Agent-based Semantic Web Services

Nicholas Gibbins *, Stephen Harris, Nigel Shadbolt

Department of Electronics and Computer Science, University of Southampton, Highfield, Southampton, SO17 1BJ, UK

Abstract

The Web Services world consists of loosely-coupled distributed systems which adapt to changes by the use of service descriptions that enable ad-hoc, opportunistic service discovery and reuse. At present, these service descriptions are semantically impoverished, being concerned with describing the functional signature of the services rather than characterising their meaning. In the Semantic Web community, the DAML Services effort attempts to rectify this by providing a more expressive way of describing Web services using ontologies. However, this approach does not separate the domain-neutral communicative intent of a message (considered in terms of speech acts) from its domain-specific content, unlike similar developments from the multi-agent systems community.

We describe our experiences of designing and building an ontologically motivated Web Services system for situational awareness and information triage in a simulated humanitarian aid scenario. In particular, we discuss the merits of using techniques from the multi-agent systems community for separating the intentional force of messages from their content, and the implementation of these techniques within the DAML Services model.

Key words: Web Services, Semantic Web, DAML-S, Agent Communication Languages, Ontologies

1 Introduction

The world of Web Services may be characterised as a world of heterogeneous and loosely-coupled distributed systems where adaptivity to ad-hoc changes

---

* Corresponding author.
  Email addresses: nmg@ecs.soton.ac.uk (Nicholas Gibbins),
swh@ecs.soton.ac.uk (Stephen Harris), nrs@ecs.soton.ac.uk (Nigel Shadbolt).

in the services offered by the components of the systems is considered advantageous. By loosely-coupled, we mean that the interactions between system components are not rigidly specified at design time, but that system components may opportunistically make use of new services that become available during their lifetime without having been explicitly told of their existence from the outset.

The task of searching for a system component which can perform some given service, or *service discovery*, is the enabling technique that makes loosely-coupled systems possible, and provides a process by which system components may find out about new services on offer. An essential adjunct to service discovery is *service description*, by which names or descriptive expressions are attached to services, allowing both the advertisement of services by providers and the formulation of queries about services by users. Service discovery services (in which we include *matchmakers*) have been seen as an essential component of loosely coupled systems such as (but not limited to) multi-agent systems [1,2].

A typical service discovery service (also often referred to as a *directory service*) consists of a registry (possibly distributed) which provides two services. The first allows service providers to advertise the services that they offer in the registry, while the second enables service users to query the registry and so determine which service providers can provide relevant services.

One rough characterisation of the technologies used for service discovery in the Web Services world can be made by studying the difference between approaches which could be considered semantically poor and those which are semantically rich. In the former case, services are often referred to by opaque names or function signatures which give little or no indication of the nature of the services being managed. In the latter, however, service descriptions are more complex expressions which are based on terms from agreed vocabularies, and which attempt to describe the meaning of the service, rather than simply ascribing a name to it. For example, a semantically poor description of a service might be that it takes a string as input and returns two integers, while a semantically rich description could be that it takes the name of a football team and returns the score from their last match.

A key component in the semantics-rich approach is the *ontology*, the formal, agreed vocabulary whose terms are used in the construction of service descriptions. An ontology is a conceptualisation of an application domain in a human-understandable and machine-readable form, and typically comprises the classes of entities, relations between entities and the axioms which apply to the entities which exist in that domain. Ontologies are currently a fast-growing research topic, with interest from several communities, not least the agent-based computing, Semantic Web and knowledge management commu-

nities, because they offer a more formal basis for characterising the knowledge assets held by software agents, Semantic Web services or organisations [3,4].

Although such an ontology defines the agreed meaning for the application domain-specific terms used in the content of messages, it does not define the meaning of the message types themselves, or their effects upon the recipient. The current approach in the Semantic Web to Web Services, such as that taken by DAML Services, does not provide a common basis for defining the pragmatics of different message types, as we might expect from a speech act-like treatment of messages [5]. Such a basis would provide a way to ease the introduction of new types of messages, since there would be a common understanding of what was meant by, for example, a directive message (which instructs a system component to perform an action) or an assertive message (which informs a system component of some fact) which was independent of any domain specific meaning.

The technique of factoring out the common attributes of message types and ascribing them to different classes of speech acts or performatives is commonly used in the design of *agent communication languages* (ACL) for multiagent systems [6,7], where there is a clear separation made between the domain-specific and domain-independent aspects of communication. For example, a message which asks a hypothetical meteorological agent for the weather forecast for a certain region may be divided into two distinct parts. The domain-independent part identifies this message as belonging to a particular class of utterances (or speech acts), in this case a directive (a question is a request that the recipient inform the sender of some fact). The domain-specific part indicates that this message is concerned with weather reports for a certain region, and is expressed in an ontology specific to the application domain.

Given that the multi-agent systems and Web Services communities hold similar aspirations with respect to loose coupling, we believe that an approach in which an ACL component is integrated into semantically rich service descriptions can be applied to Web Services. Such an approach could have benefits beyond the immediate Web Services community, given the interest in Web Services technologies in the Grid Computing community [8,9]. There are a number of other studies on the integration of multiagent systems with Web Services, but these are predominantly concerned with enabling the agents in existing systems to use, provide or broker Web services [10,11], rather than attempting to adapt a core agent technology into the Web Services infrastructure.

In this paper we outline our experiences of building semantically rich Web Services based on the integration of ontologically-motivated DAML-S-based Web Services and an agent communications language, and describe our prototype demonstrator, a situational awareness application based on a humanitarian

aid scenario.

2 Semantic Web Services

The aim of this work has been to investigate the integration of the nascent Web Services infrastructure with the richer semantics of the Semantic Web, in particular through the use of more expressive languages for service description. In their implementation of service descriptions, the existing Web Services specifications are more concerned with the signature of services. Such signatures comprise the types of the parameters of the service (typically expressed in terms of XML Schema datatypes), rather than with any form of ontological classification of the services.

The notion of an ontology is central to the Semantic Web, which uses languages such as RDF Schema [12] or DAML+OIL [13] (or in future, the Web Ontology Language OWL, a current work in progress) to describe ontologies. An integration of Web Services with the Semantic Web should involve the use of these languages to describe and characterise services in a manner which the existing Web Services service description languages cannot.

There are two options for the form of this integration. We could choose to layer RDF or DAML+OIL on top of an existing XML-based service description language (such as the Web Services Description Language or WSDL [14], for example), so that the description includes an RDF expression that characterises the service. Alternatively, we could choose to build on a service description language which is itself written in RDF or DAML+OIL, such as the DAML Services ontology [15].

We have chosen the latter approach, and have used DAML Services as the basis for our design because it allows the definition of classes of related services, which makes service reuse more feasible (because agents are better able to reason about the relationships between services) and the system more adaptable as a whole (because rich service descriptions give agents the means to determine whether they can use new types of service).

In addition to reuse and adaptability concerns, DAML Services also allows the types of service parameters to be specified as DAML class expressions, in addition to the XML Schema datatypes [16] that are used by WSDL and other Web Services languages, so the parameter values that are passed when a service is invoked may be objects from a knowledge base as well as literal values.

At the time this work was carried out, DAML Services was (and to some

4

extent still is) a 'moving target'. Our description of agent-based semantic web services is based on version 0.7 [15], and predates the introduction of the OWL vocabulary for service description introduced in version 0.9.

3   Agent Web Services

In the conventional Web Services approach exemplified by WSDL [14] or even by DAML Services, the communicative intent of a message (for example, whether it is a request or an assertion) is not separated from the application domain. This is at odds with the convention from the Multi-Agent Systems world, where there is a clear separation between the intent of a message, which is expressed using an agent communication language, and the application domain of the message, which is expressed in the content of the message by means of domain-specific ontologies.

This separation between intent and domain is beneficial because it reduces the brittleness of a system. If the characterisation of the application domain (the ontology) changes, then only that component which deals with the domain-specific information need change; the agent communication language component remains unchanged. In addition, the domain-neutral performatives in an ACL may be combined to form common patterns of interaction (protocols, in effect) such as contract nets, markets or auctions which enable the behaviour of a system to be considered in more abstract terms (a limited form of this protocol-level description is also possible in the emergent Web Services choreography languages, but the resulting protocols are not domain-neutral as they are here).

The use of agent communication languages to describe Web services has a parallel with systems for the description and brokering of *problem solving methods* (PSMs) from the knowledge acquisition community, an example of which is the IBROW3 system [17]. IBROW3 makes a similar distinction between domain-specific and -independent characterisations of reasoning services by separating the description of the particular problem solving method used by a reasoner (expressed in the UPML language [18]) from the application domain-specific ontology to which the PSM is to be applied.

The division of service descriptions into a profile and a process component, as in DAML Services, provides a means to compartmentalise Web Services in a manner similar to that found in agent systems. We describe the pragmatics of message types in the process component, giving an abstract ontology of message types that corresponds to the agent communication language, while the more application-specific details of the abilities of a particular agent (expressed as constraints on the content of messages) are expressed in the profile

service description	
domain-specific profile	FIPA ACL process ontology
profile	process
DAML Services	

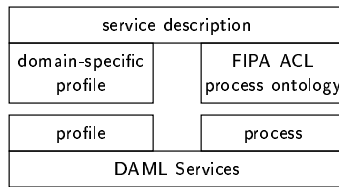Fig. 1. Service Description with ACL Process Ontology

```
<daml:Class rdf:ID="Query-Ref">
 <rdfs:subClassOf ref:resource="&damlsproc;#AtomicProcess"/>
 <rdfs:comment>The action of asking another agent for the object
  referred to by a referential expression. Query-ref is the act of
  asking another agent to return the object idientified by a
  descriptor.</rdfs:comment>
</daml:Class>

<rdf:Property rdf:ID="query-ref-descriptor">
 <rdfs:subPropertyOf rdf:resource="&damlsproc;input"/>
 <rdfs:domain rdf:resource="&fipa;Query-Ref"/>
 <rdfs:range rdf:resource="&daml;Thing"/>
</rdf:Property>

<rdf:Property rdf:ID="query-ref-response">
 <rdfs:subPropertyOf rdf:resource="&damlsproc;output"/>
 <rdfs:domain rdf:resource="&fipa;Query-Ref"/>
 <rdfs:range rdf:resource="&daml;List"/>
</rdf:Property>
```

Fig. 2. FIPA ACL Process Ontology Fragment

component, as shown in Figure 1.

To this end, we have designed a simple process ontology of message types based on the FIPA agent communication language [6]. In this ontology, ACL message types are represented as atomic processes (see Figure 2 for a fragment of this ontology containing the query-ref performative), with the content of the message as a parameter of the process. The Query-Ref process has two properties, one a sub-property of input which is used to pass the query expression that forms the content of the message, and one a sub-property of output which is used to return the entities which satisfy the query.

In addition to input and output parameters, DAML Services also provides a facility for specifying the necessary preconditions and the side-effects of a service. While a full description of the FIPA performatives could make use of this facility to fully describe the pragmatics of the messages (as described by FIPA in the appendix to [6]), the facility is not yet fully specified. Likewise, the expression in DAML of the FIPA theory of agency and the mentalistic stance it adopts (a necessary prerequisite for expressing the pragmatics of individual performatives) is not the focus of this work.

Another important difference between multi-agent systems and Web Services concerns the manner of their communications with respect to synchrony. In FIPA, the performatives (query-if and query-ref, for example) are treated as asynchronous messages. A query does not return an answer directly, but causes the formation of an intention in the recipient to send an inform message (containing the answer) to the sender of the query. These message types may be combined to form more complex interaction patterns, such as the

6

FIPA Request protocol, but the basic notion is that agents communicate as peers, with all agents able to both send and receive messages. This message-passing idiom is at odds with the predominant communication idiom found in the Web Services environment (procedure-calling, effectively synchronous) as exemplified by the Simple Object Access Protocol [19]. There are exceptions to this in the Web Services choreography literature, such as the Web Services Conversation Language [20], but this models interactions at a higher level, rather than at the level of the messaging protocol itself.

In our adaptation of the FIPA ACL for a procedure-calling Web Services environment, we have chosen to amend the semantics of the query performatives and make them synchronous messages which return the answer to the query (this decision was made necessary by the lack of suitable facilities for service choreography in the DAML-S ontology). An advantage of this approach is that we no longer need to track the conversations in which a service is participating (in order to determine which response message corresponds to which query) because a response message cannot be separated from the the query to which it is providing an answer. This has the effect of simplifying the service's implementation of the ACL (at the expense of protocol descriptions), and allows us to concentrate instead on the service profiles which are used to determine whether or not a service will be of use to us.

From our experiences of using an agent communication language characterisation of web service process descriptions, we have come to similar conclusions to those drawn in [21]. By itself, the underlying RPC-like invocation model assumed by Web Services is insufficient to express the sort of interactions found in multiagent systems. In particular, while the DAML Services model allows a service provider to specify the effects of invoking a service (as part of the service profile, in addition to its inputs, outputs and preconditions), there is no standard way to state that the effect of a particular service invocation is that another service is invoked (as would be the case in even a simple FIPA protocol, such as FIPA Request). The ability to represent a chain of service invocations is not covered by the process composition features of the DAML Services process model; the request service invocation and the response service invocation are both atomic processes, but the notion of a composite process defined in DAML Services does not cover the relation between them.

The profile component of the DAML Services expression is used to express the service being offered or requested. This profile description defines the parameters of the service, cross-referenced to the corresponding parameters of the process of which the service is an instantiation. In conventional DAML Services usage, the parameter type restriction in a service's profile (expressed using the `restrictedTo` property) should be consistent with the range of the parameter properties on the process on which the service is based, but there is no logical constraint expressed within DAML-S which requires this.

```
<profile:OfferedService rdf:ID="UNHCR-Query-Ref-Profile">
 <profile:has_process rdf:resource="&fipa;Query-Ref"/>
 <profile:input>
  <profile:ParameterDescription>
   <profile:parameterName rdf:resource="query-ref-descriptor"/>
   <profile:refersTo rdf:resource="&fipa;query-ref-descriptor"/>
   <profile:restrictedTo>
    <daml:Class>
     <daml:intersectionOf rdf:parseType="daml:collection">
      <daml:Class rdf:about="&flood;Report"/>
      <daml:Restriction>
       <daml:onProperty rdf:resource="&flood;reportsOn"/>
       <daml:toClass>
        <daml:Class>
         <daml:intersectionOf rdf:parseType="daml:collection">
          <daml:Class rdf:about="&flood;MovementEvent"/>
          <daml:Restriction>
           <daml:onProperty rdf:resource="&flood;actor"/>
           <daml:toClass>
            <daml:Class>
             <daml:intersectionOf rdf:parseType="daml:collection">
              <daml:Class rdf:about="&flood;Vehicle"/>
              <daml:Restriction>
               <daml:onProperty rdf:resource="&flood;memberOf"/>
               <daml:hasValue rdf:resource="&flood;UNHCR"/>
              </daml:Restriction>
             </daml:intersectionOf>
            </daml:Class>
           </daml:toClass>
          </daml:Restriction>
         </daml:intersectionOf>
        </daml:Class>
       </daml:toClass>
      </daml:Restriction>
     </rdfs:intersectionOf>
    </daml:Class>
   </profile:restrictedTo>
  </profile:ParameterDescription>
 </profile:input>
 <profile:output>
  <profile:ParameterDescription>
   <profile:parameterName rdf:resource="query-ref-response"/>
   <profile:refersTo rdf:resource="&fipa;query-ref-response"/>
   <profile:restrictedTo ref:resource="&daml;List"/>
  </profile:ParameterDescription>
 </profile:output>
</profile:OfferedService>
```

Fig. 3. Sample Profile

We have adapted this usage so that the range of the process parameter is a
superclass of the profile parameter restriction. The process therefore gives an
abstract, domain-neutral description of the ACL performative which charac-
terises the service, and the profile gives a more domain-specific description of
the service which constrains the parameter type. For example, Figure 3 gives
the profile of a service from our situational awareness system for a simulated
humanitarian aid scenario (see Section 5 for further details of this system).
This service allows an agent to ask queries about reports on UNHCR vehicle
movements (terms from the humanitarian aid domain ontology are indicated
by the use of the flood namespace). The restriction on the input parameter of
this service is the class of reports about the movements of vehicles belonging to
the UNHCR, which is a subclass of the range of the corresponding parameter
on the abstract Query-Ref process (Thing, the most general class) as shown
in Figure 2.

The profile in Figure 3 is of type OfferedService, indicating that this is a
service advertisement; an agent requesting a service would construct a service
profile of type NeededService. This use of offered and needed services allows
service brokers to support interactions that are driven both by the clients

8

```
<damls:Service rdf:ID="UNHCR-Subscribe">
 <!-- Reference to the UNHCR-Subscribe Profile -->
 <damls:presents rdf:resource="&flood;UNHCR-Subscribe-Profile"/>
 <!-- Reference to the FIPA Subscribe Process Model -->
 <damls:describedBy rdf:resource="&fipa;#Subscribe"/>
 <!-- Reference to specific grounding for this service -->
 <damls:supports rdf:resource="..."/>
</damls:Service>
```

Fig. 4. Sample Service

(those requesting services) and the services (those providing services).

The process and profile components of the service description are referenced together in a top-level service description (see Figure 4) which also includes a reference to the means which is to be used to access the service, known as the *service grounding*. In our example, we have chosen to omit the details of the grounding because they are not directly relevant to this work. The `supports` property references a service grounding, expressed in DAML-S, which relates the description of this service to a WSDL [14] description of the service which contains the specific information required to invoke the service (protocol, port and so on).

At the time at which this work was carried out, this area of DAML Services was still largely undefined, and there was no standard vocabulary for grounding DAML-S services using WSDL. This has subsequently been addressed in the most recent version of the DAML-S specification, but these additions to the specifications have not yet been reflected in our software.

In Figure 5, we have constructed a simple SOAP message (using the FIPA `inform` performative) which contains a report from a UNHCR vehicle (about itself) that is moving with a certain bearing and speed from a certain location. It should be stressed that the manner in which we have written this message is the result of an informed guess as to how one would pass RDF fragments as parameters to Web Services, and as to how a service specified by DAML Services might be grounded in SOAP.

4 Query Language

When the service in the `Subscribe` example is invoked, the value of the input parameter should be an instance of the class restriction which is given as the input parameter types in both the profile and the process descriptions. For the various query performatives (`query-if`, `query-ref` and `subscribe`), this input parameter contains the query expression which would be contained in the message content in a conventional agent-based system. However, there is as yet no standard query language for RDF, DAML+OIL or OWL, although there are several under development, including DAML Rules [22] (which builds on DAML+OIL and expresses queries as Horn clause-like structures), the DAML

9

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env="http://www.w3.org/2001/12/soap-envelope">
 <env:Body>
  <fipa:inform-proposition xmlns:fipa="http://www.fipa.org/ontology/acl#">
   <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
            xmlns:flood="http://www.example.org/ontology/flood#">
    <flood:Report rdf:about="">
     <flood:reportsOn>
      <flood:MovementEvent>
       <flood:actor>
        <flood:Vehicle rdf:about="&flood;UNHCR-2323">
         <flood:memberOf rdf:resource="&flood;UNHCR"/>
        </flood:Vehicle>
       </flood:actor>
       <flood:headingTowards>
        <flood:Direction>
         <flood:bearing>284.5</flood:bearing>
         <flood:velocity>43</flood:velocity>
        </flood:Direction>
       </flood:headingTowards>
       <flood:locatedAt>
        <flood:Location>
         <flood:longditude>32.23427</flood:longditude>
         <flood:latitude>16.33871</flood:latitude>
        </flood:Location>
       </flood:locatedAt>
       <flood:occursAt>2002-04-12T12:23:48</flood:occursAt>
      </flood:MovementEvent>
     </flood:reportsOn>
     <flood:reporter rdf:resource="&flood;UNHCR-2323"/>
     <flood:certainty>1.0</flood:certainty>
     <flood:occursAt>2002-04-12T12:23:48</flood:occursAt>
    </flood:Report>
   </rdf:RDF>
  </fipa:inform-proposition>
 </env:Body>
</env:Envelope>
```

Fig. 5. Sample SOAP message

Query Language [23], RDQL [24] and SeRQL [25].

Due to this lack of any standard format for expressing queries, we have chosen to express queries as anonymous resources, also known as blank nodes or bNodes. These are instances which are not identified by a URI, but by the values of their properties. As a result, they can be considered to be existentially quantified query expressions which denote objects of interest. In effect, the query is a template subgraph which is matched against the RDF graph, with the query solutions being the locations where the template matches. The resource at the root of the RDF/XML document fragment (possibly a bNode) is considered to be the object of the query. This allows us to express the capabilities of a service by describing the class of queries which may be asked of that service. The class expression given as the input parameter type in the profile should contain as instances all the possible queries (expressed using the bNode technique) that may be asked of the service.

The main limitation of this approach is that it is only applicable to queries of a certain structure. The item about which the query is phrased (the anonymous resource) must be the subject of the RDF triples, not the object, if more than one property of the resource is to be specified (this is largely a limitation brought about by the RDF syntax [26]). This is a significant limitation, but also one which can be mitigated against by a suitable design of the domain ontology in which the objects which are most likely to be the subject of queries appear as the subjects of RDF triples rather than as the objects (see Section 5.2 and Figures 8 and 9).

10

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<!DOCTYPE rdf:RDF [
 <!ENTITY rdf 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
 <!ENTITY flood 'http://example.org/ontology/flood#'>
]>
<rdf:RDF xmlns:rdf="&rdf;" xmlns:flood="&flood;">
 <flood:Report>
  <flood:reportsOn>
   <flood:MovementEvent>
    <flood:actor>
     <flood:Vehicle>
      <flood:memberOf rdf:resource="&flood;UNHCR"/>
     </flood:Vehicle>
    </flood:actor>
   </flood:MovementEvent>
  <flood:reportsOn>
 </flood:Report>
</rdf:RDF>
```

Fig. 6. Sample Query

As an example, the domain ontology which we have designed for this application is centred around events and reports of events. We have taken the approach that communication in the system will be about these events and reports (rather than about any persistent world state which the reports might suggest), so the queries can be expressed using the anonymous resource technique by specifying the properties that the report (and the event it contains) must possess. It should be noted, however, that we did not specifically design the ontology in this report to circumvent the expressive limitations of our chosen query language, but rather that the query language was chosen because it was appropriate for use with the domain ontology that we had already designed.

In Figure 6 we show an RDF fragment which expresses the notion that there exists some report which reports on the movements of vehicles owned by the UNHCR; this may be interpreted as a query about reports with those properties.

An additional limitation of this approach to query construction, which is unfortunately also shared with several of the other query languages currently under development, is that it is not possible to specify literal ranges in queries. An example of such a literal range might be a query of movement reports about entities that were north of a particular point, or to put it a different way, whose latitude was greater than a certain value. This limitation arises because the RDF and DAML+OIL models have no notion of how different literal datatypes behave (particularly with respect to ordering and inequalities). The most likely solution to this problem is based on the use of *oracles*, entities which have specific knowledge of the behaviour of different literal types (integers, latitude/longitude pairs, dates, etc.) and which may be used by inference and query engines to evaluate tests based on those types.

At present, the development and standardisation of query languages for the Semantic Web is largely immature. As Semantic Web development in general becomes more mature, we expect that the current Burgess Shale-like diversity of query languages will come to a close with standardisation on a small number

of languages. The investigation and design of suitable query languages for use with the style of agentified Web Services that we discuss in this paper therefore remains an open direction for future research.

5 A Prototype Agent Web Services System

As a proof of concept of the technologies discussed above, we have designed a system which demonstrates the use of Agent Web Services in the application domain of situational awareness in a humanitarian relief scenario.

The scenario for this study is set in a river delta region which has experienced flooding due to unseasonally heavy rainfall. The people who have been displaced from their homes by the flooding are being sheltered in relief camps. The timeline for the scenario includes a rapid flooding event which forces the creation of new relief camps, and a hostile event upon a relief convoy which requires military intervention and support.

The system contains a number of agents which generate reports on the state of the world (e.g., refugee movements, meteorological reports and forecasts) with differing degrees of certainty. A feature of this scenario is that it includes a number of different types of user, each of which has different information needs, and so each of which should be sent a different subset of the reports generated by the entities in the system. The aim of this system is the provision of filtered report streams to these users in a timely manner, a process often referred to as *information triage*.

In addition, the sets of agents which produce and consume reports are not static; agents may join and leave the system while it is running. The requirement for the system both to adapt to the loss of agents, and to opportunistically integrate new agents provides a motivation for the semantically richer service descriptions that were discussed earlier in this paper.

5.1 System Architecture

In our system architecture, illustrated in Figure 7, the flow of information is from left to right. On the left are various data sources which correspond to entities in the domain environment and generate streams of reports about different types of events, while on the right are consumers which take the reports and present them to the user (the map panel), or which perform some further processing on the reports.

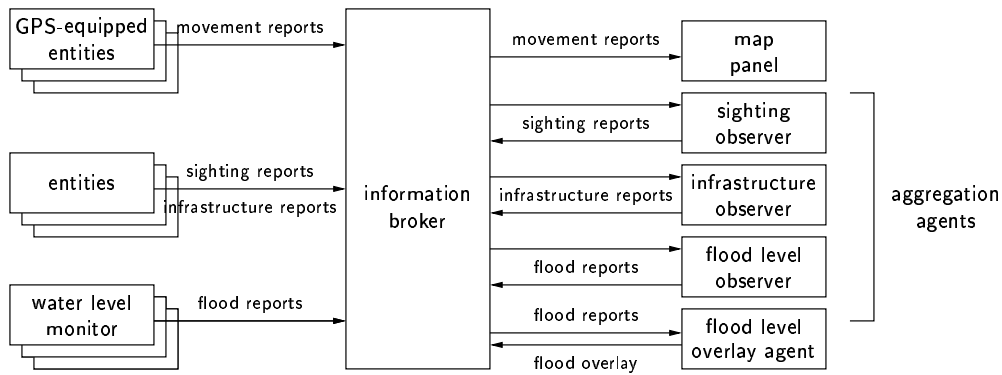The key component of the system is the central broker which mediates the

Fig. 7. System architecture

interaction between the other system components. Data consumers use the broker to find sources which can satisfy their information needs, which typically depend on the user view which is being presented, by registering a service requirement (expressed as a `NeededService` in the DAML Services profile model) with the broker. Similarly, the data sources register a service advertisement describing their capabilities with the broker (expressed as a `OfferedService` in the DAML Services profile model). The broker compares the service requirement to the service advertisements that it has received from the data sources and responds with the matching services. The consumers then communicate directly with the data sources, typically by formulating a subscription to some subset of the reports that the source offers, as was illustrated in Figure 3.

At present, the broker matches advertisements to requirements in a naive fashion based on the types of the parameters in the profile. The parameter restrictions in `NeededService` and `OfferedService` are translated into triple patterns (we implement only a reduced expressivity subset of class expressions consisting of class intersection and the `toClass` and `hasValue` restrictions), and then tested for graph subsumption. A `NeededService` will match an `OfferedService` if the graph derived from its parameters (that is, resulting from the translation into triple patterns of the class expressions used to restrict its parameters) is subsumed by the corresponding graph in the `OfferedService`. We do not perform full DAML+OIL class subsumption reasoning on the service profiles; integration with a description logic reasoner such as FaCT has been left for future work. In particular, the development of a system component which can broker composite processes based on simple processes expressed using an ACL process ontology (effectively interaction patterns which involve several agents) was beyond the scope of this work.

The data sources in the system are grouped into three rough categories. The first category consists of entities which have GPS devices and so can produce high-certainty reports of their own positions and movements. The second category consists of entities which are able to observe their immediate environ-
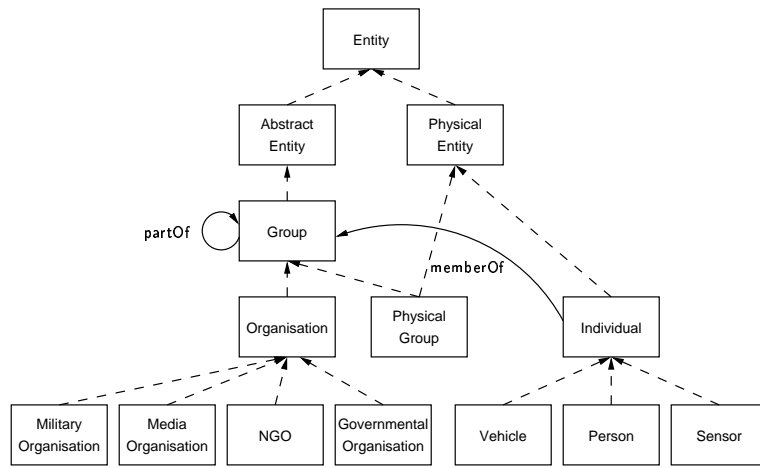
13

Fig. 8. Domain ontology - entity

ment, and so are able to generate moderate certainty reports on the movements of other entities, on hostile, support and relief events or on changes to the infrastructure present in the environment (damage to roads and bridges, for example). The final category consists of meteorological sensors which provide reports on the level of the flood waters.

5.2 Domain Ontology Design

As a demonstration of our approach to Agent Web Services, we have designed an ontology to describe the application domain and scenario that we have outlined earlier. This domain has a number of features which are interesting from an information management point of view. An agent in such a system is unlikely to have direct knowledge of the status of entities in the domain environment, since almost all knowledge is mediated through reports of events (entity state changes) which are issued by other entities in the system. For this reason, the provenance and certainty of the reports become of prime importance, and the role of the agents through which users interact with the system becomes one of information triage and filtering.

Therefore, the queries which agents ask of the system are less likely to be about domain entities directly, and are more likely to be about reports about those entities. In a scenario where there may be many conflicting and partial reports, the query idiom would be to ask only for high certainty reports from trusted sources about those entities which are of interest, but one has the ability to configure or change this assumption. To this end, the ontology that we have designed comprises two main parts. The first part consists of the entities in the domain environment and their invariant properties, as shown in Figure 8
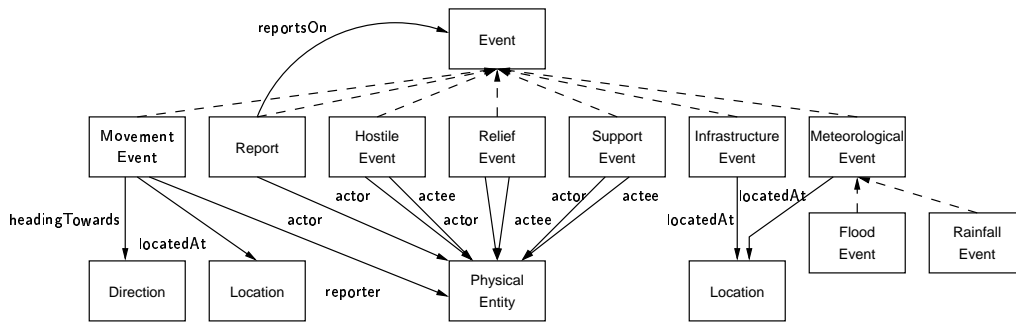
Fig. 9. Domain ontology - event

The other part of the ontology consists of the events which describe changes to the state of the entities, as shown in Figure 9. The key class in this hierarchy is the Report, which represents information which has been gleaned from some source (newsfeed, satellite image, etc.) about some event which has occurred in the environment. For example, if the movement of an entity (a relief convoy) has been observed by a third party (a journalist from CNN), this datapoint is represented by a report about a movement event by the convoy, which has been reported by the journalist. This approach captures the provenance of the report (as the entity which reported it), as well as the degree of certainty that the reporter has in the report. We have adopted the Stanford Certainty Factor Algebra [27] for dealing with certainty measures; although this has some shortcomings, it is a well-understood formalism and provides a general representation of confidence.

The separation of reports from events makes it possible to separate the time at which the event occurred from the time at which the report was made, which allows us to represent both the timeliness of reported events (yet another report facet which can be used to filter the stream of incoming reports) and also to represent event predictions as reports about future events. Finally, by making Report a type of Event, we make possible secondary reporting (reports about reports).

We have designed this ontology using the Protege ontology editor [28] which uses RDF Schema as its output format, but in the examples that follow in this report, we use DAML+OIL constructs to define new (unnamed) classes based on the primitive classes in the RDF Schema ontology. An example of these unnamed classes can be found in the parameter restriction in Figure 3. These unnamed classes are especially useful because they allow us to specify classes which were not explicitly created by the ontology designer by describing the necessary and sufficient conditions for class membership. The ability to specify unnamed classes simplifies ontologies by removing the need to formally name classes that are only used in the definition of other classes (for example, in local property range constraints), and is a particular strength of description logic characterisations of ontologies.
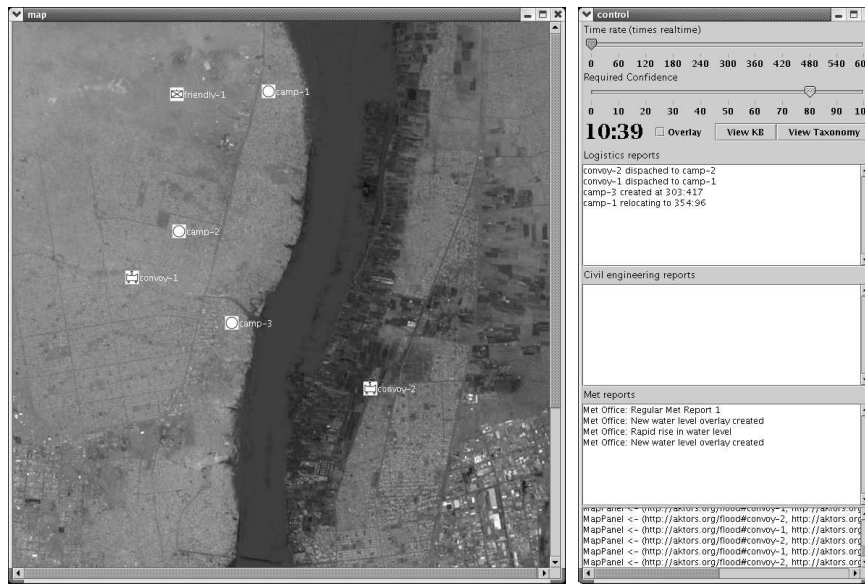
15

Fig. 10. Simulator Interface

5.3 Simulator Software

Our proof of concept implementation is a Java application which simulates a day of events for the situational awareness system. The events and reports themselves are predetermined and are served according to a script, but the behaviour of the information agents (requesting services and responding to service requests) is not fixed beforehand.

The application consists of two parts, a map panel which provides an overview of the entities in the simulated environment and the current whereabouts, and a control panel which provides a more detailed view of the flow of reports in the system and contains a number of report consumers (users).

The map panel shown on the right of the diagram in Figure 7 is a canonical example of a consumer; it registers its service requirement with the broker, and then subscribes to movement reports from the relevant sources (see the screenshot on the left in Figure 10). The next three consumers provide an aggregation service to the system (and so are not 'pure' consumers) by subscribing to certain types of report, cross-correlating those reports which deal with the same event, and then generating new composite reports which describe those events (often with greater certainty, due to the combination of knowledge from different sources). Finally, the flood level overlay agent takes flood reports and generates an overlay for the map panel which indicates the areas of the map which are under flood waters (again, see the screenshot in Figure 10).

The control panel for the simulator, shown on the right in Figure 10, provides

16

an overview of the system. For this simulation, we assume a context in which users have an interest in supply, an interest in keeping the bridges up and a requirement to understand the state of the weather. Consequently, the three panes in the middle of the panel show the filtered streams of reports which are being delivered to the various users in the system (respectively, logistics, civil engineering and meteorological). The scrolling pane at the bottom shows an aggregated feed of all the reports that have been made in the system (effectively a user who subscribes to all reports).

As a proof of concept and illustrative example, our demonstration implementation of this system differs from the description above in several important ways. We have chosen not to implement the interactions between system components using Web Services technologies such as SOAP. In our implementation, the agents communicate using standard Java method invocation. This choice is largely unimportant because the process of transferring the system to use SOAP as a message transport technology instead of Java is straightforward, given the procedure-call approach that we have taken in the FIPA ACL process ontology.

The hybrid WSDL-based service grounding in the most recent version of DAML-S is sufficiently expressive for us to be able to use SOAP instead of Java method invocation for communication; our future plans for this system include a migration to a SOAP-based message transport layer.

Even though we do not currently use Web Services technologies for message transport, our implementation is still ontologically informed; the reports and events are described using our domain ontology, and expressions from the domain ontology are used as service requests, demonstrating that a formal ontology of a domain can be effectively used to filter and aggregate knowledge and services.

6   Conclusion

In this paper, we have described our experience of building a flexible agent-based Web Services system which performs information triage on heterogeneous streams of data in order to provide a situational awareness capability in a simulated humanitarian aid scenario.

A key aspect of the system design is the separation of the intentional force of the messages from their application domain-specific content, and the embodiment of this separation in the process and profile components (respectively) of a DAML Services service description. The resulting rich service descriptions provide a powerful way of assembling information resources in contexts that

17

require the agile construction of virtual organisations. This agent-based perspective on Web Services is very consistent with the views on the construction of distributed information systems to be found in the Semantic Web and also the Semantic Grid [9].

This work has highlighted the need for expressive query languages which fit well with existing Web Services and Semantic Web technologies. In addition to the migration of the message transport layer to SOAP as mentioned in the previous section, our plans for future work on this system include the investigation of such query languages and their interactions with service brokerage, particularly where compisite processes are involved.

In addition to the issue of query languages for the Semantic Web (and Semantic Web Services), this work also demonstrates a mismatch between the approaches taken to the combination of services in the Web Services and multiagent systems communities. The models of service composition assumed by ontologies such as DAML Services appear to be insufficient to express the agent protocols found in FIPA and other agent systems; the investigation of techniques for representing such stereotypical interactions within the Web Services environment is a strong candidate for future research.

7 Acknowledgments

References

[1] L. Gasser, MAS Infrastructure: Definitions, Needs and Prospects, in: Infrastructure for Agents, Multi-Agent Systems and Scalable Multi-Agent Systems, Vol. 1887 of Lecture Notes in Artificial Intelligence, Springer-Verlag, 2001, pp. 1–11.

[2] K. Decker, K. Sycara, M. Williamson, Matchmaking and Brokering, in: Proceedings of the Second International Conference on Multi-Agent Systems

(ICMAS-96), 1996, p. 432.

[3] N. Guarino, P. Giaretta, Ontologies and Knowledge Bases: Towards a Terminological Clarification, in: N. Mars (Ed.), Towards Very Large Knowledge Bases, IOS Press, 1995, pp. 25–32.

[4] A. Gómez-Pérez, O. Corcho, Ontology Languages for the Semantic Web, IEEE Intelligent Systems .

[5] J. Searle, Speech Acts: An Essay in the Philosophy of Language, Cambridge University Press, 1969.

[6] FIPA, FIPA Communicative Act Library Specification, Tech. Rep. XC00037I, Foundation for Intelligent Physical Agents (Oct. 2002).
URL http://www.fipa.org/specs/fipa00037/

[7] KQML Advisory Group, An Overview of KQML: A Knowledge Query and Manipulation Language (Mar. 1992).

[8] I. Foster, C. Kesselman, J. M. Nick, S. Tuecke, The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration (Jan. 2002).
URL http://www.globus.org/research/papers/ogsa.pdf

[9] D. de Roure, N. Jennings, N. Shadbolt, The Semantic Grid: A future e-Science infrastructure, in: F. Berman, A. J. Hey, G. Fox (Eds.), Grid Computing: Making the Global Infrastructure a Reality, John Wiley and Sons, 2003, pp. 437–470.

[10] M. Lyell, L. Rosen, M. Casigni-Simkins, D.Norris, On software agents and web services: Usage and design concepts and issues, in: Proceedings of the AAMAS2003 Workshop on Web Services and Agent-based Engineering (WSABE2003), 2003.

[11] Z. Maamar, Q. Z. Sheng, B. Benatallah, Interleaving web services composition and execution using software agents and delegation, in: Proceedings of the AAMAS2003 Workshop on Web Services and Agent-based Engineering (WSABE2003), 2003.

[12] D. Brickley, R. Guha, Resource Description Framework (RDF) Schema Specification 1.0, Tech. Rep. CR-rdf-schema-20000327, World Wide Web Consortium (Mar. 2000).

[13] D. Connolly, F. van Harmelen, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, L. A. Stein, DAML+OIL (March 2001) Reference Description, W3C Note, World Wide Web Consortium (Dec. 2001).
URL http://www.w3.org/TR/daml+oil-reference

[14] E. Christensen, F. Curbera, G. Meredith, S. Weerawarana, Web Services Description Language (WSDL) 1.1, W3C note, World Wide Web Consortium (Mar. 2001).
URL http://www.w3.org/TR/wsdl

[15] The DAML Services Coalition, DAML-S: Semantic Markup for Web Services (Dec. 2002).
URL http://www.daml.org/services/

[16] P. V. Biron, A. Malhotra, XML Schema Part 2: Datatypes, W3C Recommendation, World Wide Web Consortium (May 2001).
URL http://www.w3.org/TR/xmlschema-2/

[17] V. Benjamins, E. Plaza, E. Motta, D. Fensel, R. Studer, B. Wielinga, G. Schreiber, Z. Zdrahal, IBROW3 - An Intelligent Brokering Service for Knowledge-Component Reuse on the World Wide Web, in: Proceedings of KAW'98, 1998.

[18] D. Fensel, V. Benjamins, E. Motta, B. Wielinga, UPML: A Framework for Knowledge System Reuse, in: Proceedings of the International Joint Conference on AI (IJCAI-99), 1999, pp. 16–23.

[19] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. F. Nielsen, S. Thatte, D. Winer, Simple Object Access Protocol (SOAP) 1.1, W3C note, World Wide Web Consortium (May 2000).
URL http://www.w3.org/TR/SOAP/

[20] A. Banerji, C. Bartolini, D. Beringer, V. Chopella, K. Govindarajan, A. Karp, H. Kuno, M. Lemon, G. Pogossiants, S. Sharma, S. Williams, Web Services Conversation Language (WSCL) 1.0, W3C Note, World Wide Web Consortium (Mar. 2002).
URL http://www.w3.org/TR/wscl10/

[21] L. Ardissono, A. Goy, G. Petrone, Enabling conversations with web services, in: Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS2003), 2003.

[22] S. Decker, DAML Rules - An RDF Query, Inference and Transformation Language, draft available online at http://www-db.stanford.edu/~stefan/daml/2001/07/03/rules/damlrules.ps (2001).

[23] R. Fikes, P. Hayes, I. Horrocks, DAML Query Language (DQL), Tech. rep., DAML Joint Committee (Apr. 2003).
URL http://www.daml.org/dql/

[24] H.-P. Labs, RDQL - RDF Data Query Language (2003).
URL http://www.hpl.hp.com/semweb/rdql.htm

[25] Aidministrator, User Guide for Sesame (Sep. 2003).
URL http://sesame.aidministrator.nl/publications/users/

[26] O. Lassila, R. Swick, Resource Description Framework (RDF) model and syntax specification, Tech. Rep. REC-rdf-syntax, World Wide Web Consortium (Feb. 1999).

[27] B. Buchanan, E. Shortliff (Eds.), Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project, Addison-Wesley, 1984.

[28] N. Noy, M. Sintek, S. Decker, M. Crubezy, R. Fergerson, M. Musen, Creating Semantic Web Contents with Protege-2000, IEEE Intelligent Systems 16 (2) (2002) 60–71.