Ontology definition languages for Multi-Agent Systems: the Geographical Information Ontology case study

Luís Mota
ISCTE
Av. das Forças Armadas
1649 – 026 Lisboa, Portugal
+351 21 790 39 13
luis.mota@iscte.pt

João Bento
DECivil - IST
Av. Rovisco Pais
1049 - 001 Lisboa, Portugal
joao@civil.ist.utl.pt

Luís Botelho
ISCTE
Av. das Forças Armadas
1649 – 026 Lisboa, Portugal
luis.botelho@iscte.pt

ABSTRACT

In the scope of the AgentCities project, we worked on the definition of a Geographical Information Ontology, intended for a wide use, in particular in agent systems.

Since the choice of the best modeling approach was not clear, we decided to express the ontology in four different languages: UML, RDF, DAML+OIL and Ontolingua. This paper describes the results of this modeling and highlights the most interesting characteristics of each approach from our point of view.

We recognized that each approach has its own advantages and specific use scenarios, but we conclude that, due to its history, expressive power and tools, Ontolingua is the most powerful solution for our modeling needs. Furthermore, we present proposals to solve some open questions and shortcomings: the introduction of actions in the analyzed frameworks, the creation of a XML[3] syntax for Ontolingua and the definition of decidable subsets for Ontolingua.

1. INTRODUCTION

While participating in the AgentCities.RTD [1] project, we are developing an ontology for Geographical Information (GI).

The goal of the AgentCities project is to create an on-line, distributed testbed to explore and validate the potential of agent technology for future dynamic service environments. One of its goals is the validation and refinement of agent communication technologies, such as ontology models.

In order to analyze the advantages and disadvantages of the available solutions, we decided to express the GI Ontology in different languages. The development of the ontology was accomplished through the following tasks:

Informal gathering of the requirements and use-cases (Section 2).

Research on the available languages and tools for modeling ontologies. After a short overview of the existing models, we chose four different approaches: UML [5], DAML+OIL [7], RDF [4] and Ontolingua [8].

Formal expression of the GI ontology in the four languages. The code and results can be found in [2].

Analysis and evaluation of the results of each modeling approach (Section 3).

Section 4 proposes the extension of current ontology frameworks with the capability to represent actions, using an approach based on artificial intelligence planning and communicative acts semantics.

Finally, section 5 presents conclusions and suggests directions for future developments, in particular the need to define a XML syntax for Ontolingua and the definition of decidable subsets of Ontolingua.

A full version of this paper can be found in [2].

2. REQUIREMENTS OF THE ONTOLOGY

In the AgentCities project, some of the planned agents will deal with Geographical Information. We will generally refer to these agents as the GI Agent. There is an agent that deals with geographic information (GIAgent) and two types of agents that interact with the GIAgent: general agents (General Agent) and service suppliers.

The geographic information maintained by each GI Agent pertains to a single city.

We have modeled several entities in the domain of geographical information, the most relevant of which, are location, path, distance and spatial reference system. Locations can be created and modified, upon request from any service supplier, and can be queried by any agent.

Postal address (in the ontology: AddressLocation), geodesic coordinates (longitude-latitude-height triples - GeodesicLocation) and cartographic coordinates (map points – CartographicLocation) are foreseen as possible forms to express location.

Any agent can query path entities. A path is a way to go from a point to another. It is possible to constrain a path to include a specified set of stopovers. It is also possible to choose the means of transportation. Similarly to Locations, there are three kinds of Paths: GeodesicPath, CartographicPath and RoadPath, which is made of road and street segments.

In our type of application, there is no need to use entities of the type polygon. Therefore, these will not be included in the ontology.

There is the need to convert between different location coordinates. As an example, this functionality allows the conversion of a location in geodesic coordinates into cartographic coordinates.

In order to accurately specify a location, geodesic locations always refer to an ellipsoid and datum (DatumDescription), and

cartographic locations refer to a cartographic projection (ProjectionDescription). A datum and a projection are generally called Space Reference Systems. There is therefore the need to query the details of a Space Reference System.

To be able to determine the geographic distance between locations, we need a function called Distance, which takes two Locations as arguments and outputs a value in a pre-defined unit.

As it has been pointed out, GI Agents need to perform some operations: *register*, i.e., accept one entity and register its existence in the agent's Data Base; *modify*, i.e., change the details of an object already in the DB; and *delete*, i.e., remove an object from the DB. The URL to access a map in the Web can also be asked. If such map exists (or can be created), there must be a way to relate it to the respective URI.

3. ANALYSIS OF THE RESULTING ONTOLOGIES

3.1 UML

Due to its graphical notation, UML is a good tool for the creation of a first sketch of the ontology and can be used as a sharing tool.

The possibility to use methods to represent actions allows defining at least part of the service ontology and also actions performed by domain entities. The existence of n-ary associations also enables the intuitive creation of concepts that relate n entities.

UML lacks some formal modeling primitives such as lists and sets. The list concept is well suited for the definition of Paths between points, since these are ordered lists of geographical positions. However, these lists could only be rigorously modeled through the use of OCL [5], a constraint language which is part of UML, but whose integration in UML's meta-model is questioned [6]. To keep the graphical simplicity of the diagrams, an essential characteristic for its easy understanding, we decided to model Paths (CartographicPath, GeodesicPath and RoadPath.) as classes with an association with the n path nodes, each with an ordering number.

Part of the modeling in UML, e.g. the creation of comments and restrictions, is usually made in natural language, possibly resulting in some formal accuracy. UML's graphical nature facilitates its use by people but impairs its direct treatment by agents. To cope with this limitation, OMG created a specification for the sharing of diagrams, based on XML [10]. This solution is based on DTD definitions, which are being abandoned in favor of the widest development of XML Schema.

3.2 DAML

Given the relative youth of DAML+OIL, the creation of ontologies in this format still faces some problems: rarity of good quality dedicated editors and poor quantity and quality of other tools. The editing of the ontology had thus to be made with a text editor, which does not provide any kind of assistance. Since a DAML+OIL document is not easily readable, the editing ends up being difficult if it lacks automated assistance.

DAML+OIL aims explicitly at the definition of ontologies, thus it supplies some ways to write meta-information about the ontology, such as the version, the author and an informal description, and the imported ontologies.

The use of XML data types was very convenient, allowing the rigorous representation of natural concepts such as longitude, which was defined as a real number between -180 and 180.

The distinction between DatatypeProperty and ObjectProperty, in conjunction with the XML data types, adds semantic power to DAML+OIL since it makes it possible to distinguish relations between entities from simple object features.

The use of lists and sets was difficult and inadequate. DAML+OIL provides the primitive concept 'Container' with subclasses 'Bag' (unordered and allowing duplicates), 'Seq' (ordered list) and 'Alt' (alternative elements). Although Seq could be used to define the subclasses of Path, this was not done because it is impossible to restrict the elements of the list to a single class, which is essential to this modeling. For this reason, we created the class PathNode, representing the elements of a path. PathNode (CartographicPathNode, GeodesicPathNode and RoadPathSegment) has the attribute 'nodeOrder' that represents the position of the node in the list. Since, in DAML+OIL the expression of arbitrary axioms is not allowed, the 'nodeOrder' property can only be restricted to positive integers, which is not enough since the node ordering must be a sequential number, not just any positive integer.

DAML+OIL does not support the representation of actions or functions, thus the concept of registering a new location could not be modeled, and the distance between two locations had to be modeled as a class. The unavailability of actions hinders the direct use of this language in multi-agent systems in which the communication language requires these, as in FIPA ACL[11]. There are already proposals to overcome this shortcoming, but they still lack general acceptance.

Since properties are first-class entities, their scopes are not restricted to a particular class. Therefore, properties that could have similar names if defined within their classes, must actually have different names, e.g. name of ProjectionDescription, and name of Datum', had to be modified to projectionName and datumName to prevent ambiguities.

3.3 RDF

The relation between RDF and DAML+OIL is very strong but RDF's expressiveness, without any extensions to its current version, is more limited than that of DAML+OIL. Therefore, all the limitations previously identified for DAML also apply in this case, but RDF additionally lacks support for other features.

The expression of ontologies is not the purpose of RDF, thus, contrarily to DAML+OIL, there is no ability of expressing meta-information about an ontology. RDF does not allow the specification of the cardinality of attributes, which hinders the characterization of a property as being single-valued, among other things. It is also impossible to specify that one property identifies an object.

In the present version of RDF Schema there is no way to deal with data types, thus the range of a property is limited to classes and strings. Consequently, some attributes such as *latitude*, could not be treated conveniently. The inexistence of data types also implies that there is no distinction between value type properties and object properties.

3.4 Ontolingua

The ontology in Ontolingua was entirely defined through the web interface available in [8]. This interface is user friendly and performs satisfactorily. Besides the ontology editor, other services are available: Chimaera, an ontology analyzer and an ontology graphical presentation tool. The editor also allows to export any available ontology to several other formats, for external use through knowledge representation or reasoning systems, such as, CLIPS, CML, EpiKit, Loom or Prolog.

Ontolingua also allows ontologies to specify the inclusion of other ontologies, which enables the reuse of previous work. We chose to include, among others, the 'Kif-Lists' ontology, which enabled us to correctly model the subclasses of Path as lists. With a vast community of users, a considerable number of ontologies are already defined and available for use in ontology servers.

Ontolingua accepts the definition of generic axioms in KIF. We used this feature to write some simple axioms that may be used to create arbitrary relations between the entities of the ontology. For example, in order to constrain the elements of a CartographicPath (a list) to be nodes defined by cartographic coordinates (CartographicPathNode), we wrote the following axiom:

```
(=> (And (Cartographicpath ?Path) (Item ?Node ?Path))
        (Cartographicpathnode ?Node))))
```

This axiom states that any member of a CartographicPath is a CartographicPathNode. The previously analyzed approaches did not allow writing arbitrary axioms.

An extensive collection of data types is available. These can be constrained through the use of facets, such as '*numeric-maximum*'. With these facets we were able to conveniently define the range of some slots, e.g. longitude, without the use of an external language, as it was the case with XML and DAML+OIL. To code this data type, we defined the following restriction on a real number:

```
:Template-Facets
((Longitude (Numeric-Minimum -180) (Numeric-Maximum 180)).
```

It is possible to declare functions. In this ontology the distance between two locations was modeled as a function, and not as a class, as in DAML+OIL. The definition of this function is:

```
(Define-Function Distance (?Location-0 ?Location-1):->?Value
  "Distance, in meters, between two locations"
  :Def (And (Location ?Location-0) (Location ?Location-1)
  (Number ?Value)))
```

The declaration and definition of actions is not supported. This implied that part of the service ontology could not be represented.

3.5 Summary

All the ontology representation frameworks analyzed in this paper lack a natural and complete way of declaring actions. Functions and general predicates can be declared and defined only in Ontolingua. Ontolingua and, in a less straightforward way, UML can appropriately represent the concept of a sequential collection of points such as the path between two locations. Ontolingua is also the only framework enabling the expression of arbitrary axioms that capture general relations between entities of the ontology.

Although Ontolingua is substantially more expressive than the other analyzed approaches, it has two drawbacks. First, it is not possible to define a general inference procedure for the whole power of the language. Second, its lisp-like notation is not usual in the Web.

In the remaining of the paper we consider some of the mentioned problems, namely the representation of actions and the definition of decidable subsets of Ontolingua.

4. EXTENTION OF THE FRAMEWORKS TO SUPPORT ACTIONS

The basic modeling of actions could be obtained through the introduction of new primitives in the different languages. Such primitives should minimally allow stating the name of the action and its arguments, and additionally its pre-conditions and its consequences. However, this support cannot be easily introduced. In UML, which is an object oriented approach directed towards software developers, to go beyond the modeling of actions through methods, profiles and stereotypes could be used. In DAML and RDF, concepts as variable, function or action parameter do not exist. Ontolingua, the most mature approach, is not as easy to extend as DAML and RDF. Therefore, this support can only be obtained through a revision of the KIF specification. At the present time, the creation of a 'Common Logic' [12], from which KIF is one of the foreseen concretizations, is under debate. This is an appropriate forum to discuss the modeling of actions.

5. LIMITATION OF ONTOLINGUA EXPRESSIVENESS

Ontolingua's potential undecidability hinders its use in reasoning systems without the introduction of additional assumptions, such as the closed world assumption, and/or the reduction of the underlying expressiveness. To deal with this, we propose the creation of subsets for Ontolingua. Since KIF foresees the existence of 'conformance profiles' to limit its' expressiveness [14], the creation of the desired subsets of Ontolingua is expectably simple.

6. ACKNOWLEDGMENTS

The research described in this paper is partly supported by UNIDE/ISCTE and partly by the EC project Agentcities.RTD, reference IST-2000-28385. The opinions expressed in this paper are those of the authors and are not necessarily those of the Agentcities.RTD partners.

7. REFERENCES

[1] Willmott, S.; Dale, J.; Burg, B.; Charlton, P; and O'Brien, P. 2001. "Agentcities: a worldwide open agent network". *Agentlink News, 8:13-15*

[2] ADETTI, Associação para o Desenvolvimento das Telecomunicações e Técnicas da Informática, http://www.adetti-linha4.org

[3] eXtensible Markup Language, http://www.w3.org/XML

[4] O. Lassila, R.R. Swick (Editors), 'RDF Model and Syntax Specification', W3C, 1999, http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/

[5] 'Unified Modeling Language, v1.4', OMG, http://www.omg.org/technology/documents/formal/uml.htm

[6] 'UML 2.0 OCL RFP', p. 24, Object Management Group, http://cgi.omg.org/cgi-bin/doc?ad/00-09-03

[7] DAML+OIL, Darpa Agent Markup Language, http://www.daml.org/2001/03/

[8] Ontolingua server, http://ontolingua.stanford.edu/

[9] 'FIPA Ontology Service Specification', Foundation for Intelligent Physical Agents, 2000, http://www.fipa.org/specs/fipa00086/

[10] 'OMG XML Metadata Interchange (XMI) Specification', Object Management Group, http://www.omg.org/cgi-bin/doc?formal/2002-01-01

[11] 'FIPA ACL Message Structure Specification', http://www.fipa.org/specs/fipa00061/

[12] 'Common Logic', cl.tamu.edu/minutes/stanford-minutes.html

[13] 'FIPA SL Content Language Specification', http://www.fipa.org/specs/fipa00008/

[14] 'Knowledge Interchange Format - draft proposed American National Standard (dpANS)', http://logic.stanford.edu/kif/dpans.html

[15] 'Epilog Inference Package', http://logic.stanford.edu/sharing/programs/epilog/

Table 1: Summary of the languages' features

	RDF/RDFS	UML	Ontolingua	DAML+OIL
Classes	Supports the concept of class. Basic modeling capacity: inheritance in classes and properties.	Supports the concept of class. Basic modeling capacity: inheritance in classes and restrictions through OCL.	Supports the concept of class. Rich modeling capacity: inheritance in classes, definition of functions, general axioms, and pre-defined facets.	Supports the concept of class. Medium modeling capacity: inheritance in classes and properties, disjunction and identity between classes, identity and cardinality of properties.
Entities	As instances of classes.	As instances of classes.	As instances of classes.	Capacities from RDF.
Attributes	Support for only one data type (*literal/string*). No distinction between object features and relations between objects. Both cases are dealt with properties, of binary type and with autonomous existence	Supported through '*attributes*'. Their scope and existence is internal to the class.	Supported through '*template-slots*', whose scope and existence are external to the class.	Supported through '*DatatypeProperties*', with XML-Schema data types.
Relations		Supported through n-ary '*associations*'.	Supported through relations and functions	Supported through '*ObjectProperties*'. In both cases, properties are binary.
Functions	Not natively supported.	Can be declared, internally to classes, through '*methods*'.	Allows the definition of functions, through KIF expressions.	Not natively supported.
Actions	Not natively supported.	Only as methods, internally to classes.	Not supported.	Not natively supported.
Sharing and access by agents	Written in text, sharable through the Web.	Expressible in text (XMI), sharable through the Web.	Written in text, sharable through the Web, accessible through OKBC.	Written in text, sharable through the Web.
Parsing tools	Several dedicated parsers available, for different programming languages.	Several XML parsers available, for different languages.	Several KIF parsers available, for different programming languages.	Tools available for RDF only. Limited specific support for DAML.
Reasoning tools	Only a few, in development.	No.	Epilog[15], a reasoning package with KIF's full expressiveness.	Tools for RDF and OIL only. Limited specific support for DAML.
Editors	One known application.	Several commercial and free source editors.	Through a Web interface, with free access.	One dedicated application, some adaptable tools.
Integration in platforms	JADE accepts RDF as a content language.	No.	Integration mentioned in [9].	No.