COSAR-TS Project Report

Applying DAML Languages and Ontologies in CoSAR-TS

Stuart Aitken and Austin Tate
June 23, 2003

Prepared for the DAML Program

Abstract

This technical report describes an application of DAML languages, ontologies and agents to a search and rescue scenario. This scenario provides both a test of the current technical infrastructure and capabilities of DAML, and of DAML-S in specific, and also explores the adequacy of the conceptualisations of DAML-S in support of issues such as authority and obligation, and policies governing these, in the access to knowledge and the handling of tasks. The CoSAR-TS project will also work towards a shared representation of tasks, processes and responsibilities, that can be utilised across the KAoS and I-X systems developed by IHMC, University of West Florida, and AIAI, University of Edinburgh.

Contents

Applying DAML Languages and Ontologies in CoSAR-TS

This technical report describes an application of DAML languages, ontologies and agents to a search and rescue scenario. This scenario provides both a test of the current technical infrastructure and capabilities of DAML, and of DAML-S in specific, and also explores the adequacy of the conceptualisations of DAML-S in support of issues such as authority and obligation, and policies governing these, in the access to knowledge and the handling of tasks. The CoSAR-TS project will also work towards a shared representation of tasks, processes and responsibilities, that can be utilised across the KAoS and I-X systems developed by IHMC, University of West Florida, and AIAI, University of Edinburgh.

This report begins with the scenario, then continues with an overview of relevant DAML resources. Next, we describe I-X, and its use in an agent system that enacts task handling in the aforementioned scenario.

1 The Scenario

The following scenario has been developed as a focus to explore the use of existing DAML agents, databases and services.

The scenario begins with:
a) an event which reports a downed airman in the Red Sea between the coastlines of Binni (to the West) and Arabello (to the East). In this initial scenario it is assumed that excellent location knowledge is available, and that there are no local threats to counter or avoid in the rescue. The airman reports his own injuries via his suit sensors in this initial scenario.

This is followed by:
b) an investigation of the facilities available to rescue the airman – there will be three possibilities, using: a US ship borne helicopter; a Goan helicopter from a land base in Binni; or a patrol boat from off the Arabello coastline.

And then:
c) the establishment of available medical facilities for the specialised injury reported - using the information provided about the countries in the region. It is expected that Arabello will have a hospital that is best placed to provide the facilities (to allow for an aspect of the scenario in which the facilities can only be used by the coalition in circumstances where no Goan aircraft/helicopters are involved.

Later enhancements to the scenario could involve possible threats from Agadez submarines in the area, lack of knowledge of the position of the airman requiring search

and sensor coordination, and a need to establish the medical condition of the airman, and later provide (perhaps mobile agent) medical monitoring aids during the rescue. It is also proposed that a much richer and involved scenario involving a downed airman on land near an Agadez Weapons of Mass Destruction Site, and in an area where coalition special operations forces are deployed, could be investigated.

2 DAML Languages and Ontologies

DAML-S is an ontology for web services, it describes their characteristics in terms of what they do, and their inputs and outputs. This ontology is implemented in DAML-O, a web ontology language which is an extension of RDFS - in the sense that additional language constructs are introduced and given semantics. These languages are all expressed within the RDF data model. On a purely syntactic level, they are expressed in XML.

DAML-O builds on other W3C standards: XML, RDF, and RDFS. Therefore, a short introduction to the relevant elements of these proposals is given.

In RDF (Resource Description Framework), 'things' (*Resources*) are identified through URIs, and are described in terms of simple properties and property values. The basic unit is the triple, composed of <subject predicate object>, for example, the *creator* of a html page can identified as follows:

<http://www.example.org/index.html
http://purl.org/dc/elements/1.1/creator
http://www.example.org/staffid/85740>

RDF triples represent a graph; subjects and objects are viewed as nodes, while predicates are seen as the edges in the graph. Resources can have types, denoted by `rdf:type`, i.e. *type* is a predefined predicate relating instances to classes. Defined classes include *Bag*, *Alt* and *Seq* for collections.

RDF does not define datatypes, these are stated by referring to some other resource. XML Schema datatypes can be referenced for this purpose. The namespace mechanism is commonly used to make RDF more readable.

RDF properties represent relationships between resources. However, there is no way to further describe these properties, or any relationships between these properties and other resources at a general level within RDF. RDFS provides this additional expressivity. RDFS allows classes to be defined, and the domain and range of properties to be stated. Consider the *author* relation, the domain would be the class: *Document*, and the range the class: *Person*.

The following list details some of the commonly used RDF and RDFS language primitives. As can be seen, the two languages are interrelated, RDFS providing an interpretation of RDF.

`rdf:type` relates instances to classes;
`rdfs:Class` the class of classes;
`rdfs:Resource` the class of everything, an instance of `rdfs:Class`;
`rdfs:Literal` the class of literal values e.g. string and integer, which may be plain or typed (instances of a datatype class);
`rdf:Property`, the class of properties, an instance of `rdfs:Class`;
`rdfs:domain`, the relation defining the domain of a property by specifying the class that domain instances must belong to, an instance of `rdf:Property;`
`rdfs:range`, the relation defining the range of a property by specifying the class that range instances must belong to, an instance of `rdf:Property;`
`rdfs:subClassOf` the subclass relation, holding of two classes, an instance of `rdf:Property;`
`rdfs:subPropertyOf` the subproperty relation, holding of two properties, an instance of `rdf:Property`;

RDFS has an XML syntax, a quick summary follows. To introduce a new class or relation an `rdf:ID` is given in the description:

```
<rdf:Description rdf:ID="name"> …
</rdf:Description>
```

To reference a class or relation `rdf:about` is used:

```
<rdf:Description rdf:about="URIRef"> …
</rdf:Description>
```

A more concise way to introduce a new instance of an existing type, for example, a new class, is as follows:

```
<rdfs:Class rdf:ID="name"> …
</rdfs:Class>
```

The body of the description is a set property elements---each is a triple---where the subject is identified by the `ID` or `about` expression. The property elements give the predicate and the object of the RDF triple, three commonly used patterns are:

```
<ns:predicate> plain literal </ns:predicate>  |
<ns:predicate rdf:datatype="URIRef#type"> literal
</ns:predicate>  |
<ns:predicate rdf:resource="URIRef"/>
```

The predicate may be stated using the namespace abbreviation (`ns:predicate`), while the object may be enclosed in tags, optionally with a datatype specified, or may be a URIRef. In the above descriptions, the *`URIRefs`* should be the full URI - namespace abbreviation is not permitted in these cases.

2.1 DAML-O

DAML-O is an ontology language, built on RDFS, and with efficient reasoning support based on Description Logic (DL) reasoners. The semantics of the class and role definition constructs are based on those of DL. For reference, these are listed in Table 1.

Construct	Syntax	Semantics
Atomic Construct	A	A ⊆ U (the Universal set)
Atomic Role	R	R ⊆ U × U
Conjunction	C ⊓ D	C ∩ D
Disjunction	C ⊔ D	C ∪ D
Negation	¬C	U\C
Exists Restriction	Some R.C	{ x | ∃y <x,y> ∈ R ∧ y ∈ C }
Value Restriction	All R.C	{ x | ∀y <x,y> ∈ R → y ∈ C }
Role Hierarchy	R ⊑ S	R ⊆ S

Table 1. Syntax and semantics of a Description Logic

DAML-O adds 12 classes and 26 properties to RDFS. A DAML-O knowledge base is a set of RDF triples whose meaning is defined by the DAML-O language. The key language constructs are introduced below.

`daml:Class` defines a class element – this refers to a class name (URI) and can be defined in terms of an enumeration; by the `rdfs:subClassOf` relation; by the `daml:disjointWith` relation; or by a boolean combination of class expressions. A class expression is a class name; an enumeration of classes; a property restriction; or a boolean combination of these. A boolean combination is created by the operators of intersection, union or complement. DAML-O defines the classes of *Thing*, the class of everything, and *Nothing*, the class with no members.

Property restrictions qualify a defined class, *A*, in terms of an existing class *C*. For example, *RedWine* is *Wine* where the property *hasColour* must be *RED*:

```
RedWine:= Wine ⊓ All hasColour.RED
```

The DAML-O property restrictions are:

`daml:toClass` is the value restriction, i.e. for all *x*, if *property(x,y)* holds of an element *y*, then *y* is in *C*

`daml:hasClass` is the exists restriction, i.e. for some *x*, *property(x,y)* holds of some element *y* of *C*

DAML-O provides a number of cardinality restriction relations. These permit the definition of a maximum, minimum, or exact cardinality. Datatypes are expressed in RDF syntax, and given semantics in terms of XML Schema datatypes. Classes can be defined by any of the methods described above, hence, in the general case, the subsumption relation (and the class hierarchy) must be determined by proof.

The DAML-O classes and relations are axiomatised by translating the RDF triples into first-order logic. The basic idea is to translate a triple: `<subject predicate object>` to a logical assertion: *(PropertyValue predicate subject object)* in KIF. For convenience, the type relation is defined in KIF by the rule:
(<=> (PropertyValue type ?x ?y) (Type ?x ?y))
the semantics of classes and relations are defined by FOL axioms, for example, *subClassOf* has the definition:

```
(<=> (PropertyValue subClassOf ?C ?D)
     (and (Type ?C rdfs:Class) (Type ?D rdfs:Class)
     (forall (?z) (=> (Type ?z C) (Type ?z ?D)))))
```

2.2 DAML-S

DAML-S is an ontology, expressed in DAML-O, for the description of web services. It is intended to be used in an agent framework to allow agents to advertise, discover, invoke, compose and monitor web services. A web *Service presents* a *Service Profile* (a description of the service on offer), a *Service Model* (how the service is to be achieved) and supports a *Service Grounding* (which supplies the details of the protocols and ports to be used).

The *Service Profile* contains the name of the *Service*, contact information, plus a textual description. It is *providedBy* an *Actor*, which in turn has a title, phone, fax, email, physical address and a URL.

A *Service* is *describedBy* a *Service Model*, being a process model which is described in terms of a process ontology. The process ontology distinguishes three subtypes of *Process*: *Atomic Process, Composite Process* and *Simple Process*. It also contains control constructs which are used to describe how component processes are combined into a composite process. Control constructs include: *sequence, split, split+join, unordered, choice, if-then-else, iterate* and *repeat-until*. These are described in text, but no official formal definition is provided. However, both (Ankolekar et al., 2002) and (Narayanan et al., 2002) have developed formalisations for the purposes of verification and simulation.

The *Service Grounding* that a *Service supports* defines the concrete means of realising the service. The abstract notions of the ontology and process models are expressed in terms of existing XML-based technologies: WSDL (Web Service Description Language), SOAP (Simple Object Access Protocol) and UDDI (Universal Discovery, Description and Integration).

Both the *Service Profile* and the processes in a *Service Model* can have *inputs, outputs, conditions* and *effects* associated with them. Each of these properties is represented by a *Parameter Description*, composed of:

parameterName: the name, a literal or URI;
restrictedTo: a restriction on the values the relation holds of;
refersTo: a reference to the process model.

It is common to state that the thing output by one process is the same thing input to another. However, Description Logics, and ontologies based on them, lack the variable binding mechanism that is normally used for this purpose. All we can do is specify the types of the inputs and outputs. A solution to this problem will come from the DAML-Rules initiative. In the meantime, a solution based on annotating processes with information about how objects are shared between processes has been proposed (DAML-S Coalition 2002).

In summary, services can be identified by name; by considering the *Service Profile* *inputs, outputs, conditions* and *effects*; and by considering the *Service Model.*

2.3 Accessing DAML-S Services

CMU have implemented a semantic matchmaker which matches advertisements and service requests – expressed as DAML-S *Service Profiles*. The profiles contain the name of the service, plus the input and output. Subsumption and heuristic measures are used to evaluate the candidate matches. The CMU system implements a UDDI grounding, connecting DAML-S to existing technology.

3 I-X and <I-N-C-A>

I-X and <I-N-C-A> are technologies developed at AIAI, The University of Edinburgh. I-X is a new systems integration architecture. Its design is based on the O-Plan agent architecture (O-Plan Team 1999). I-X provides an issue-handling workflow style of architecture, with reasoning and functional capabilities provided as plug-ins. Also via plug-ins it allows for sophisticated management of the internal model representations. I-X agents may be recursively or fractally composed, and may interwork with other processing cells or architectures. [1] <I-N-C-A> is a meta-model supporting interoperability of planning and design systems. It forms the basis of the conceptual framework of I-X. The key notions are:

Issues: unsatisfied objectives or questions;
Nodes: activities or objects;
Constraints: relationships between *Nodes* (and other objects) constraining the space of interpretations;
Annotations: decision rationale and other notes;

3.1 DAML-S and I-X

A service can be seen as an event-type or activity, thus, services could be modelled as *Nodes*. Inputs and outputs of the *Service Profile* (in DAML-S) can be seen as *Constraints*. Following the DAML-S ontology, it is necessary to represent the *presents* relationship between the profile and the service. (We might also interpret the *Service*

[1] Taken from http://www.aiai.ed.ac.uk/project/ix/

Profile as an event-type, but this does not appear to be the intended interpretation.) The associated inputs and outputs will typically specify a class, e.g. *Price* and *Car* will be the argument types of these roles in a *BuyCar* Service:
Node: BuyCar-Service
Constraints: [presents(BuyCar-Service,BuyCar-Profile), input(BuyCar-Profile,Price), output(BuyCar-Profile,Car)]

In a request for the *BuyCar* service, the description might be:
Node: BuyCar-Request
Constraints: [presents(BuyCar-Request,BuyCar-Request-Profile), input(BuyCar-Request-Profile,Price), output(BuyCar-Request-Profile,FordExplorer)]

However, we might observe that a more conventional model would associate the inputs and outputs with the event-type (i.e. with the *Service*) rather than the profile:
Node: SellCar-Service
Constraints: [input(SellCar-Service,Price), output(SellCar-Service,Car)]
Alternatively, we might consider the *Service Profile* as the *Node*, and ignore the *presents* relation.

The process model of the *Service Model* could be represented in a to-do list. DAML-S processes and sub-processes are clearly activities. Some simplification of the DAML-S control constructs might be required to represent a process in an I-X Panel. An I-X process could represent a sequence of service requests, or a mixture of user tasks and service requests. The combination of processes in an I-X Panel could itself be a Service Model – and advertised as a DAML-S service.

4 Applying DAML-S

In the CoSAR-TS scenario, the I-X agent requests an information service from an external agent. This agent might be found by a broker agent. The information request will be an element of a search and rescue process. Identifying existing DAML resources, such as DAML tools, agents and databases, to assist in implementing the scenario is a critical, on-going, task.

The basic information request service might only require the inputs and outputs of services to be known. The extension of the basic scenario to consider policies, authorities and permissions might exercise additional aspects of DAML-S. Authority could be represented as a *condition* of a *Service*: clearance to enter airspace could be an example. Permission for an agent to access a resource might also be a condition on a service, achieved as a result of a subactivity.

4.1 The May 2003 Demo

In COSAR-TS, DAML-S is applied in the context of existing multi-agent architectures, communication protocols and policy services. The KAoS architecture (Bradshaw et al 1995) provides an agent framework, supporting agent registration, communication and adherance to policies (though a publication mechanism). The CMU matchmaker is embedded in a distinct multi-agent framework. Therefore, the use of DAML-S is not only a question of representation and description of services, agent architecture is also an important issue. In the integration of I-X with these existing components, we consider both representation and architectural factors.

Considering the knowledge content and ontologies of the demo, a number of DAML ontologies define the SAR domain and the services that are available. Knowledge of medical facilities is obtained from a medical DAML ontology stored in the BBN SONAT database. This has been extended to include data on the countries in the Binni scenario: Agadez, Arabello, Binni and Gao. The medical ontology includes several instances of *Hospital* and other medical facilities, and associated information provides the latitude and longitude of hospital locations. Services that may be invoked are defined in DAML-S, for example, the *GaoMarineHelicopter* service profile, which has the associated atomic process *PickUpDownedPilot*. This profile is provided by *Gao*, and has an input defined by the (constrained) parameter description *PickUpLocation* which refers to the property *pickUpLocation_In* restricted to *Location*[2]. The *HospitalLocation* and *CountryOfHospital* are further inputs to the service profile, and these have similar definitions. These resources provide the domain knowledge and service capabilities that are required to plan the SAR mission.

Turning to the agent architecture of the demo, four agents are created: CoSAR, US-SAR, Hospital, and Resource, with the respective roles and functions: Coalition SAR coordinator, US SAR officer, hospital information provider, and SAR resource provider. The Coalition SAR coordinator has an I-X process panel, which can be used to follow a standard operating procedure. This is represented by a rescue process in I-X, which, at the top-level, contains the five activities: *select hospital*; *establish country*; *select SAR resource*; *notify SAR resource*; *notify hospital*; which are executed sequentially. The *select hospital* activity is broken down further into three steps, one of which is *lookup hospital* - an action which can be carried out by querying the SONAT database of DAML-encoded facts (as described above). The decomposition of the *select SAR resource* activity includes *lookup SAR resource*, an activity that can be carried out by querying the CMU matchmaker for a service with a matching profile (i.e. one of the rescue services encoded in DAML-S such as *GaoMarineHelicopter*).

The SAR demo combines the above components within the IHMC KaoS policy enforcement restrictions as follows: After the agents have been created a policy on overlying Gao is published, and enforced by agents which manage inter-agent communication. The US SAR agent requests a rescue for the downed airman, and this

[2] Thanks to Andrezj Uszok of IHMC for developing these ontologies.

request is passed to the Coalition SAR coordinator who expands it into subactivities. The *lookup hospital* action is passed to a SONAT agent, which extracts the response from the DAML knowledge base. In a later step in the standard operating procedure, the Coalition SAR Coordinator requests a rescue service, knowing the locations of the airman and the hospital, and the injuries sustained. This service is located via the CMU matchmaker over a SOAP connection. The responses to this query are filtered such that only those conforming to the published policy are passed back to the SAR Coordinator. This filters out the use of a Goan helicopter in cases where a medical facility in Arabello is used. The demo is completed once all activities are discharged by the Coordinator.

The demo illustrates the use of knowledge bases and services, both represented in DAML, in a distributed environment. DAML-S is exercised to the extent that services are selected through their input conditions, that is, through restrictions on the type of objects that the service process *PickUpDownedPilot* can act on. In future, outputs, preconditions and effects might also be specified, and used in the service matching procedure, and in the selection activity that occurs in the operating procedure once a set of results (hospitals, rescue services) are returned to the Coalition SAR Coordinator. The I-X agents themselves might locate each other via DAML-S service advertisement and matchmaking, and might build-in the service representation to the process in a more direct way. The enforcement of policy could also be seen as a precondition on services, i.e. it might be made an explicit part of the service description.

5 Conclusions

The CoSAR-TS scenario provides a realistic testbed for DAML-S services, and for the evaluation of the conceptualisations of the DAML-S ontology. This report has presented some of the most relevant elements of the DAML technologies that are required to achieve the evaluation goals. While still at an initial simplistic level, the demo has successfully integrated two existing multi-agent frameworks, operating over different communication layers and protocols, and has accessed DAML and DAML-S knowledge bases of facts and services.

6 References

Ankolekar, A., Huch, F., and Sycara, K. (2002) Concurrent Execution Semantics for DAML-S with Subtypes. *Coordination Models and Languages* pp. 14-21.

Boley, H., Tabet, S. and Wagner, G. (2001) Design Rationale of RuleML: A Markup Language for Semantic Web Rules *International Semantic Web Working Symposium (SWWS).*

Bradshaw, J.M., Dutfield, S., Carpenter, B., Jeffers, R., and Robinson, T. (1995) KAoS: A Generic Agent Architecture for Aerospace Applications *Proceedings of the CIKM '95 Workshop on Intelligent Information Agents*

DAML Services Coalition (2002) DAML-S: Semantic Markup for Web Services http://www.daml.org/services

Dean, M. and Schreiber, G. (eds.) (2003) Web Ontology Language (OWL) Reference Version 1.0 W3C Working Draft 21 February 2003

Grosof, B., Horrocks, I., Volz, R., and Decker, S. (2003) Description Logic Programs: Combining Logic Programs with Description Logic. *Proc. WWW2003*.

van Harmelen, F., Patel-Schneider, P.F. and Horrocks, I. (2001) Reference description of the DAML+OIL (March 2001) ontology markup language. http://www.daml.org/2001/03/reference

Manola, F. and Miller, E. (eds.) (2003) RDF Primer W3C Working Draft 23 January 2003 http://www/w3.org/TR/rdf-primer

Martin, D. and Burstein, M. et al. (2002) Describing Web Services using DAML-S and WSDL. DAML-S Coalition working document, August 2002

McGuinness, D.L., Fikes, R., Hendler, J. and Stein, L.A. (2002) DAML+OIL: An Ontology Language for the Semantic Web *IEEE Intelligent Systems* Sept/Oct 2002, pp 72-80

Narayanan, S. and McIlraith, S. (2002) Simulation, Verification and Automated Composition of Web Services *Proc. WWW 2002*

O-Plan Team (1999) O-Plan Website http://www.aiai.ed.ac.uk/project/oplan

Paolucci, M., Kawamura, T., Payne, T., and Sycara, K. (2002) Semantic Matching of Web Service Capabilities. *Proc. ISWC 2002*

Tate, A. (2003) <I-N-C-A>: An ontology for Mixed-initiative synthesis tasks. http://www.aiai.ed.ac.uk/project/ix/

Volz, R., Decker, S. and Oberle, D. (2003) Bubo - Implementing OWL in rule-based systems *Proc. WWW 2003*