-mission assignment problem.

To summarise, the MMF ontology defines the concepts used to underpin our sensor-mission matching framework, in terms of relatively high-level classes (most importantly Task, Capability, Platform and Sensor) and relationships between them (most importantly requires, provides, and mounts). In addition, we need ontologies to describe more specialised concepts (e.g. types of capability, types of sensor or platform) and relationships (e.g. kinds of provision, including sensor provisioning and platform provisioning). We seek to build upon pre-existing ontologies wherever possible in these areas, as described in the next section.

4. ONTOLOGIES FOR INTELLIGENCE, SURVEILLANCE AND RECONNAISSANCE (ISR)

A wealth of work already exists in providing ontologies and ontology-like schemas in domains relevant to our sensor-mission matching problem. For example, sensors and sensor platforms are addressed in SensorML,[9] OntoSensor,[10] CIMA,[11] and the MMI Platforms ontology.[6] Mission tasks are defined in detail in military doctrine, most prominently the US military Universal Joint Task List,[12] which has been formalised as an ontology using the DAML representation, a forerunner of OWL[‖]. While many of the concepts and relationships in these ontologies can be imported into our framework (as specialisations of our MMF Sensor, Platform, and Task classes respectively), what is missing are the definitions of various kinds of Capability needed to match tasks to assets.

Looking at the range of possible capabilities, one sees that there are natural relationships between, on the one hand, ISR capabilities provided to tasks by sensors and, on the other hand, ISR capabilities provided to tasks by platforms. As examples of the former, different sensing modalities — optical, acoustic, infrared, etc — relate to various forms of intelligence a commander may require — Imagery Intelligence (IMINT), Acoustic Intelligence (ACINT), Infrared Intelligence (IRINT), etc. As examples of the latter, some platforms by nature of their mobility and/or range are suitable or unsuitable for wide-area surveillance; suitability for land, sea or air ISR operations, and in different kinds of weather also have a big impact on choice of platform. Moreover, there are multiple capability dimensions for both sensors and platforms, for example:

[¶]Primitive capabilities are called *functions* in MMF, but we have simplified this.

[‖]http://orlando.drc.com/semanticweb/daml/ontology/condition/ujtl/condition-ont

- for platforms: mobility, realm (air, land, sea), performance (range, endurance, altitude, speed, ...), application or mission type (surveillance, reconnaissance, target acquisition, ...), firepower, landing and takeoff, communications, vulnerability and survivability, availability, ...;
- for sensors: phenomena detected (type and spectrum), performance (resolution, sample rate, ...), vulnerability, interferences with other sensors, weather/terrain/contamination influence, ...

Therefore, it was clear to us that we needed to capture these various dimensions through a set of concept taxonomies, in such a way that we could characterise assets in terms of multiple concepts from an arbitrary number of the taxonomies. Figure 5 shows some of the taxonomies we have developed for our ISR ontology. The platforms (a) and sensors (b) taxonomies are used to characterize ISR assets, where a specific kind of asset may be a subclass of multiple classes in different branches of the tree. For example, PredatorB is asserted to be a subclass of MALE-UAV (Medium Altitude Long Endurance UAV), but it is also classified as a subclass of CombatUAV. ISR tasks (c) represent the main requirements used to select a type of platform, while intelligence disciplines (d) are used to select sensor types supporting the production of specific types of intelligence (e.g. optical sensors support the production of IMINT).

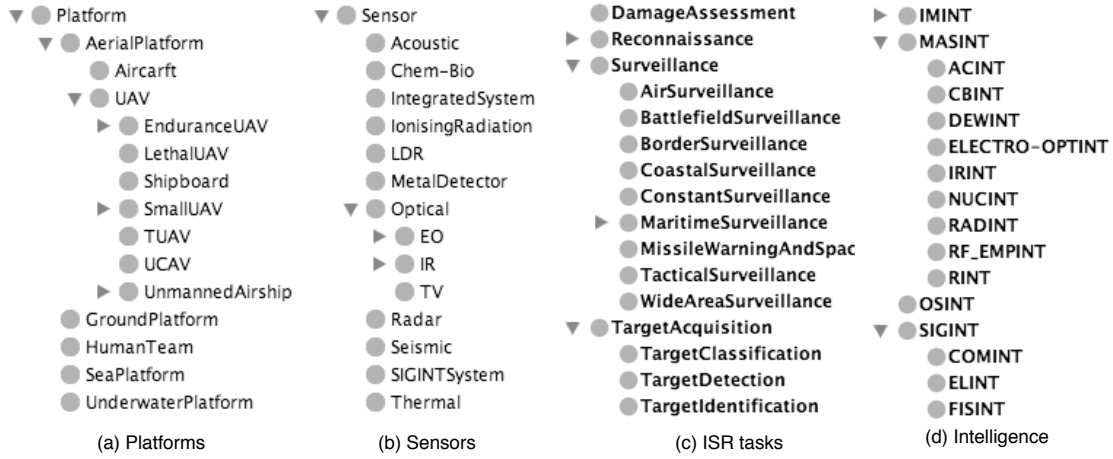

Figure 5. Some concept taxonomies representing the ISR domain

Using the description logic sublanguage of OWL, OWL DL[13] we have defined the taxonomies so that classification under a particular superclass entails the provision of certain types of Capability. In other words, the definitions restrict the kinds of value that the provides property shown in Figure 4 can have for a particular class. As an example, anything classified as a subclass of CombatUAV in the platforms taxonomy (a) has, by definition, a Firepower capability; similarly, every MALE-UAV provides both MediumAltitude and LongEndurance capabilities. Similarly, the various kinds of sensing modalities entail "INT-type" capabilities in (d). For example, any IR sensor provides IRINT capability; any Acoustic sensor provides ACINT, and so on. Where it is convenient to do so, we use the *subproperty* feature of OWL to define specialisations of the generic provides property to denote more specific kinds of provisioning. In (a), for example, we define properties such as hasRange and hasCoverage while in (b) we define properties such as providesFogPenetration and providesIdentification. For sensors, we also define the detects property to indicate the kinds of energy sensed by that kind of device (this property is adopted from CIMA,[11] and also similar to the measures property in OntoSensor[10]).

One of the most powerful features of a description logic such as OWL DL is the ability to define classes in terms of necessary and sufficient. New concepts can be defined by specifying property restrictions and relations on existing concepts, and a reasoning engine will "sort out" the resulting set of classifications, highlighting any inconsistencies. It is good practice in defining any DL-based ontology to state as little as possible explicitly about any concept, and to leave the reasoner to classify it to the greatest extent possible, so as to avoid stating redundant and inconsistent information. So, for example, we could define the necessary and sufficient conditions

for a UAV to be classed as a CombatUAV are that it provides Firepower capability. Stating then that the class PredatorB is a subclass of UAV and provides Firepower would result in PredatorB automatically being classified as a CombatUAV.

Figure 6 shows examples of both a platform (PredatorB) and a sensor (FLIR: forward looking infrared) class, as they appear using the Protégé ontology editor tool. The left-hand side shows the definition of PredatorB as a subclass of MALE-UAV. The PredatorB can carry several types of sensors, including optical sensors (FLIR) and synthetic aperture radar (SAR). It also has some datatype properties specifying endurance, range, etc. Some capabilities are inherited from superclass membership (e.g. MediumAltitude is inherited from MALE-UAV), while others are specifically asserted (e.g. Firepower). The right hand-side of Figure 6 shows the definition of the FLIR sensor class, which is a subclass of IR sensor. Among the capabilities stated here are that FLIR sensors are able to detect thermal energy, which gives them the ability to operate night and day; further, they have foliage penetration (FOPEN) capability.

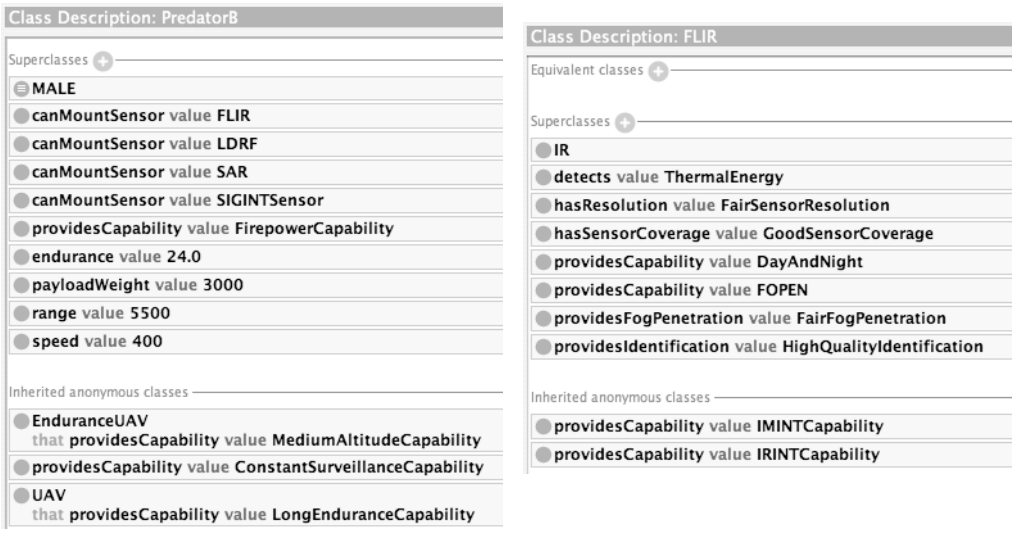

Figure 6. Examples of ISR classes: PredatorB platform and FLIR sensor

The next section describes how the ISR and MMF ontologies are used in our solution to the sensor-mission matching problem.

5. SENSOR-MISSION MATCHING PROCEDURE

Given some task T, our matching procedure uses the previously-described ontologies to recommend a set of *package configurations* (PCs) of types of platforms and sensors, such that each recommended PC meets all of the capability requirements of T. In other words, for every capability c_i required by T, there is at least one type of sensor or platform in each recommended PC that provides c_i. An individual PC can be as simple as a single type of platform with a single type of sensor mounted on it, or arbitrarily complex with many types of platform, each mounting a variety of sensor types. Note that the matching procedure works with sensor and platform types, where each type is formally defined by a class in one of the ontologies introduced above. The matching procedure does not consider instances of sensors or platforms, since all instances of a class share the same definition and so, from the point-of-view of capability provision, are identical. In other words, if a coalition has two Predator B aircraft at its disposal (two instances of the PredatorB class), it is not the job of the matching procedure to select a specific instance of Predator B, only to recommend that (some of) the capability requirements of some task can be met by a Predator B. We consider the subsequent problem of allocating sensor/platform instances to tasks in Section 7. However, as a pragmatic feature of the approach, we allow pre-filtering of the asset types based on availability. So for example, the matching procedure can be set up to consider only sensor/platform

types which are known to be available, based on information from the logistical database — this was shown in the top-right of Figure 1 in Section 1

Our matching procedure uses ideas from the Semantic Web Service literature, which has identified various kinds of "semantic" or "logical match"[14] between the specification of the capabilities of some service, and the specification of a user's requirements. We illustrate these with a simple example from our domain, where a night reconnaissance task T has a set of required ISR capabilities:

$\mathcal{C}^T = \{\mathsf{Night}, \mathsf{IRRadiation}\}$

That is, T requires night operations capability, and the ability to detect IR radiation**. Table 1 shows six potential solutions in terms of devices and their capabilities. The matching is complicated by the need to consider the subsumption (specialisation/generalisation) and disjunction relations between the capabilities required by the task, $\mathcal{C}^T$, and the capabilities offered by a set of assets, $\mathcal{C}^A$. In this example, DayNight capability is more specific than Day or Night, while the latter two are disjoint. Similarly, OpticalRadiation is more general than IRRadiation and VisibleLight, and the latter two are disjoint.

	Device type	Capabilities provided
$S1$	IRDevice	{DayNight, IRRadiation}
$S2$	OpticalDevice	{DayNight, OpticalRadiation}
$S3$	NightVisionDevice	{Night, VisibleLight}
$S4$	VideoDevice	{Day, VisibleLight}

Table 1. Some potential solutions to the example night reconnaissance task

Based on the Semantic Web Service literature we identify five possible outcomes from an attempt to match some $\mathcal{C}^T$ and $\mathcal{C}^A$. These are listed in decreasing order of desirability:

- *Exact match*: the required capabilities are identical to those provided by the assets, $\mathcal{C}^T \equiv \mathcal{C}^A$. This does not occur in our example because there is no device that only works at night and detects specifically IRRadiation.
- *Plugin match*: the required capabilities subsume (are less specific than) those provided by the assets: $\mathcal{C}^T \sqsupseteq \mathcal{C}^A$. This occurs between T and $S1$ because T's Night requirement is less specific than $S1$'s DayNight capability (where also IRRadiation matches exactly).
- *Subsumes*: the required capabilities are subsumed by (more specific than) those provided by the assets: $\mathcal{C}^T \sqsubseteq \mathcal{C}^A$. This occurs between T and $S2$ because T's IRRadiation requirement is more specific than $S1$'s OpticalRadiation capability (though DayNight matches exactly here).
- *Overlaps*: the conjunction of the required capabilities and those provided by the assets is not empty: $\mathcal{C}^T \sqcap \mathcal{C}^A \neq \bot$. This occurs between T and $S3$ because both include Night capability, although the VisibleLight and IRRadiation capabilities are disjoint.
- *Disjoint*: the conjunction of the required capabilities and those provided by the assets is empty: $\mathcal{C}^T \sqcap \mathcal{C}^A = \bot$. This occurs between T and $S4$ because the Night and Day capabilities, as well as the VisibleLight and IRRadiation capabilitires are disjoint.

It can be seen from these definitions that an exact match is a special case of plugin match, and only these kinds of match meet all of the requirements of the task. Therefore, we consider the exact and plugin matches to be *acceptable matches*, while the subsumes, overlaps, and disjoint cases are *unacceptable*. While we may consider an exact match to be better than a non-exact plugin match, in that there are no "extraneous" capabilities provided by the assets in the former, we have elected not to make this distinction in our matching procedure

**This example is intentionally simple, with a very low-level set of requirements. We would not normally expect so few requirements for a task, nor for a commander to specify requirements at the level of particular kinds of IR radiation.

as we believe there are more useful and significant ways of ranking solutions. This point is discussed further in Section 7.

There are three sets of constraints to be taken into account in finding acceptable matches of package configurations to tasks:

- *Task-sensor constraints*: these consider matches between the ISR capabilities required by the task, and those provided by sensor assets. For example, capabilities such as the intelligence types in Figure 5(d) in Section 4, as well as the kinds of energy detected by sensors (e.g. ThermalEnergy), and "environmental" capabilities such as foliage and fog penetration.
- *Task-platform constraints*: these consider matches between the ISR capabilities required by the task, and those provided by platform assets. Many of these are associated with the concepts in Figure 5(c) in Section 4, such as the range and endurance requirements which affect choice of platform (as seen in the simple UAV example in Section 2).
- *Platform-sensor matching*: these are mostly defined by the canMount property shown in the example in Figure 6 in Section 4, indicating that a kind of sensor can be carried by a kind of platform as its "payload".

We define a *platform configuration* $\pi = \langle P, \mathcal{S} \rangle$, where P is a type of platform, and $\mathcal{S} = \{S_1, \ldots, S_m\}$ is a set of sensor types that can be mounted in P simultaneously (there are no interferences among them). Given a task T with a set of required ISR capabilities $\mathcal{C}^T = \{C_1, ..., C_n\}$, a single platform configuration is a *valid match* for T if the combined capabilities of P and $\mathcal{S}$ satisfy $\mathcal{C}^T$, where satisfaction is computed as a *plugin* match:

$$\langle P, \mathcal{S} \rangle \in \mathcal{V}(T) \iff \forall C_i \in \mathcal{C}^T : (P \sqsubseteq \exists \mathsf{provides}.C_i) \sqcup (S_i \in \mathcal{S} \sqsubseteq \exists \mathsf{provides}.C_i)$$

where $\mathcal{V}(T) = \{\pi_1, ..., \pi_n\}$ is the set of valid matches for task T, and the notation $A \sqsubseteq \exists \mathsf{provides}.C_i$ denotes that some asset type A (sensor or platform) provides the capabilty C_i. (Any property that is a subproperty of provides is thus applicable here.)

More commonly, the requirements of a task will not be satisfied by a single platform configuration. We define a *package configuration* $\Pi = \{\pi_1, ..., \pi_n\}$, where π_i is a platform configuration. Π is a *valid match* for T if collectively the platforms and sensors in $\{\pi_1, ..., \pi_n\}$ satisfy the capabilities in $\mathcal{C}^T$, and Π is minimal with respect to $\mathcal{C}^T$ (there does not exist any subset of Π that is a *valid match* for T).

6. PROOF-OF-CONCEPT IMPLEMENTATION

To test the matching approach, we have implemented a prototype tool in Java that uses the Jena[††] and Pellet packages to process and reason with our OWL DL ontologies as described in the preceding sections. The core of the prototype is a procedure that computes valid matches in terms of platform configuration and package configurations for the aggregate ISR requirements of a task, as described above. The vast majority of the "work" here is done by the ontological definitions that define the subsumption relationships, allowing the Pellet reasoner to compute plugin matches, from which we then construct platform and package configurations via a simple set-covering algorithm. The prototype tool, called SAM (Sensor Assignment to Missions) is currently focussed on the UAV domain, but extending it is simply a matter of adding new ontological knowledge bases.

The tool offers a simple Web-based user interface allowing a commander to choose ISR requirements. For simplicity, this is currently done at the operational level: a user of SAM creates a mission and specifies a number of operations, then selects ISR requirements for each from a picklist as shown in Figure 7. The output of SAM is a list of recommended package configurations, as shown in Figure 8 (for example, the first package configuration involves a pair of UAV platforms, an I-GNAT and a WASP, each with specific sensors). In the current version of SAM, recommendations are ranked in order of cost, cheapest first.

SAM is intended only as a proof-of-concept; initial feedback from domain experts in the ISR field in both the US and UK militaries has been positive, and their feedback will be used to develop our ideas further. Some areas for future work are outlined below.

[††]http://jena.sourceforge.net/

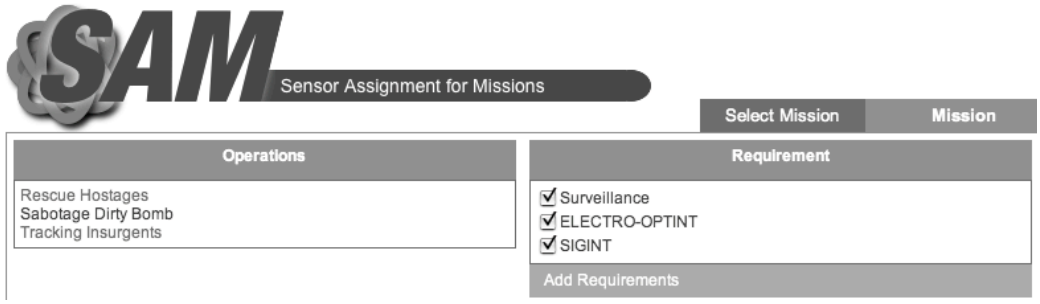

Figure 7. Sensor Assignment to Missions (SAM) tool: mission screen

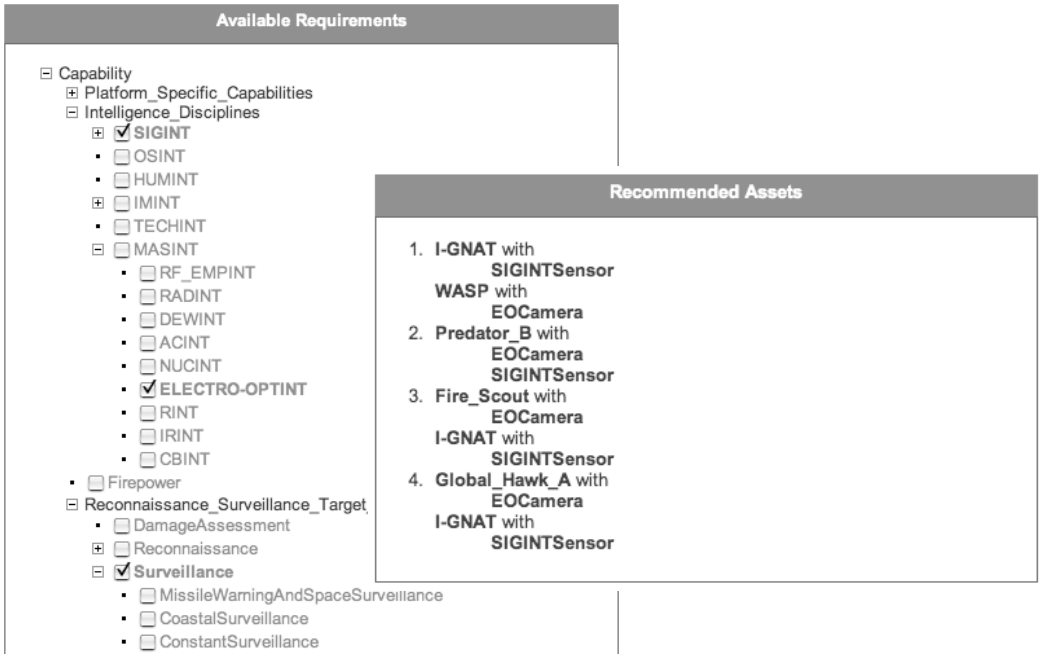

Figure 8. Sensor Assignment to Missions (SAM) tool: recommendation screen

7. CONCLUSIONS AND FUTURE WORK

In this paper, we have described a new approach to solving the sensor-mission matching problem using a collection of interlinked knowledge bases in the form of ontologies, represented using the OWL DL formalism. We have shown how the subsumption-based reasoning enabled by the semantics of the OWL DL representation allow us to define and compute matches between the ISR capabilities required by mission tasks, and those capabilities provided by sensor and platform assets. Returning to the wider vision of sensor-mission assignment illustrated in Figure 1 in Section 1, we conclude by identifying some areas of current and future work needed to realise this.

Ranking of recommendations Currently, our prototype implementation offers only a simple ranking based on cost ("cheapest-first"). We would like to address other ways of preferring one package configuration over another. Examples under consideration include the use of quality-of-information (QoI) knowledge about the various assets in terms of their ability to satisfy the ISR requirements. For example, given a particular Imagery Intelligence requirement, alternative imaging sensors will be known to provide information at various QoI levels; ranking could take this information into account ("best-first")[‡‡] Another possibility would involve factoring-in the readiness of assets in the ranking ("most available first"). In general, we doubt that "one-size-fits-all"

[‡‡]How to compute aggregate QoI for package configurations over sets of ISR requirements is an open research issue.

approach is sensible; instead, users of a future version of our matching tool would probably wish to set their own preferences on ranking of recommendations.

Resource allocation for deployment The sensor-mission matching subproblem is defined to consider only the types of assets and their capabilities. For deployment, the commander needs to assign instances of (available) assets, pre-configured into operational packages. To address this larger problem we need to consider resource allocation, where resources are instances of package configurations (comprising sensor and platform instances). In general, this will require the consideration of additional constraints on package composition from the operational environment. For example, the physical location of in-situ sensors will need to be factored-in, as will logistical data such as the current battery life and damage status, and also (in cases where the assets belong to a coalition) the ownership of assets in terms of who has the authority to assign them. We currently view the resource allocation procedure as a separate stage in the overall sensor-mission assignment problem, which takes as input the recommendations from sensor-mission matching, and uses these to reduce the potentially-large search space of possible allocations. This is an active area in our current research.

REFERENCES

[1] Joint Publication, "2-01: Joint and national intelligence support to military operations," (2004).

[2] US Army, "Field manual FM2-0: Intelligence," (2004).

[3] Gruber, T. R., "Toward principles for the design of ontologies used for knowledge sharing," *Journal of Human Computer Studies* **43(5/6)**, 907–928 (1994).

[4] Antoniou, G. and van Harmelen, F., [*A Semantic Web Primer, 2nd edition*], MIT Press (2008).

[5] Guarino, N., "Formal ontology and information systems," in [*Proceedings of the 1st International Conference on Formal Ontologies in Information Systems*], 3–15 (1998).

[6] Bermudez, L., Graybeal, J., and Arko, R., "A marine platforms ontology: Experiences and lessons," in [*Proceedings of the ISWC 2006 Workshop on Semantic Sensor Networks*], (2006).

[7] Sheehan, J. H., Deitz, P. H., Bray, B. E., Harris, B. A., and Wong, A. B. H., "The military missions and means framework," in [*Proceedings of the Interservice/Industry Training and Simulation and Education Conference*], 655–663 (2003).

[8] Tanenbaum, P. J. and Yeakel, W. P., "A framework linking military missions and means," in [*Mathematics for Industry: Challenges and Frontiers*], SIAM (2005).

[9] Robin, A., Havens, S., Cox, S., Ricker, J., Lake, R., and Niedzwiadek, H., "OpenGIS sensor model language (SensorML) implementation specification," tech. rep., Open Geospatial Consortium Inc (2006).

[10] Russomanno, D., Kothari, C., and Thomas, O., "Building a sensor ontology: A practical approach leveraging ISO and OGC models," in [*Proceedings of the International Conference on Artificial Intelligence*], 637–643 (2005).

[11] McMullen, D. and Reichherzer, T., "The common instrument middleware architecture (CIMA): Instrument ontology & applications," in [*Proceedings of the 2nd Workshop on Formal Ontologies Meets Industry (FOMI 2006)*], 655–663 (2006).

[12] Joint Staff, "Universal joint task list (UJTL)," (2002).

[13] Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., and Patel-Schneider, P. F., eds., [*The Description Logic Handbook: Theory, Implementation, and Applications*], Cambridge University Press (2003).

[14] Paolucci, M., Kawamura, T., Payne, T. R., and Sycara, K. P., "Semantic matching of web services capabilities," in [*ISWC '02: Proceedings of the First International Semantic Web Conference on The Semantic Web*], 333–347, Springer-Verlag, London, UK (2002).

Acknowledgements

This research was sponsored by the U.S. Army Research Laboratory and the U.K. 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 author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.