Toward Semantic Interoperability in Agent-Based Coalition Command Systems

David N. Allsopp, Patrick Beautement, John Carson and Michael Kirton
{d.allsopp, j.carson, m.kirton}@signal.QinetiQ.com, pbeautement@QinetiQ.com

QinetiQ Ltd
Malvern Technology Park
St Andrews Road, Malvern, WR14 3PS
United Kingdom

Abstract

The Coalition Agents Experiment (CoAX) is an international collaboration carried out under the auspices of DARPA's Control of Agent-Based Systems (CoABS) programme. The overall aim of CoAX is to demonstrate how an agent-enabled infrastructure, based upon the CoABS Grid, can enhance interoperability between heterogeneous components, including actual military systems, in a realistic scenario. The scenario is based on a peace-enforcement operation in the year 2012 in 'Binni' (a mythical state in Africa) which requires a mix of Coalition forces to work together.

The agent infrastructure allows the construction of a coalition command support system, with agents grouped into domains to reflect real-world organisational and national boundaries. Each domain is a community of agents that has its own secure communications, capabilities and information spaces, and is governed by policies at the domain, host, virtual machine and agent levels.

Communications between agents, whilst given some minimal semantic grounding via the use of conversation policies, should in future utilise emerging Semantic Web technology. In particular, the Resource Description Framework (RDF) and the DARPA Agent Mark-up Language (DAML) promise to deliver a greater degree of semantic interoperability. This paper describes practical experiences taking initial steps towards this goal, implementing agents that use a query language to exchange data between RDF models and implementing a prototype RDF browser. We discuss the specific requirements for querying, merging and attributing of RDF data by agents in a coalition environment, and the particular restrictions affecting agents on controlled networks, rather than on the Web. Finally, some challenges and requirements for the future are outlined.

1. Introduction

In order to demonstrate how planning, visualisation and execution activities in coalition operations can be augmented by agent technology, a collaborative programme of work has been put together under DARPA's Control of Agent-Based Systems programme (CoABS). This work involves 16 partner organisations, and is entitled the Coalition Agents Experiment (CoAX) [1,2].

This paper begins with an introduction to the challenges of coalition operations and our technical approach to solving them, and gives an overview of the CoAX demonstrations. It then outlines the infrastructure and systems in the current demonstration and the techniques used to integrate them. In evaluating this demonstration, areas that would benefit from Semantic Web technology are noted.

Our initial work on implementing agents that use RDF, including a basic query language and a RDF browser, is then described. This is followed by discussion of the benefits gained from this approach and the limitations of the current technology, in particular the querying and merging of RDF models and the attribution of data via the reification mechanism. The importance of software tools for rapid integration is emphasised. Finally, we note the potential advantages of the emerging DAML language [3] over RDF, and discuss future work.

2. Background

The year is 2012, and climate change in the Sudanese plain of East Africa has enabled the production and export of wheat in large quantities. The only way to transport this increasing volume of food to the European market is by sea. Competition over sea port access has led the government of Gao to launch a pre-emptive strike to open a corridor to the sea, declaring the annexed area to be the independent country of Binni. This action has incensed the government of neighbouring Agadez, who have launched repeated guerrilla activities to dislodge the Gao forces. Because of this dangerously unstable situation in Binni, the UN has passed a Resolution to create and deploy a UN War Avoidance Force for Binni (UNWAFB).

"Binni - Gateway to the Golden Bowl of Africa" [4] is a hypothetical scenario based on the Sudanese Plain. The countries of Gao, Agadez and Binni are fictitious, as are the events, organisations and personalities that lead to the crisis requiring UN intervention.

The vignette used for the CoAX experiment concerns a specific operation in which the UN forces are attempting to keep mutually hostile forces apart long enough to enable peace negotiations (figure 1). Incomplete, changing information, as well as deliberate misinformation from some parties hampers the UN efforts.

2.1 Coalition operations

Coalition military operations will become an increasingly important feature in future years. In any military operation, enabling commanders to have access to timely and relevant information is crucially important to a successful outcome. The difficulties are compounded in the virtual organisation of the coalition since there will be a mixture of equipment, operational procedures, languages, etc. Moreover, there is a pressing need to set up such organisations rapidly in order to respond decisively to emerging crises.

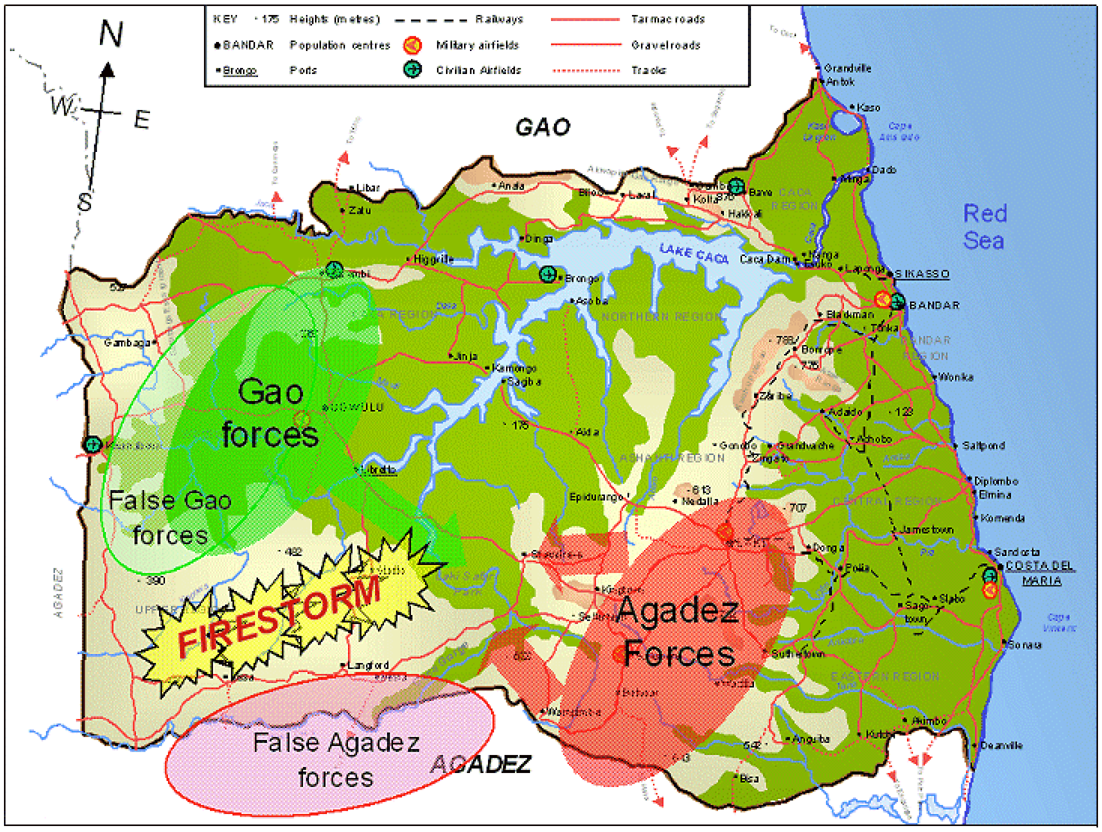


Figure 1. The fictional nation of Binni. The UN forces are considering a controversial firestorm to separate the warring forces of Gao and Agadez. Misinformation from Gao agents initially leads the UN to believe that the forces are further to the west than is in fact the case.

2.2 Technical approach

From a technical perspective, coping with the inherent heterogeneity and tight time-scales are major challenges. Traditional approaches to software integration are too inflexible to share information between such disparate command systems at short notice. Agent-oriented approaches are believed to offer advantages in such environments [5].

The principal motivation of the CoAX experiment is to investigate the conjecture that software agent technology can provide an advanced infrastructure able to support the demanding information, Command and Control requirements of a coalition force [2].

For the purposes of this research, an agent is defined as a software entity acting on behalf of, or mediating the actions of, a human user and having the ability to autonomously carry out tasks to achieve goals or support the activities of the user. Here, agents are supporting a community of human experts; they must not ‘take over’ or become obtrusive or obstructive. Their purpose is to help people cope with the complexities of working collaboratively in a distributed information environment. Agents operate mostly behind the scenes, integrated into familiar tools and methods of working, linking and fusing disparate sources of information as available — tasks well-matched with Semantic Web technologies. The CoAX project is producing a series of staged demonstrations of increasing complexity, showing agents and agentwrapped legacy systems communicating over the CoABS Grid [6], developed at Global InfoTek, Inc (GITI). The Grid is a framework for federating heterogeneous systems. Although the Grid is being developed with a military application in mind, it is a general-purpose agent framework with potential use by a variety of applications.

In recent demonstrations, agents are grouped into domains, using the Knowledgeable Agent-oriented System (KAoS) from Boeing and the University of West Florida's Institute for Human and Machine Cognition (IHMC) [7]. This system enables domain policies to be changed at runtime; for example, to cut off communications with a domain containing hostile agents or to drastically reduce CPU and network resources to agents launching a denial-of-service attack.

The current demonstration at the time of writing has grown to 25 agents grouped into 6 domains; elements of this demonstration are outlined below. The full demonstration (documents, images and video) may be seen in detail on the CoAX web-site [1]. The demonstration is genuinely heterogeneous, comprising systems and agents from at least 6 different organisations (including two real military systems) with more to be added in future demonstrations for 2001 and beyond.

3. Current demonstration

3.1 CoABS Grid Infrastructure

At the most basic level, the agents and systems to be integrated require infrastructure for discovery of other agents, and messaging between agents. This is provided by the CoABS Grid. Based on Sun's Jini services [8], the Grid allows registration and advertisement of agent capabilities, and communication by message passing. Agents can be added or removed, or their advertisements updated, without reconfiguration of the network. Agents are automatically purged from the registry after a short time if they fail. Multiple lookup services may be used, located by multicast or unicast protocols.

In addition, the Grid provides functionality such as logging, visualisation, encryption and authentication.

3.2 Knowledgeable Agent-oriented System (KAoS)

At a higher level, the KAoS framework is used to group agents into domains, to facilitate policy administration. Agents in a domain are subject to the policies of that domain. A given domain can extend across host boundaries and, conversely, multiple domains can exist concurrently on the same host. Policies can be scoped variously to individual agent instances, agents of a given class, agents running on a given host or instance of a platform (e.g., a single Java VM), or agents in a given domain or sub-domain [7]. The KAoS Policy Administration Tool (KPAT), a graphical user interface to domain management functionality, has been developed to make policy specification, revision, and application easier for administrators without specialized training.

The concept of policy-based management necessarily extends beyond typical security concerns. For example, KAoS policies will ultimately be used to represent not only *authorization, encryption, access* and *resource control* policies, but also *conversation policies, mobility policies, domain registration policies*, and various forms of *obligation policies*. The

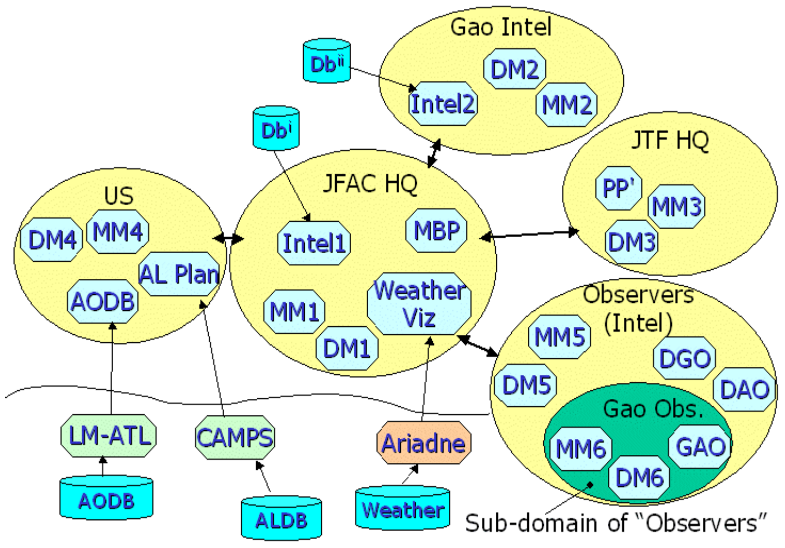


Figure 2. Agents are shown grouped into domains. Each domain contains two specialised agents: the Domain Manager (DM) and Matchmaker (MM). There are domains for countries (the US, and the fictional country of Gao), for coalition structures (such as the Joint Forces Air Component HQ), and for functional groups (Observers). Some agents (shown below the line) are not domain-aware, but have proxies within the domains. Various agents are associated with databases (DB). Other agents (MBP, Intel, CAMPS, WeatherViz, PP) are described below.

KAoS policy representation is currently very simple but an implementation of a more sophisticated DAML-based policy representation will be available later this year.
The domain structure of the current demonstration is shown in figure 2. Each domain contains two specialised agents, the Domain Manager, which enforces (directly or indirectly) the domain policies, and the Matchmaker, which provides functionality similar to the Grid registry.

KAoS also provides support for defining and using *conversation policies*: the structuring of messages for particular purposes or speech acts, such as Inform, Query, Request and others. Each basic policy forms a finite state machine, where each message, labelled with a verb identifying its purpose, represents a transition between states [9]. More general constraint-based DAML policy representations and mechanisms incorporating additional communications aspects, such as time limits, and the ability to compose policies from smaller fragments, are under development [10].

3.3 Systems integrated

The demonstration, at the time of writing, includes 25 agents from about six different organisations, grouped into six domains, written using three different programming languages. Some of the main elements are described briefly below.

3.3.1 Master Battle Planner (MBP)

A core agent in the demonstration is MBP – a highly effective visual planning-tool for air operations. MBP assists air planners by providing them with an intuitive visualisation on which they can manipulate the air intelligence information, assets, targets and missions, using a map-based graphical user interface (figure 3).

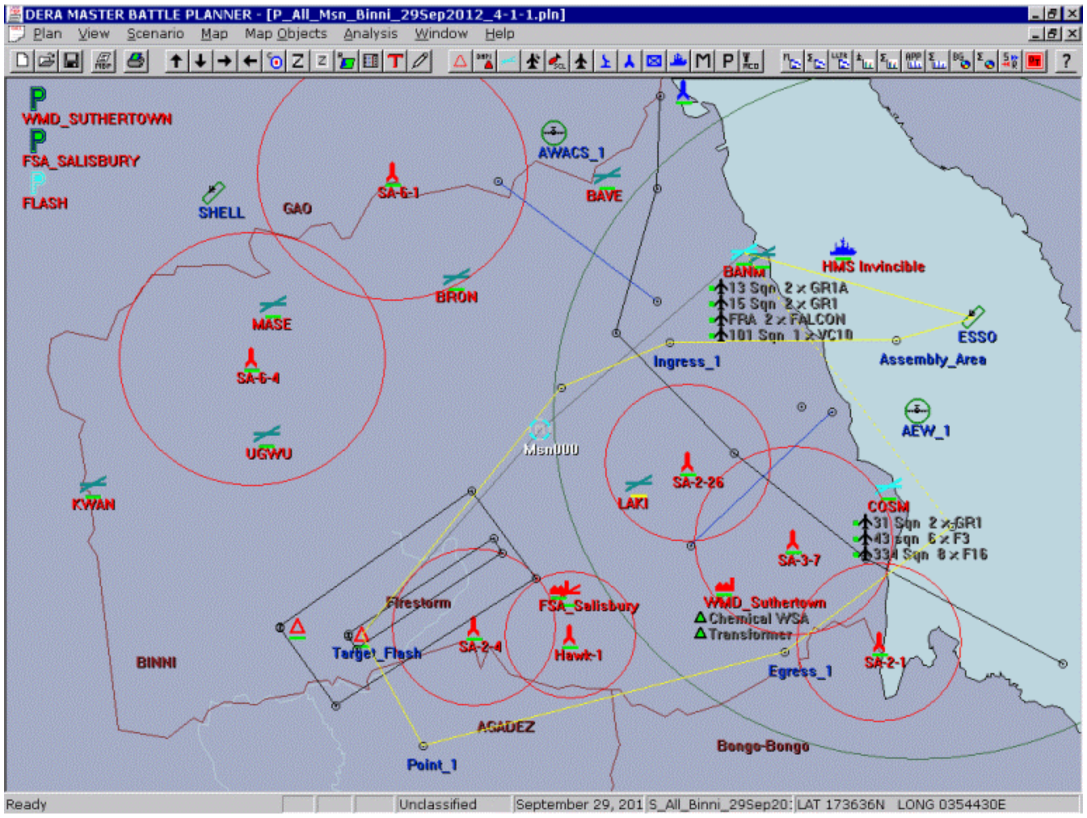


Figure 3. Master Battle Planner map display of the fictional countries of Binni, Gao and Agadez. A selected mission is highlighted in yellow; proceeding from an airbase, to refuelling tanker, via waypoints and airspaces to the target, and back to base by a different route.

The operator can interact with these operational entities and can plan individual air missions (or complex packages of missions) by dragging and dropping offensive units onto targets on the map. Supporting / defensive elements are added in the same way. The system provides the operator with analytical tools to assess the planned air operations.

MBP is a monolithic C++ application, which has been agent-enabled by wrapping it in Java code, using the Java Native Interface, and providing a proxy agent which communicates with the wrapper using the JavaSpaces API [11]. The agent enabling of MBP allows it to receive scenario data (targets, assets, airspaces etc) from other agents (Intel, figure 2), and update this information continuously. Information concerning other air missions can be accepted and merged with missions planned within MBP; export of mission data to other agents is under development.

3.3.2 Consolidated Air Mobility Planning System (CAMPS)

The second real military system integrated into the demonstration is Air Force Research Laboratories' CAMPS Mission Planner. CAMPS develops notional schedules for aircraft to pick up and deliver cargo within specified time windows. It takes into account numerous constraints on aircraft capabilities, port capabilities, etc. [12–14].

In the demonstration scenario, CAMPS schedules airlifts of cargo into Binni. These airlift flights could potentially conflict with offensive air missions, so the scheduled flights are requested from the CAMPS agent, translated by another agent, and sent to MBP, forming part of the normal MBP air visualisation.

This is an interesting example, as only partial translation is possible; CAMPS and MBP differ fundamentally in their definition of air missions. A CAMPS mission consists of an arbitrary collection of flights, where a flight is a single journey from A to B by a single aircraft. However, an MBP mission consists of a starting point and a route, *which must return to the starting point* (perhaps by a different path), and may consist of multiple aircraft. CAMPS can therefore produce routes that have no fully valid representation in MBP, although they could be partially represented or indicated graphically. This is a fundamental limit on the achievable degree of interoperability.

3.3.3 Ariadne

In a similar manner, weather information extracted from web-sites by the Ariadne system from USC/ISI is gathered by the WeatherViz agent (figure 2). It is translated and forwarded to MBP, again forming part of the normal picture of the air situation. Ariadne facilitates access to web-based data sources via a wrapper/mediator approach [15,16]. Wrappers that make Web sources look like databases can be rapidly constructed; these interpret a request (expressed in SQL or some other structured language) against a Web source and return a structured reply. The mediator software answers queries efficiently using these sources as if they formed a single database. Translation of the XML from Ariadne into the XML expected by MBP was initially handled by custom code, but can now be performed more easily using XSL Transformations [17].

3.3.4 Process Panel

The Artificial Intelligence Applications Institute (AIAI) of the University of Edinburgh has provided its IP^2 Process Panel agent (PP, figure 2), which provides user level, configurable task and process management aids for inter-agent co-operation [18]. Each user may have their own panel to reflect their role in a co-operative process. The Process Panel keeps track of the air planning process through inter-agent messages.

3.3.5 NOMADS agents

The NOMADS mobile agent system from IHMC is used in the demonstration to allow untrusted agents (Gao Observer) to run in the Observer domain alongside trusted agents (DGO, DAO, figure 2). The Aroma virtual machine provides dynamic resource control mechanisms, protecting the host from a malicious or buggy agent [19]. When a denial-of-service attack is mounted by an agent from Gao, the excessive usage of CPU, disk and network is detected and a change of policy is automatically executed, in concert with the KAoS domain management mechanisms. A human operator can then choose to lower the resource limits even further using KPAT.

3.3.6 Future additions

Future demonstrations for 2001 and beyond have introduced further agent systems and capabilities. These include a multi-level co-ordination agent from the University of Michigan [20]; a field observation system using Dartmouth College's D'Agents technology [21]; and eGents (agents communicating over e-mail) from Object Services and Consulting, Inc [22]. This demonstration extends into the execution phase of coalition operations, showing near real-time visualisation of air operations based upon data from agents.

4. Evaluation of demonstration

4.1 Aims and achievements

The aim of this demonstration was to investigate how software agent technology can provide an advanced coalition infrastructure. We were successful in integrating a variety of real military systems and agents, and were able to demonstrate increased functionality and interoperability between previously stand-alone systems. The systems were written separately by different organisations, in different languages, under different operating systems. Dynamic data sources were an important feature, removing the reliance on static data files. Agents also integrated data from disparate systems seamlessly as far as the user was concerned; weather and airlift missions were merged into the MBP view of air operations, for example.

Agent enabling of real legacy systems via wrappers was found viable, even where the original system made no allowance for integration with other systems.

Using KAoS, the ability to group agents into domains was demonstrated, and to change the domain policies dynamically; for example, to cut off communications with a domain containing hostile agents. Using KAoS domain management tools in conjunction with the Aroma virtual machine also allowed dynamic resource limits to be applied to individual agents to prevent denial-of-service attacks.

4.2 What is missing?

However, the vision of the CoAX project is to achieve rapid, dynamic coalition formation, in which agent domains are created and removed on-the-fly, and agents come and go. Interoperability between agents should be achieved very rapidly, with as little human intervention as possible. Agent interoperability *was* achieved relatively easily, due to the discovery and messaging facilities provided by the CoABS Grid, but message structures, primarily in XML, were pre-agreed, or translators were hand-coded. This process is time-consuming. Tools such as XSLT can accelerate the process (and for rapid integration, graphical tools based on languages such as XSLT may still be valuable) but are still human driven. A fundamental problem for such agents is that there is no mechanism for sharing terms and relations (via shared or partially shared ontologies). Consequently, messages have no unambiguous meaning (even to humans) outside of the agent that generates them. XML provides shared syntax, but not shared meaning.

4.3 How can Semantic Web technology help?

Berners-Lee *et al* write: [23] "Some low-level service-discovery schemes are currently available, such as Microsoft's Universal Plug and Play, which focuses on connecting different types of devices, and Sun Microsystems's Jini, which aims to connect services. These initiatives, however, attack the problem at a structural or syntactic level and rely heavily on standardisation of a predetermined set of functionality descriptions. Standardisation can only go so far, because we can't anticipate all possible future needs. The Semantic Web, in contrast, is more flexible."

The CoABS Grid infrastructure used in this work is based upon Sun’s Jini, but requires only *one* standard interface – the 'AgentRep' which provides inter-agent messaging functionality, with arbitrary message content. It is therefore possible to combine the discovery services of Jini and the messaging, security, logging and other services of the CoABS Grid with machine-understandable semantics by writing messages in emerging Semantic Web languages such as the Resource Description Framework (RDF) and the DARPA Agent Mark-up Language (DAML).

5. Initial work

The aim of this initial work was to implement simple agents using a Semantic Web language to exchange data about the demonstration scenario. Practical experience with RDF and RDF Schema will assist in understanding the issues involved and assessing the current technology.

5.1 Special features of the Coalition domain

There are some differences between the Web and the networks expected in a command information system.

Hendler [24] predicts that on the Web we will not see large complex consistent ontologies, carefully constructed by expert AI researchers, and shared by a great number of users. Rather, we will see a great number of small ontological components largely created of pointers to each other and developed by Web users in much the same way that Web content is currently created. There has been little work so far on developing explicit ontologies for coalition operations; we expect that the situation will improve as benefits from the initial Semantic Web technology are realised.

A coalition network could perhaps be regarded as being 'on the edge of the Web', consisting of multiple fire-walled networks with guarded portals between them, and between them and the Web itself. A greater degree of control will be present. Medium size ontologies would be expected, constructed with care by individual coalition members and groups of members, but not all directly interoperable or consistent with each other. The challenge is to map between them at short notice. In a best-case scenario, many of these ontologies would use standard ontology libraries developed over time for common domains. Complete mapping is not always possible, as noted in the description earlier (section 3.3.2) of mapping data between MBP and CAMPS. There is a need for techniques and tools to handle this, perhaps detecting and flagging the problematic elements for human attention. Hendler [24] notes that there are many possible techniques to map between ontologies, and that this is an interesting challenge for the future.

Some of the initial interest in Semantic Web technology has focussed on the mark-up of conventional web pages with semantic metadata for improved search engines and web-crawling agents. In a coalition system, some data may be utilised in this way, but the main focus is on inter-agent messages expressed directly in RDF or another Semantic Web language. It is important to note that not all agents in a command system will have direct access to the WWW, for obvious security reasons.

5.2 Resource Description Framework (RDF)

Our initial work was based on RDF and RDF Schema (RDFS), following the initial W3C

Recommendation and the release of a number of parsers, APIs and frameworks for RDF. Due to the time scales of this project, it was essential to investigate technologies available now, as well as potential for the future. We regard RDF as an initial test-bed for investigating Semantic Web issues, and a stepping-stone to future technologies.

DAML promises a far greater level of power but is still at an early stage of development as far as tools are concerned. For rapid coalition integration, effective tools are the essential requirement. A powerful and expressive language is not useful in practice until it can be written, modified and applied rapidly, by those without expertise in logic: "A crucial aspect of creating the Semantic Web is to enable users who are not logic experts to create machine-readable Web content" [24].

There is also an interesting correspondence between RDF and DCADM (Defence Command and Army Data Model), the UK Ministry of Defence's preferred – and indeed in many cases, mandated – solution [25]. DCADM is a combination of two technologies. The first is an innovative immutable datastore. In most datastores when a record is modified the old version is overwritten by the new version. In the DCADM immutable datastore, both versions are preserved. This has considerable potential advantages in the sort of Command and Control systems that are envisaged as the primary applications of DCADM. The datastore can support multiple competing values, reflecting the uncertainty factor present in the 'fog of war', and it maintains a complete audit trail of who changed what values and when.

The second component of DCADM is a metadata model that can be used to describe the data models that form the basis for interoperability. In this respect, DCADM and RDF provide essentially equivalent facilities. Data models developed in DCADM are readily translated into RDF, and vice versa.

We have implemented simple agents, running on the CoABS Grid inside KAoS domains, which can store and query RDF databases. They make use of basic RDF Schema ontologies for defining the class and property hierarchy and the range and domain constraints on properties and their values. The entities in the scenario fall into a number of superclasses, such as mobile or fixed objects, natural features or man-made installations, locations, airspaces, or activities. Basic constraints apply: physical objects possess a location; vehicles can have a speed; airfields have runways. RDF Schema cannot however express many other features of the domain; even the fact that friends are disjoint from enemies.

The agents need to acquire information from other agents, and from their own databases. We have therefore implemented a simple RDF query language and query engine with an SQL-like syntax, similar to other recent query languages [26,27]. Most RDF query language proposals appear to be client-server oriented, assuming fast access by the client to a local database in-memory or on disk. However, with a mainly peer-to-peer model, where access via messages over a network may be slow, special features may be required.

As an alternative to returning individual values of properties matched by a query, the query engine can return the sub-graph of triples matched by the query, as a complete RDF document, using a query of the form:

```
select triples where <constraint list>
```

This allows the returned data to be directly parsed and merged into an agent's database. Returning complete RDF documents in this way supplies the context necessary when using asynchronous messaging; if individual context-free values were returned they would have to be somehow matched up with large numbers of outstanding queries.

Queries may be formed which return the sub-graph accessible from a specified resource, so an agent can ask for everything known about a resource in a single query, using the form:

```
select reachable where <constraint list>
```

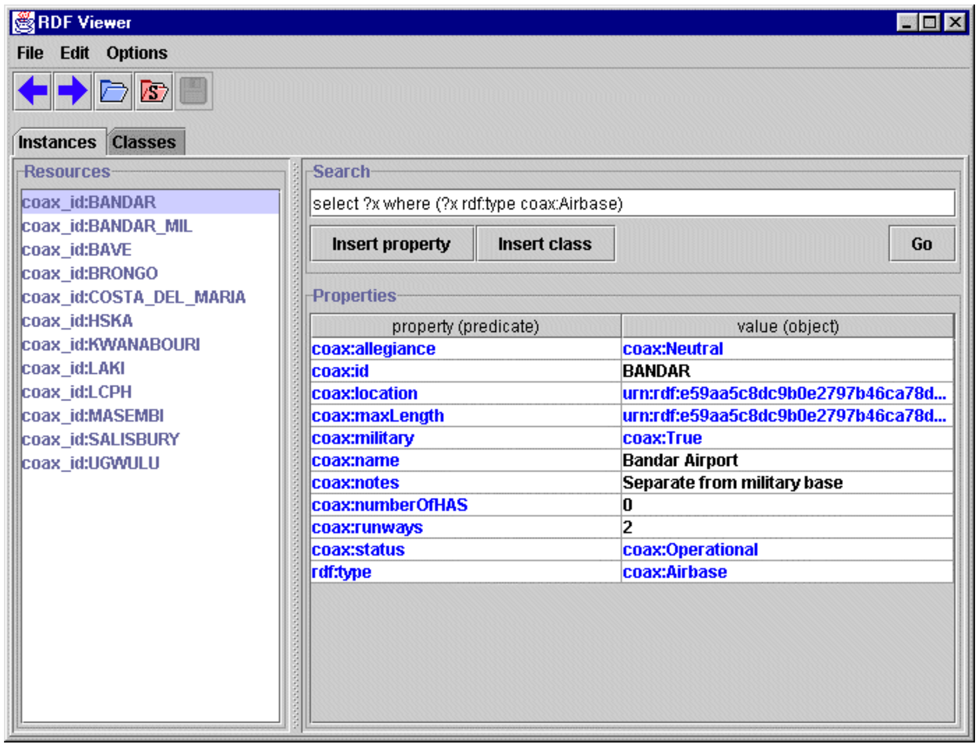
Without this feature, a potentially very large number of queries would be required.

Reading or editing raw RDF syntax is difficult, and rapidly becomes impractical as the number of triples increases. In the process of developing the experimental schemas, a prototype RDF browser has been implemented. This allows the user to seamlessly navigate through both the schema and data, searching for resources either by type or via the query language. All resources are clickable hotlinks, and a history is kept, allowing navigation of an RDF graph in the now-familiar style of a Web browser. Namespaces are automatically abbreviated to namespace prefixes, e.g. 'rdf:type'. In figure 4, scenario data from the CoAX demonstration is being queried for resources of a specified type; the properties of a selected resource can then be viewed. In figure 5, the classes in the corresponding RDF Schema are being examined; super-classes and properties related to the selected class are shown.

This browser was not intended to replace dedicated ontology editors, but to provide tighter support for the specific features of RDF, to handle data and schema seamlessly, and handle large numbers of instances. Some ontology editors do not support features of RDF such as the sub-property hierarchy, global properties, and multiple domains and ranges for properties; in general they cannot handle arbitrary RDF. The aim is to facilitate the creation and examination of models without having to hand-code RDF in XML; there is a lack of mature tools for this at present.

Desirable enhancements to the browser include:

- support for reification and containers (creation, and checking of imported data for all 4 properties of a reification quad; checking and repairing collection properties (_1, _2,...) for correctness; filtering data by origin and timestamp;
- enhanced namespaces support including filtering by namespace;
- explicit support for properties (*subPropertyOf* relationship, range and domain);
- editing support (selection of sub-graphs and multiple nodes; clean deletion of entire objects, collections, and reified statements);
- graphical display of portions of an RDF graph.

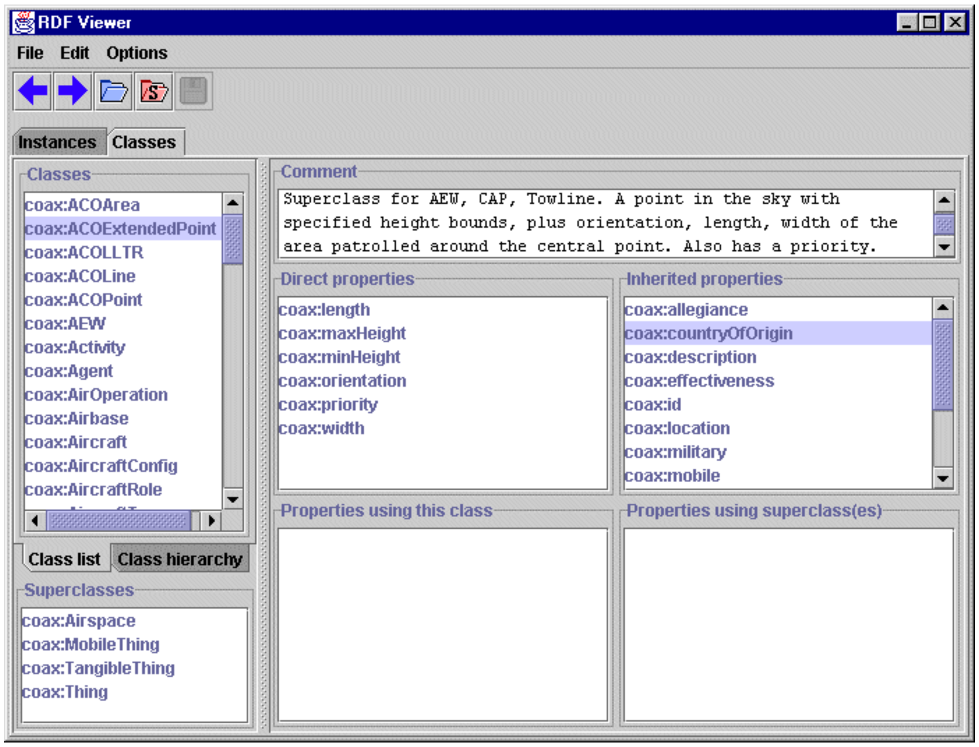


select ?x where (?x rdf:type coax:Airbase)

coax_id:BANDAR
coax_id:BANDAR_MIL
coax_id:BAVE
coax_id:BRONGO
coax_id:COSTA_DEL_MARIA
coax_id:HSKA
coax_id:KWANABOURI
coax_id:LAKI
coax_id:LCPH
coax_id:MASEMBI
coax_id:SALISBURY
coax_id:UGWULU

property (predicate)	value (object)
coax:allegiance	coax:Neutral
coax:id	BANDAR
coax:location	urn:rdf:e59aa5c8dc9b0e2797b46ca78d...
coax:maxLength	urn:rdf:e59aa5c8dc9b0e2797b46ca78d...
coax:military	coax:True
coax:name	Bandar Airport
coax:notes	Separate from military base
coax:numberOfHAS	0
coax:runways	2
coax:status	coax:Operational
rdf:type	coax:Airbase

Figure 4. RDF browser. A search for resources of a specified type has been performed; these resources are listed on the left. All the properties of the selected resource (Bandar Airbase) are shown – these form hotlinks that can be explored further by clicking on them. Literals are not hotlinks, of course; these are shown in black.



Classes:
coax:ACOArea
coax:ACOExtendedPoint
coax:ACOLLTR
coax:ACOLine
coax:ACOPoint
coax:AEW
coax:Activity
coax:Agent
coax:AirOperation
coax:Airbase
coax:Aircraft
coax:AircraftConfig
coax:AircraftRole

Comment:
```
Superclass for AEW, CAP, Towline. A point in the sky with
specified height bounds, plus orientation, length, width of the
area patrolled around the central point. Also has a priority.
```

Direct properties:
coax:length
coax:maxHeight
coax:minHeight
coax:orientation
coax:priority
coax:width

Inherited properties:
coax:allegiance
coax:countryOfOrigin
coax:description
coax:effectiveness
coax:id
coax:location
coax:military
coax:mobile

Superclasses:
coax:Airspace
coax:MobileThing
coax:TangibleThing
coax:Thing

Figure 5. RDF Browser class display, showing superclasses, properties of the class (both direct and inherited) and usage of this class by other properties.

6. Discussion

6.1 Advantages of RDF

RDF provides a formal data model for representing entities (resources) and the relations between them, and a syntax for XML serialisation of data models. RDF describes directed labelled graphs, rather than just labelled trees as in XML. The use of RDF is a move away from inflexible XML data structures and DTDs, allowing more natural handling of partial data, which inevitably occur in uncertain military environments. For example, a system that represents an entity such as a radar installation may store many attributes such as location, allegiance, range, type and so on; yet if a radar is detected electronically at a great distance, we know very little about it and cannot 'fill in all the blanks' of a database form or XML DTD. Rather, data arrive gradually from disparate sources and must be merged together. Partial data records or changes in document structure can be problematic when handling XML, although more recent XML technologies such as XPath and XQuery have increased flexibility to some extent. These advances also counter the size and complexity of code previously required to handle XML trees.

RDF Schema adds a very basic ontological framework, although it appears from the many discussions on the RDF mailing lists that the semantics of RDF and RDFS are unclear in the current specifications [28]. This may be hindering the development of systems with formal semantics on the top of RDF, such as DAML. Very basic inference is possible using the RDFS class and property hierarchy definitions, and property restrictions [24]. This is already used to some extent by current query engines.

Although this seems trivial, it offers a substantial improvement over the keyword searches that Web users suffer at present, and a qualitatively different capability from semantics-free XML data. In many cases, the level of power needed to achieve significant benefits in interoperability is quite small, and so RDF and RDF Schema are useful in themselves. For example, in this coalition scenario, it is easy to search for data about any damaged friendly ground units, without needing to know their exact type. One could also learn their exact types (subclasses of the class *GroundUnit*) in the same query. Simple constraints on properties would also allow queries to specify particular regions in space or time. A flexible query interface to agents creates possibilities for all kinds of useful (and perhaps unforeseen) interactions between systems, in contrast with rigid one-to-one message links.

6.2 Current limitations

The growing industry support for XML points to an XML-based solution for semantic mark-up, yet RDF is verbose and not easily readable or writeable by humans. ‘Notation 3’ [29] is an interesting experiment with a more concise syntax (with additional logic elements). Authoring and processing tools would of course make the syntax less of an issue.

For rapid integration of agents, tools are the emphasis, not languages. A more powerful language is of no use in this domain if it cannot be deployed easily. Tools and APIs are needed throughout the lifecycle of the data – creation, checking, storing, querying and inference, mapping and converting.

Judging from the large volume of debate on the RDF mailing lists, the development of such tools appears to be suffering delays due to the substantial number of unresolved issues concerning the RDF and RDF Schema standards, and their interpretation [28].

A practical problem encountered is that there is an assumption in some APIs and parsers that schemas will always be loaded over the Web from their home site (rather than being cached) which is not realistic on a coalition network.

6.3 Querying requirements

A query language allows agents to seek and exchange information, especially when enhanced by a schema/ontology, but of course we rely on agents understanding the same query language, and there is no standard query language for RDF.

A standard language is of less importance when querying local databases or Web documents; any suitable language could be used. In a community of heterogeneous agents there is a need to talk to other agents in a common tongue, or at least translate down to some 'lowest common denominator'.

A variety of languages have been developed [26, 27, 30–33] with differing capabilities. Many do not yet deal with some of the more controversial aspects of RDF such as reification and collections. Kokkelink [34] has suggested work on standard RDF querying by attempting to re-use previous experience with XML in the development of XPath, XQuery and XSL Transformations.

In the absence of a standard query language, a simple technique for gaining rapid interoperability at the expense of increased messaging load is the use of a remote model interface. The majority of current RDF frameworks allow an RDF graph (model) to be queried using a single triple where each of the subject, predicate and/or object can be a wildcard. This can be regarded as the lowest common denominator for querying. Instead of sending complex queries for a remote agent to process, the work could be done locally, sending multiple messages invoking only the basic query method of the remote agent. As far as the initiating agent is concerned, the process is the same as querying a local (albeit very slow) model database (figure 6). Primitive agents can be made to support complex queries, and agents using differing query languages can be rapidly integrated, although the load on the network will greatly increase.

One could also send a mobile agent implementing complex queries to a remote host, interrogate agents there using low-level messages, and return the results. This could reduce the network load but requires more sophisticated infrastructure, especially for security issues.

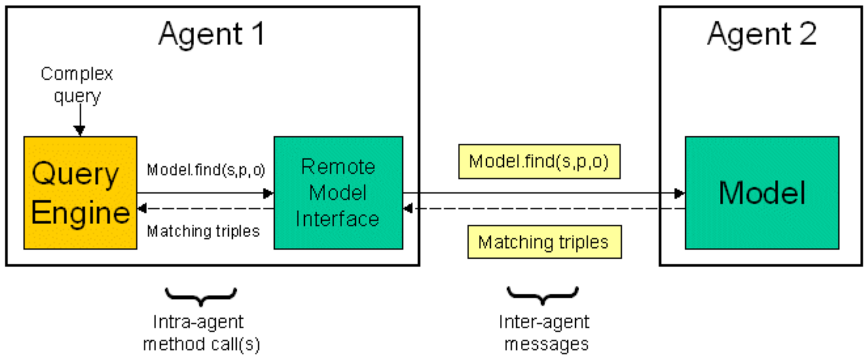


Figure 6. Use of a remote model interface, allowing Agent 1 to perform complex querying of the more primitive Agent 2, which only supports matching of single wild-carded triples.

6.4 Merging models

A fundamental process in a command information system is the collection and fusion of data from many sources. Merging data from multiple RDF sources (agents) into a single model raises several issues. Firstly, the identification of redundant anonymous resources. Anonymous resources are usually assigned locally unique identifiers by the parser; if the data in an RDF model originates from multiple documents, the same (anonymous) resource in different documents will normally receive two different identifiers. For example, figure 7 shows data obtained from three separate sources, which require merging.
Unfortunately, it does not seem possible to determine in general whether the duplication is intended, and therefore whether the redundancy can safely be eliminated. Cardinality restrictions, available in DAML, could help to resolve this issue. For example, in RDF, either of the graphs in figure 8 might be chosen. Can John Smith only have one weight? It might initially appear so, but suppose the data track his weight during a diet! Each weight would then be valid for a particular point in time, and could possess equivalent values in various units.

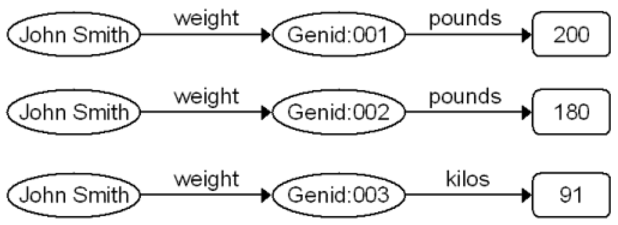


Figure 7. Three RDF data models which we are attempting to merge. The anonymous middle nodes are assigned a locally unique identifier by the parser, but we cannot determine unambiguously whether they are in fact the same resource.

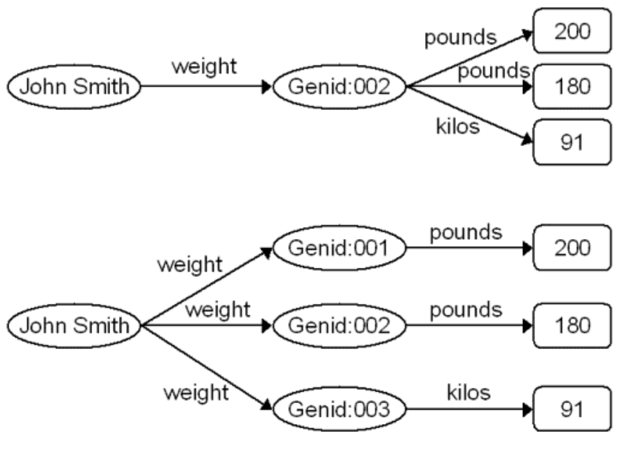


Figure 8. Two possible results of merging the models in figure 7.

Secondly, merging of RDF *Bags* or *Alternatives* cannot be achieved without renumbering all the member attributes (*rdf:_n*) of one of the collections. Merging of *Sequences* does not necessarily make sense in general. Other problems with the RDF containers have been identified [28]; for example, adding an element to an RDF *Bag* requires one to know all the existing elements.

6.5 Attribution

As mentioned earlier, it is valuable for an agent operating in an uncertain environment to store multiple competing values, and maintain an audit trail of changes. As levels of trust and other factors change, the most reliable or up-to-date data can be selected. In a military coalition context this ability is essential.

Reification provides part of the solution; we can reify statements and attach additional data (timestamp, origin, trust values) to them. In the example above, it would then be possible to determine that John Smith weighed 200 pounds (91 Kg) last year but only 180 pounds last week, although the initial value was measured by his doctor and the recent value is his own optimistic claim! However, once the criteria for selecting competing data values have been chosen, we still need to query the data as easily as if it were 'flat'. Most current APIs and query languages do not appear to provide a solution yet. Reification is a contentious issue in RDF with a number of problems identified. The concept of contexts has been introduced as an improvement [28,35].

In practical cases, it may be that no decision can be made automatically; to assist human decisions, there is a need for ways of visualising the changes in data according to time, or source.

6.6 DAML versus RDF(S)

The case for DAML is strongly made in several recent papers [24,36], and centres on DAML's greater expressive power and formal semantics. RDF(S) is not considered to provide sufficient expressive power for many applications.

DAML+OIL, an ontology language based upon description logics and encoded in RDF, provides a rich set of constructs not available in RDF(S). In addition to defining classes and properties, one can express the disjointness or equivalence of classes and a variety of restrictions on the usage of classes and properties, such as cardinality. Additional information can be expressed about properties, for example stating that they are transitive. New classes can be constructed by taking the complement of another class or the intersection of other classes. A Property can be declared as the inverse of other properties. In recent versions, DAML+OIL has also been integrated with the XML Schema datatypes. A logical language, DAML-L, is under development.

There are some incompatibilities between RDF(S) and DAML. Firstly, the semantics of *domain* and *range* are different from those in RDF Schema. Secondly, the RDF Schema specification demands that the subclass relation between classes must be acyclic. DAML+OIL deliberately has no such restriction. Finally, parsers based on the current RDF specification will not support the daml:collection parse type, although a pre-processing stage can overcome this problem. The DAML collections address the previously mentioned problems with RDF Bags (section 6.4).

6.7 What next ?

Initial proof-of-concept contributions to the CoAX experiment will involve replacing some of the XML communication links with RDF to compare the relative complexity and robustness of the code. The simple agents used for this will provide a test-bed for showing the increased flexibility of RDF(S), for example assembling partial data from several sources to form a complete description, and queries assisted by the class hierarchy defined in the schema. Agents supporting a common query language, or a remote model interface, are far more accessible to other agents, greatly increasing the interaction possibilities.

We also wish to explore the use of RDF Reification or Contexts [35] for tagging data with their origin, reliability, timestamp etc. This is obviously of crucial importance in a command information system where it is essential to be able to roll back or exclude false or inaccurate data. There is much scope here for work on tools to visualise the data, providing different views of the data filtered by time, origin, etc.

A further step would be the interoperation of agents without fully shared terminology, via translator agents. There are interesting practical issues in handling the agent conversation policies with interruptions to request translation of terms.

Further steps, such as more expressive ontologies and the use of inference rules, move into the territory of DAML; we expect that emerging DAML tools and the DAML-L logic language will provide us with an opportunity to explore these capabilities. As previously noted, we are collaborating with IHMC on specification and implementation of a DAML-based policy representation (KAoS Policy Language, or KPL), which will used to represent both simple atomic policies (e.g., Java permissions) and complex constructions. Representation of both authorizations (i.e., permitting, permitting with qualification, or forbidding some action) and obligations (i.e., requiring some action to be performed) will be possible. We expect that an initial specification of KPL will be available later this year.

7. Conclusions

Although RDF and RDF Schema have a number of flaws, they are useable for a variety of applications and for agent experimentation. In the context of CoAX, for example, they allow richer interactions between agents, and more useful and complex queries. Thus, RDF and RDF Schema still offer interesting possibilities and can contribute significantly to agent interoperability in a coalition setting. Alongside the development of languages, tools and agents, expertise in constructing ontologies for coalition operations will be needed. Some of the flaws will be addressed by the emerging DAML family of languages; others relate to the RDF data model and syntax; these need to be addressed. There is also a profusion of RDF query languages, although work toward a standard has been proposed.

DAML adds the formal semantics and expressiveness demanded by the logic community, but is incomplete as yet; we look forward to future developments, particularly in the area of tools for creating, parsing, checking, querying and reasoning with the languages.

One of the greatest challenges ahead lies in the development of mechanisms for merging or mapping between ontologies automatically, as far as possible, and fusing data from disparate sources, enabling heterogeneous agent systems to interoperate rapidly and effectively. It is important to acknowledge that any fundamental differences between systems (such as described in section 3.2.2 will always be a barrier to complete interoperability.

Acknowledgements

QinetiQ work was carried out as part of the Technology Group 10 of the UK Ministry of Defence Corporate Research Programme. We acknowledge the contributions of the following CoAX partners:

- Ariadne web wrapper/mediator (University of Southern California, Information Sciences Institute)
- Consolidated Air Mobility Planning System (CAMPS) (Global InfoTek, Inc / BBN Technologies / Air Force Research Laboratories).
- Knowledgeable Agent-oriented System (KAoS) Policy-Based Tools and Framework (University of West Florida, Institute for Human and Machine Cognition / Boeing).
- EMAA / CAST agents (Lockheed Martin Advanced Technology Laboratories).
- NOMADS Mobile Agent System (University of West Florida, Institute for Human and Machine Cognition).
- Process and Task Management tools (University of Edinburgh, Artificial Intelligence Applications Institute).
- The DARPA CoABS Grid (Global InfoTek Inc, ISX Corporation).

We would like to thank our QinetiQ colleagues Chris Walker and Don Brealey for provision of the Master Battle Planner (MBP) tool for Air Battle Planning, and Sergey Melnik of Stanford University for provision of RDF API software.

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing official policies or endorsements, either express or implied, of the UK MoD, DARPA, the Air Force Research Laboratory, the US Government, the University of Edinburgh or the University of West Florida.

References

[1] Coalition Agents eXperiment: http://www.aiai.ed.ac.uk/project/coax/

[2] David Allsopp, Patrick Beautement, Jeffrey M. Bradshaw, John Carson, Michael Kirton, Niranjan Suri and Austin Tate, *Software agents as facilitators of coherent coalition operations*, 6th International Command and Control Research and Technology Symposium, 19-21 June 2001, US Naval Academy, Annapolis, MD, USA.

[3] DARPA Agent Mark-up Language: http://www.daml.org/

[4] R. A. Rathmell, *A Coalition Force Scenario 'Binni – Gateway to the Golden Bowl of Africa*', Proceedings of the International Workshop on Knowledge-Bases Planning for Coalition Forces, (ed. A. Tate) pp. 115-125, Edinburgh, Scotland, 10-11 May 1999.

[5] Nicholas R. Jennings, *An agent-based approach for building complex software systems*, Communications of the ACM, April 2001/Vol. 44, No. 4, pp 35-41.

[6] CoABS Grid: http://coabs.globalinfotek.com/

[7] Jeffrey M. Bradshaw, Niranjan Suri, Martha Kahn, Phil Sage, Doyle Weishar and Renia Jeffers, *Terraforming Cyberspace: Toward a policy-based grid infrastructure for secure, scalable, and robust execution of Java-based multi-agent systems*, Proceedings of the Workshop on Agent-based Cluster and Grid Computing, IEEE International Symposium on Cluster Computing and the Grid, Brisbane, Australia, 14-18 May, 2001. (Enlarged version in IEEE Computer, July 2001, pp 48-56).

[8] Jini: http://www.sun.com/jini/

[9] Jeffrey M. Bradshaw, Stewart Dutfield, Pete Benoit and John D. Woolley, *KAoS: Toward an industrial-strength generic agent architecture*, In J. M. Bradshaw (Ed.), *Software Agents*, 1997*,* pp. 375-418, Cambridge, MA: AAAI Press/The MIT Press.

[10] Jeffrey M. Bradshaw, Mark Greaves, Heather Holmback, Wayne Jansen, Tom Karygiannis, Barry Silverman, Niranjan Suri, and Alex Wong, *Agents for the masses?* In J. Hendler (Ed.) Special issue on agent technology, IEEE Intelligent Systems, March/April 1999, 53-63.

[11] JavaSpaces: http://www.sun.com/jini/specs/

[12] Mark Burstein, and Douglas Smith, *A Portable, Interactive Transportation Scheduling Tool Using a Search Engine Generated from Formal Specifications*, Proceedings of the Third International Conference on Artificial Intelligence Planning Systems, B. Drabble (ed.), The AAAI Press, Menlo Park, CA, May, 1996 ISBN 0-929280-97-0.

[13] Thomas Emerson and Mark Burstein, *Development of a Constraint-based Airlift Scheduler by Program Synthesis from Formal Specifications*, Proceedings of the 1999 Conference on Automated Software Engineering, Orlando, FL, September, 1999.

[14] Mark Burstein, George Ferguson, and James Allen*, Integrating Agent-Based Mixed-Initiative Control with an Existing Multi-Agent Planning System*, Proceedings of the Fourth International Conference on MultiAgent Systems, Boston, MA, 2000.

[15] Craig A. Knoblock and Steven Minton, *The ariadne approach to web-based information integration*, IEEE Intelligent Systems, 13(5), September/October 1998.

[16] Ariadne: http://www.isi.edu/info-agents/ariadne/

[17] XSL Transformations: http://www.w3.org/Style/XSL/

[18] AIAI's IX Technology: http://www.aiai.ed.ac.uk/project/ix/

[19] N. Suri, J. M. Bradshaw, M. R. Breedy, P. T. Groth, G. A. Hill and R. Jeffers, *Strong mobility and fine-grained resource control in NOMADS*, Proceedings of the 2nd International Symposium on Agents Systems and Applications and the 4th International Symposium on Mobile Agents (ASA/MA 2000), Zurich, Switzerland; Berlin: Springer-Verlag

[20] Michigan MCA: http://ai.eecs.umich.edu/people/durfee/COABS/

[21] Dartmouth College: http://actcomm.dartmouth.edu/

[22] OBJS eGents: http://www.objs.com/agility/

[23] Tim Berners-Lee, James Hendler and Ora Lassila, *The Semantic Web*, Scientific American, May 2001 http://www.scientificamerican.com/2001/0501issue/0501berners-lee.html

[24] James Hendler, *Agents and the Semantic Web*, IEEE Intelligent Systems, vol. 16, no. 2, March/April 2001, pp. 30-37. http://www.cs.umd.edu/~hendler/AgentWeb.html

[25] Robert Andrews, *Data Interoperability with DCADM and XML*, DERA report DERA/CIS/CIS3/TR000265/1.0, March 2000, Defence Evaluation & Research Agency (now QinetiQ Ltd), St. Andrews Road, Malvern, UK.

[26] RDFDB: http://web1.guha.com/rdfdb/query.html#query

[27] Squish: http://swordfish.rdfweb.org/rdfquery/

[28] RDF issue tracking: http://www.w3.org/2000/03/rdf-tracking

[29] Notation 3: http://www.w3.org/DesignIssues/Notation3.html

[30] RQL: http://139.91.183.30:9090/RDF/

[31] RDFQL: http://www.intellidimension.com/RDFQLmanual.html

[32] RDFQuery: http://www.w3.org/TandS/QL/QL98/pp/rdfquery.html

[33] XWMF: http://nestroy.wi-inf.uni-essen.de/xwmf/

[34] RDFPath: http://zoe.mathematik.Uni-Osnabrueck.DE/QAT/

[35] RDF Contexts: http://public.research.mimesweeper.com/RDF/RDFContexts.html

[36] Sheila A. McIlraith, Tran Cao Son, and Honglei Zeng, *Semantic Web Services,* IEEE Intelligent Systems, vol. 16, no. 2, March/April 2001, pp. 46-53.