



Citation for published version:

Duke, M & Patel, M 2003, *An Ontology Server for AgentCities.NET*. UKOLN, University of Bath.

Publication date:
2003

[Link to publication](#)

Publisher Rights
CC BY

University of Bath

Alternative formats

If you require this document in an alternative format, please contact:
openaccess@bath.ac.uk

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

AGENTCITIES TECHNICAL NOTE

An Ontology Server for Agentcities.NET

Agentcities Task Force Technical Note

actf-note-00008, 15 November, 2010

Authors:

Monica Duke, UKOLN, University of Bath

Manjula Patel, UKOLN, University of Bath

Copyright © 2003 is retained by the Authors and/or their respective organizations. The content is the sole responsibility of the authors and ACTF takes no responsibility for its correctness or fitness for purpose. The authors are also responsible for ensuring that this publication does not violate copyright agreements applying to the content in whole or in part.

Comments and requests should be addressed to tech-editor@agentcities.org.

This document and the information contained herein is provided on an "AS IS" basis and THE AGENTCITIES TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Status

Final

This version: <http://www.agentcities.org/note/00008/actf-note-00008a.html>

Latest version: <http://www.agentcities.org/note/00008/>

Abstract

Within this six month deployment project[1] we have concentrated on taking forward the ideas and systems developed in a number of initiatives in which UKOLN has been involved, chiefly among these the EU-funded DESIRE[6] and SCHEMAS projects[7], the UK MEG Registry project[15] and the Dublin Core Metadata Initiative[5]. All of these projects explored approaches to declaring and sharing metadata vocabularies using RDF Schemas[18]. We have adapted software for a metadata vocabulary registry to serve as an ontology server which can be queried by agents on the Agentcities.NET network. The contents of the server comprises metadata vocabularies which may be regarded as simple forms of ontology.

42 **Contents**

43 Agencities Technical Note 1

44 1 Introduction 3

45 2 Ontologies and Metadata Vocabularies 3

46 2.1 Ontology Description Languages 4

47 2.1.1 RDF Schema 4

48 2.1.2 DAML+OIL 4

49 2.1.3 DAML+OIL 4

50 2.1.4 RDFS(FA) 4

51 3 Ontology Servers and Metadata Registries 5

52 3.1 The SCHEMAS Metadata Registry 5

53 3.2 BT's Ontology Server 5

54 3.3 The Dublin Core Metadata Initiative's Registry 5

55 3.4 Other Initiatives 5

56 3.5 The MEG Registry 6

57 3.5.1 The MEG Registry model of metadata vocabularies 6

58 4 The UKOLN Ontology Server 7

59 4.1 Web Interface 8

60 4.2 The UKOLN Agent Platform 9

61 4.3 Overview of functionality 9

62 4.3.1 The Server Agent 10

63 4.3.2 Server Ontology 11

64 4.4 Interrogating the Server Agent 14

65 4.4.1 The GUI Agent 14

66 4.4.2 The Command Line agent 16

67 4.4.3 Behaviours 16

68 5 Conclusions 17

69 6 Acknowledgements 17

70 7 References 18

71 Appendix: The MEG Registry Data Model 21

72

73 1 Introduction

74 This is a report on the work carried out between 1st September 2002 and 28th February 2003
75 at UKOLN, as part of the European Commission funded 5th Framework IST project
76 Agentcities.NET [4]. UKOLN was awarded a grant under the Deployment support program, a
77 "series of grants to support independent new innovative exploratory work related to the
78 Agentcities.NET network. The intention is to enable members to connect their existing or new
79 agent systems to the Agentcities network and carry out exploratory mini-projects - leading to
80 innovative ideas, technology development and new larger scale collaborative projects."

81
82 UKOLN [3] is a centre of expertise in digital information management, providing advice and
83 services to the library, information, education and cultural heritage communities. UKOLN is
84 involved in many standardization activities, including the Dublin Core Metadata Initiative
85 (DCMI)[5]; the Research and Development team at UKOLN has taken part in several EU
86 projects including DESIRE[6] and SCHEMAS[7].

87
88 The aim of this project is to investigate the support of automated querying of metadata
89 vocabularies by agents, to acquire the semantics associated with specific metadata terms.
90 The approach taken is that of using a registry within which metadata vocabularies are
91 expressed and through which they are communicated. In a registry environment, individual
92 terms as well as whole vocabularies can be investigated by agents. The registry supports the
93 discovery, sharing and re-use of vocabularies, facilitating the convergence of vocabularies (or
94 ontologies), in particular for specific domains. The hope is that alignment in this way will
95 improve the prospects of interoperability of systems in specific sectors.
96

97 2 Ontologies and Metadata Vocabularies

98 Ontologies provide a common vocabulary of an area and define, with different levels of
99 formality, the meaning of the terms and the relations between them. They aim to capture
100 domain knowledge in a generic way and provide a commonly agreed understanding of a
101 domain, which may be reused and shared across applications and groups [10]. Ontologies
102 are used by people, databases, and applications that need to share domain information.
103 There are several other definitions and typologies of ontologies; for an overview [10, 11] are
104 good sources. Some definitions may follow from the way that ontologies are built and used;
105 distinctions are made between lightweight and heavyweight ontologies, where taxonomies are
106 considered to be one of the former, whereas the latter kind of ontologies would be expected
107 to include axioms. For example Sowa [12] defines a terminological ontology as "an ontology
108 whose categories need not be fully specified by axioms and definition". WordNet [27] is
109 an exmple of such an ontology. Other distinctions are based on the kind of languages used
110 to implement ontologies, such that some ontologies are rigourously formal if they are defined
111 in a language with formal semantics, theories and proofs (e.g. of soundness and
112 completeness). Others are only highly informal being expressed only in natural language.
113 Some ontologies are intended to be reusable across domains but several are specific to a
114 domain.

115
116 Knowledge in ontologies is mainly formalized using five kinds of components: classes,
117 relations, functions, axioms and instances. For a description of these components refer to
118 [10]. However, in this project we are concerned with only a specific type of simple ontology,
119 referred to in the SCHEMAS project as a vocabulary[13]:

120 *"In our usage, the term evokes a semantically rich dictionary environment, with pointers to*
121 *related terms – more than just a flat word list. (Another common synonym for "vocabulary" is*
122 *"element set". Similarly, though we prefer to speak of metadata "terms", the term "elements"*
123 *is a close synonym.)"*

124
125 Further, the SCHEMAS project developed the notion of an *Application Profile*[9] which is a type of
126 metadata vocabulary that draws on canonical vocabularies and customizes them for local use. The
127 precise use of the terms vocabulary and application profile and how they are modeled in our work will
128 be expanded on in section 3.1.

129 **2.1 Ontology Description Languages**

130 Semanticweb.org [25] provides an encapsulation of the history of the representation of
 131 ontologies on the Web. More recently the OWL Web Ontology Language[22] is being
 132 designed by the W3C Web Ontology Working Group[19] in order to provide a language that
 133 can be used for applications that need to understand the content of information instead of just
 134 understanding the human-readable presentation of content. OWL facilitates greater machine
 135 readability of web content than XML, RDF and RDF Schema[18] by providing an additional
 136 vocabulary for term descriptions. The OWL language is a revision of the [DAML+OIL web](#)
 137 [ontology language](#) incorporating learnings from the design and application use of
 138 DAML+OIL[36].

139 **2.1.1 RDF Schema**

140 The Resource Description Framework (RDF) is a general-purpose language for representing
 141 information on the Web. The RDF Schema specification [18] describes how to use RDF in
 142 order to describe RDF vocabularies.

143 **2.1.2 DAML+OIL**

144 DAML+OIL [21] is a semantic markup language for Web resources. It builds on earlier W3C
 145 standards such as RDF and RDF Schema, and extends these languages with richer
 146 modelling primitives. DAML+OIL provides modelling primitives commonly found in frame-
 147 based languages. A DAML+OIL knowledge base is a collection of RDF triples. DAML+OIL
 148 prescribes a specific meaning for triples that use the DAML+OIL vocabulary

149 **2.1.3 DAML+OIL**

150 The Web Ontology Language OWL [22] is a semantic markup language for publishing and
 151 sharing ontologies on the World Wide Web. OWL is developed as a vocabulary extension of
 152 RDFS and is derived from the DAML+OIL Web Ontology Language[21]. OWL is a language
 153 for defining and instantiating *Web ontologies*. Different subsets of the OWL language are
 154 defined, to suit different uses. OWL has been designed for maximal compatibility with RDF
 155 and RDF Schema, and an OWL ontology is represented as a set of RDF triples.

156 **2.1.4 RDFS(FA)**

157 RDFS(FA)[28] as a sub-language of RDFS introduces a Fixed layered metamodeling
 158 Architecture to RDFS, based on a relatively standard model-theoretic semantics. Therefore,
 159 first order languages, like DAML+OIL and [OWL](#), can be built on top of *both* the syntax and
 160 [semantics](#) of RDFS(FA). On the other hand, all RDFS(FA) statements are still *valid* RDFS
 161 statements, since RDFS(FA) imposes the restriction of stratification on the syntax of RDFS. It
 162 is intended to address the 'dual-roles' problem in RDF.

163 RDFS(FA) is designed to be a clean schema layer language (as a sub-set of RDFS), such
 164 that

- 165
- 166 • it is easy to understand and to use
- 167 • first order logics (e.g. [DAML+OIL](#) and [OWL/DL](#)) can be built on top of both its syntax
- 168 and semantics
- 169

170 RDFS(FA) is a Semantic Web schema language introducing a UML-like metamodeling
 171 architecture to RDFS. Built-in modelling primitives of RDFS are stratified into different strata
 172 (or layers) of RDFS(FA), so that certain modelling primitives belong to certain strata
 173 (layers). The semantics of modelling primitives depend on the stratum they belong to. All
 174 these strata form the metamodeling architecture of RDFS(FA). Theoretically there can be
 175 infinite number of layers in the metamodeling architecture, while in practice, four layers are
 176 usually described:

- 177 Stratum 0 (Instance Layer)
- 178 Stratum 1 (Ontology Layer)
- 179 Stratum 2 (Language Layer)

180 **3 Ontology Servers and Metadata Registries**

181 As used in the SCHEMAS Project, the term "registry" refers to a database that harvests
182 various types of metadata vocabularies from their maintainers over the Web. In response to
183 queries, such a registry should provide term-level documentation of definitions and usage
184 along with contextual annotations. It should in effect function as an indexing engine for
185 dynamically updating, merging, and serving up a large corpus of definitions for metadata
186 terms. The context for such a registry is the notion of a Semantic Web where anybody or any
187 organisation can declare a metadata vocabulary and assert a relationship between that
188 vocabulary and any other vocabulary on the Web.

189 **3.1 The SCHEMAS Metadata Registry**

190 The SCHEMAS project developed a metadata registry which was implemented using the
191 EOR toolkit (Extensible Open RDF toolkit)[37]. An RDF approach offered the potential of a
192 scaleable system based on a common data model (RDF) both for the schema and for the
193 database. The project was looking towards implementation of a repository which would be
194 populated with schemas harvested directly from their maintainers in an open Web
195 environment. However, at that time software tools for such a solution proved immature and
196 required a level of development effort beyond that available to the project. In addition the
197 chosen standard for schema specification (RDF Schema) was itself still under development,
198 and conventions for expressing metadata schemas, in particular *Application Profiles*[9], were
199 still to emerge.

200
201 The primary motivation for the work on the SCHEMAS Registry "has been to help humans
202 find out about metadata terms in use -- their official definitions, local variations and
203 extensions, and the various schemas in which they are embedded. The purpose is to help
204 designers of information services discover metadata terms that have already been created or
205 standardized by others and align their own schemas with those of related information
206 providers." [8]. However, the longer-term goal was "to build a corpus of machine-
207 understandable schemas that can be accessed and processed directly by various software
208 applications" [8].

209 **3.2 BT's Ontology Server**

210 The BT Ontology Server [31] is part of the [Agencities.RTD](#) initiative. The Agencities
211 Ontology Service is an agent and web application for managing and accessing DAML+OIL
212 ontologies and can be accessed by agents using open standards (the Agencities
213 interoperability stack). This allows ontologies to be created, managed and shared by agents
214 [32].

215 **3.3 The Dublin Core Metadata Initiative's Registry**

216 The Dublin Core Metadata Initiative is an open forum engaged in the development of
217 interoperable online metadata standards that support a broad range of purposes and
218 business models. The overall goal of the DCMI Registry Working Group[35] is the
219 development of a metadata registry providing authoritative information regarding the DCMI
220 vocabulary and the relationship between terms in that vocabulary. The group aims to provide
221 an operational registry with both user and machine interfaces over a phased development
222 period, with the aim of supporting acceptance and use of the DCMI vocabulary and providing
223 an authoritative source of information [35]. Work in this initiative is ongoing.

224 **3.4 Other Initiatives**

225 Other initiatives within the areas of ontologies, ontology representation, storage and exchange
226 have undertaken reviews of repositories of ontologies:

227

- 228 • The OntoWeb Technical RoadMap [10] reported on repositories of ontologies, listing
229 some of the 'best-known repositories'. The ontology repositories that are described
230 include those in which ontologies are implemented in DAML, Ontolingua and SHOE.

231

- 232 • More recently, the SWAD Europe Project reviewed RDF storage systems [20] including
 233 ones that may include schema and ontological data such as RDF Schema and
 234 DAML+OIL.

235
 236 The DAML Repository [30] is a web-accessible catalogue of ontologies expressed in DAML.

237 3.5 The MEG Registry

238 The *Metadata for Education Group* (MEG)[14] was formed following a meeting of key UK
 239 stakeholders and serves as an open forum for debating the description and provision of
 240 educational resources at all educational levels across the United Kingdom. This group seeks
 241 to reach consensus on appropriate means by which to describe discrete learning objects in a
 242 manner suitable for implementation in a range of educational arenas.

243
 244 Preceding work undertaken in the DESIRE[6] and SCHEMAS[7] projects provided the basis
 245 for the MEG Registry Project[15], which adopted a slightly modified data model as described
 246 in the Appendix. The aim of the MEG registry is to provide implementers of educational
 247 systems with a means to share information about their metadata schemas and to re-use
 248 existing schemas. The benefit being a saving of time and effort currently spent in researching
 249 existing schemas and in re-inventing schemas.

250
 251 In the next few sections we describe in some depth the models and definitions employed in
 252 the MEG Registry project as they have provided the framework for our work.

253 3.5.1 The MEG Registry model of metadata vocabularies

254 The registry is based on the following model of metadata vocabularies or element sets:

255
 256 **Element Sets** are owned and maintained by **Agencies**. **Element Sets** are made up of
 257 **Elements**. An

258 **Element Usage** may:

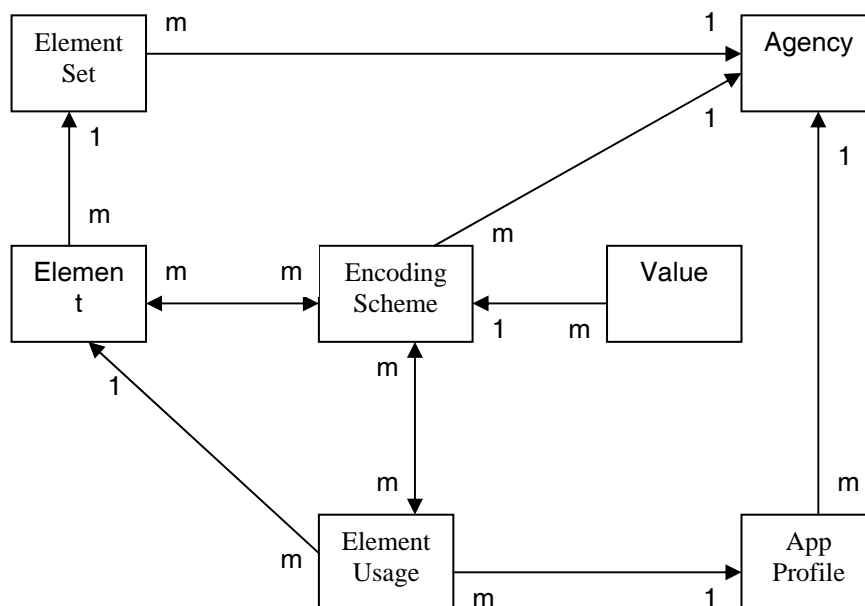
- 259 • introduce constraints on the value of an **Element** by associating it with one or more
 260 **Encoding Schemes**;
 261 • introduce constraints on the *obligation* to use an **Element** (e.g. make its use
 262 mandatory) or the *occurrence* of an **Element** (e.g. whether it is repeatable);
 263 • *refine* the semantic definition of an **Element** to make it narrower or more specific to
 264 the application domain.

265 **Encoding Schemes** constrain the value space of **Elements**. An **Application Profile** defines
 266 a set of **Element Usages** of **Elements** drawn from one or more **Element Sets**.

267
 268 The registry holds information on each of the entities and their relationships:

- 269 • **Element Sets** (i.e. on the Element Sets as units, rather than on their constituent
 270 Elements), including information on their intended scope/area of use and their
 271 relationship to other Element Sets;
 272 • the **Elements** which make up those Element Sets, including information on the
 273 semantics of the Elements and their recommended usage, and any semantic
 274 relationships to other Elements in this or other vocabularies (e.g. the relationship
 275 described by the DCMI concept of "element refinement" or by RDF Schema as a
 276 "sub-property" relation)
 277 • **Application Profiles**, including information on their intended scope/area of use and
 278 their relationship to other Element Sets and Application Profiles;
 279 • the **Usages of Elements** which make up those Application Profiles, including the
 280 Element used, any prescription of Encoding Schemes, and other constraints on
 281 element use;
 282 • **Encoding Schemes**, which constrain the value space of Elements, including
 283 information on their intended scope/area of use; where an Encoding Scheme takes
 284 the form of an enumerated list, the **values** prescribed by that Encoding Scheme may
 285 be recorded;
 286 • the **Agencies** who own/create/maintain Element Sets, Application Profiles, and
 287 Encoding Schemes

288 Diagrammatically, the relationship between the entities that are represented in the registry is
 289 modelled as follows (a more formal description is available in the Appendix).
 290
 291



292 The Meg Registry is implemented as a server based on the RDF toolkit, Redland [16]. The
 293 information about the above entities and their relationship is stored and made available in
 294 machine-processable format as RDF schemas. The existing registry API is developed in Perl
 295 and supports functions such as querying of the registry through an HTTP interface. The
 296 project also provided a tool that could support the creation and submission of metadata
 297 schemas in a distributed way, in particular promoting the re-use of elements and encoding
 298 schemes as described in [17].
 299
 300

301 The registry can be queried either through the schema creation tool so as to identify elements
 302 and encoding schemes for re-use, or directly through the HTTP APIs. One of the interfaces
 303 was intended for browsing and searching through a web browser, and returns HTML encoded
 304 representations of the structures and relationships of the element sets and related entities,
 305 which support easy navigation through the registry. Thus each of the entities (agency,
 306 element set, element, application profile, element usage and encoding scheme) can be either
 307 searched or browsed and the relationships can be explored.
 308

309 A second interface supports queries to search against element sets and encoding schemes,
 310 and returns RDF-encoded data.

311 4 The UKOLN Ontology Server

312 Recently, we have extended the work done in the MEG Registry project to re-deploy the
 313 interfaces to the registry within an agent environment, namely the Agentcities.NET[1]. The
 314 existing registry software stores information pertaining to metadata vocabularies and provides
 315 an interface for interacting with the information. We have thus transitioned from a human-
 316 centric to an agent-centric environment.
 317

318 We have deployed the MEG Registry software within an agent-enabled environment,
 319 mediating communication to the registry of schemas through an agent. The schemas (or
 320 element sets) are modelled within the Server as outlined in previous sections and in the
 321 Appendix. Exploration of the element sets is organised around the categories described by
 322 the model, (i.e. agency, element, element set, application profile, encoding scheme and
 323 element usage).
 324

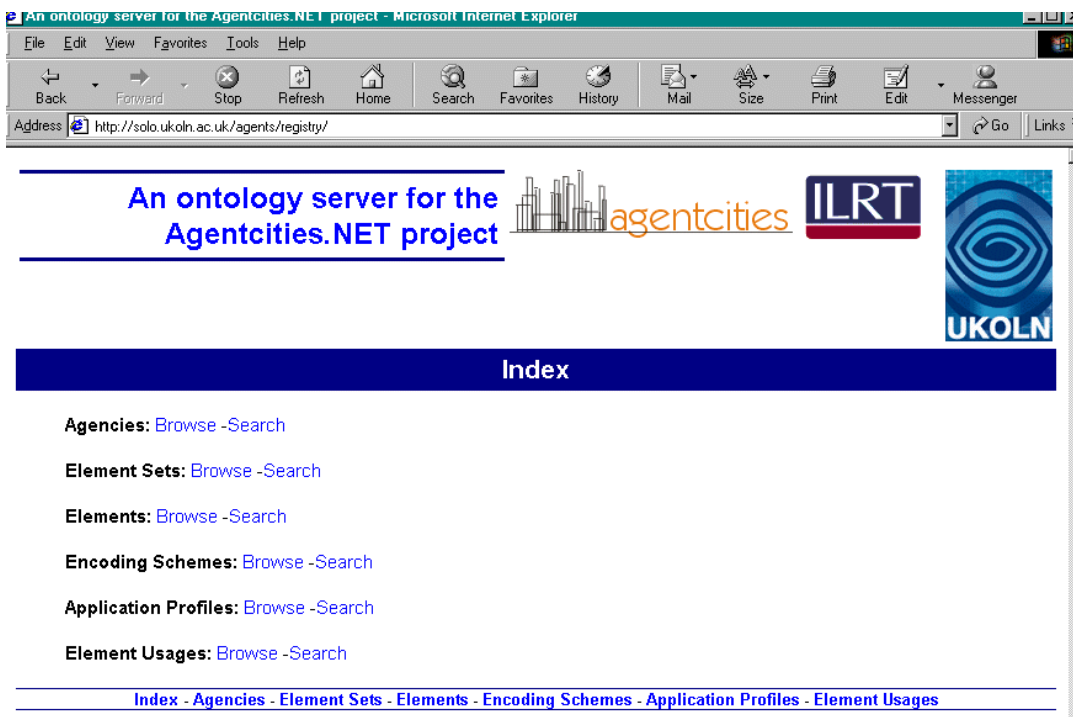
325 **4.1 Web Interface**

326 Independent of the agent interface, the Server can also be explored through a web interface,
 327 which is linked from the web page: <http://www.ukoln.ac.uk/metadata/agentcities/>.

328

329 The following screen shots illustrate browsing of the Server using a web browser:

330



331

Figure 1: The starting page for exploring the Server

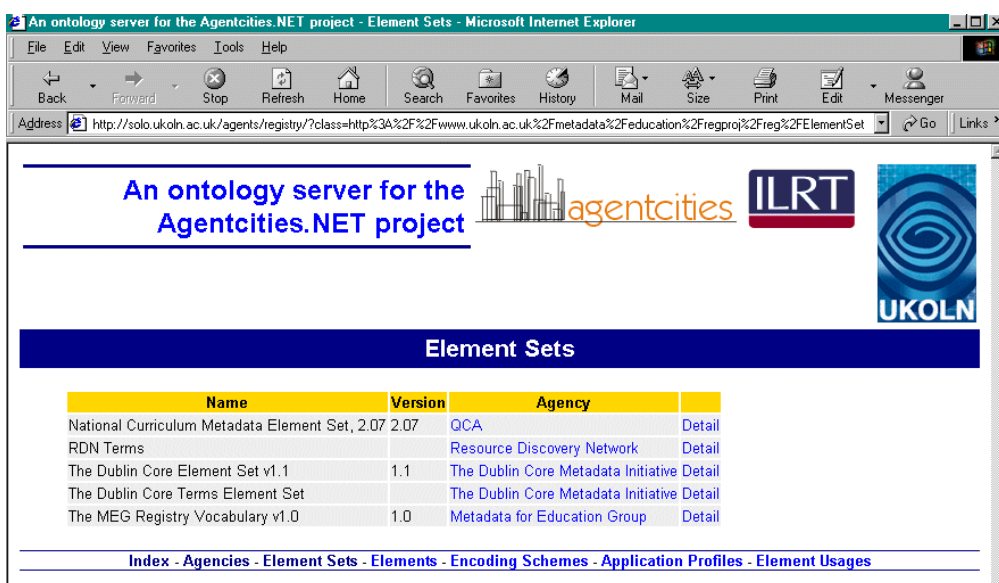
332

333

334

335 Browsing a category reveals a list of all the resources of that class, with links to further detail

336



337

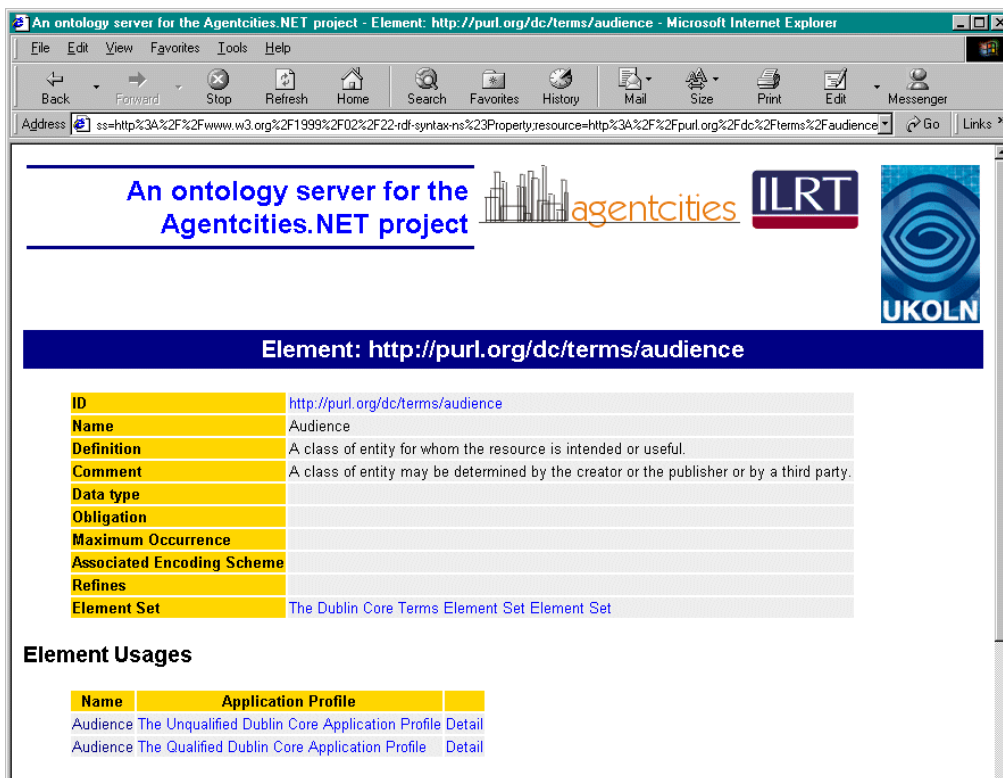
Figure 2: Browsing the list of all element sets in the Server

338

339

340 When browsing a specific resource, the details from the RDF description of that resource are
 341 displayed,

342 as well as links to related resources.



343
344 **Figure 3:** Looking at the details of a specific element
345

346 **4.2 The UKOLN Agent Platform**

347 Our implementation work has been carried out using the JADE agent platform. JADE is one
348 of the recommended platforms for developing agent systems. It is a software development
349 platform aimed at developing multi-agent systems and applications conforming to FIPA
350 standards for intelligent agents. It includes two main products, a FIPA-compliant agent
351 platform and a package to develop Java agents. JADE has provided the environment within
352 which to deploy the ontology service and for building agents.
353

354 Our platform has been registered with the platform directory at www.agencies.net. Our
355 platform name is ukoln.agencies.net[2].

356 **4.3 Overview of functionality**

357 The Server Agent runs on the UKOLN agent platform and communicates with the Server
358 using the Server API (over HTTP). It retrieves information on element sets and returns this
359 information in response to requests from other agents.
360

361 We have modified the APIs from the MEG Registry software to support search and browse
362 functions against agency, element set, element, application profile, element usage and
363 encoding scheme. Results are returned as RDF-encoded data, rather than HTML. This is
364 possible since the native store of the Server stores the element set descriptions as RDF, and
365 uses the Redland RDF toolkit within the HTTP APIs.
366
367

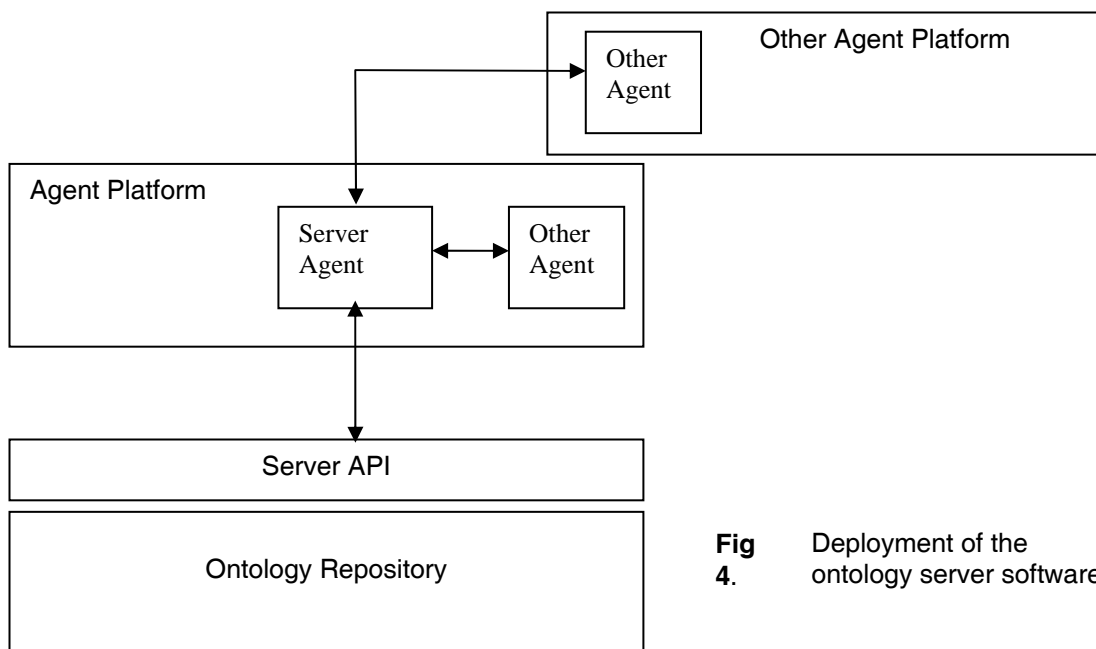


Fig 4. Deployment of the ontology server software

368
369
370

The Server Agent and two examples of requester agents are now described.

371 **4.3.1 The Server Agent**

372 The Server Agent can carry out search and browse requests on behalf of other agents, and passes on the
373 results from the Server to the requester agents.

374

375 **Search**

376 Searches are carried out within a specific category (e.g. agency or elements) and the search term is
377 matched with any part of the text between the RDF tags making up a description. If a part of the
378 description matches, the whole description for that resource is returned in the result set. When the
379 description is that of an element, the description of the associated element set is also presented.

380

381 **Browse**

382 Using the browse function, either a whole category is explored, or a specifically named
383 resource from a category is specified. The RDF descriptions for all the resources in a
384 category, or for a single resource are returned respectively.

385

386 Examples of the RDF (returned in response to both of these kinds of queries) are illustrated in
387 the following sections.

388

389 **Implementation**

390 *Behaviours*

391 The Server Agent is implemented using one behaviour. This behaviour is cyclic and will wait
392 for a message with a REQUEST performative. On receiving such a message, the behaviour

393

- 394 1. extracts components of the request (using an ontology)
- 395 2. constructs a URL from the request
- 396 3. connects to the Server using the URL
- 397 4. reads the response from the Server
- 398 5. places response into a reply message

399

400 Basic error checking is performed. Incorrect content or an unexpected performative will result in a
401 NOT_UNDERSTOOD message being returned to the sender. At present, other error conditions are
402 simply caught within the Java exception mechanism and reported on the System.err stream.

403

404 Thus the behaviour deals with one request at a time, sending a reply before attending to the
next request message in the agent queue.

405 A more complex model of behaviour, for example starting a new agent or behaviour to deal with each
 406 request, was unnecessary at this stage, given the simple functionality of the Server and the agent. In a
 407 service level Server, the issue of how to deal with a large number of requests in a responsive manner
 408 would become important. The performance of a large Server capable of complex querying would also
 409 have to be taken into account, but to date such registries are largely an unknown factor.

410 **4.3.2 Server Ontology**

411 We have defined a simple ontology (ServerSearchOntology) in which requests to the Server
 412 Agent can be expressed. This ontology is intended to encapsulate the simple kinds of
 413 requests supported by the Server that we have experimented with, and is not intended to be
 414 an exhaustive or comprehensive ontology for all the kinds of queries that schema registries
 415 should or could support.

416
 417 The ontology consists of two Action concepts, ReturnSearchResults and ReturnBrowseResults. The
 418 ReturnSearchResults action emulates a search request through a web browser; ReturnSearchResults has
 419 a searchRequest, made up of a Scope and a searchTerm. The scope limits the search for the
 420 searchTerm (which is a string) to one of the categories (agency etc.). ReturnBrowseResults emulates
 421 the browsing action carried out through the web browser. Thus a browseRequest takes a Scope (one of
 422 agency, element set, element, application profile, element usage and encoding schema) and a specific
 423 resource URI. The resource URI identifies a specific instance of the entity (e.g. a particular agency)
 424 and if a specific resource URI is specified in the browse request, the RDF description for that resource
 425 alone is returned.

426 If no resource URI is specified, the RDF descriptions of all the instances of that category are
 427 returned in a list (e.g. all the agencies are listed). The examples illustrate this behaviour.

428
 429 **Examples**

430 Example 1: An encoding of a **search** request for the **term** "network" within the **scope**
 431 "agency":

```
432 (
433     (action
434         (agent-identifier :name UKOLNServer@solo.ukoln.ac.uk:1099/JADE)
435         (ReturnSearchResults
436             (Search :Scope agency :SearchTerm network)
437         )
438     )
439 )
```

440
 441 The RDF description of an agency with the term resource in its name is returned:

```
442  

443 <rdf:Description rdf:about="http://purl.org/rdn/RDN/">
444   <rdf:type
445     rdf:resource="http://www.ukoln.ac.uk/metadata/education/regproj/reg/Agency"/
446   >
447     <reg:agencyName>Resource Discovery Network</reg:agencyName>
448     <reg:agencyHomepage rdf:resource="http://www.rdn.ac.uk/" />
449   </rdf:Description>
```

450
 451 Example 2: A **search** for the **term** "audience" in the **element** category .

```
452 (
453     (action
454         (agent-identifier :name UKOLNServer@solo.ukoln.ac.uk:1099/JADE)
455         (ReturnSearchResults
456             (Search : Scope element : SearchTerm audience)
457         )
458     )
459 )
```

460
 461 This search finds two elements. In the first element the search term 'audience' is found within the
 462 useComment tag. The second element is the Audience element in the Dublin Core (The search term is
 463

464 highlighted here for emphasis). Both these elements are part of the Dublin Core Terms element set and
 465 the description for the element set is returned at the end.

```

466 <rdf:Description rdf:about="http://purl.org/dc/terms/mediator">
467   <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
468 ns#Property"/>
469   <rdfs:label>Mediator</rdfs:label>
470   <rdfs:comment>A class of entity that mediates access to the resource and
471 for whom the resource is intended or useful.</rdfs:comment>
472   <reg:useComment>The audience for a resource in the education/training
473 domain are of two basic classes: (1) an ultimate beneficiary of the resource
474 (usually a student or trainee), and (2) frequently, an entity
475 that mediates access to the resource (usually a teacher or trainer). The
476 mediator element refinement represents the second of these two
477 classes.</reg:useComment>
478   <rdfs:subPropertyOf rdf:resource="http://purl.org/dc/terms/audience"/>
479   <reg:isElementOf
480 rdf:resource="http://www.ukoln.ac.uk/metadata/education/regproj/reg/elementS
481 et/dcterms"/>
482   </rdf:Description>
483
484 <rdf:Description rdf:about="http://purl.org/dc/terms/audience">
485   <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
486 ns#Property"/>
487   <rdfs:label>Audience</rdfs:label>
488   <rdfs:comment>A class of entity for whom the resource is intended or
489 useful.</rdfs:comment>
490   <reg:useComment>A class of entity may be determined by the creator or
491 the publisher or by a third party.</reg:useComment>
492   <reg:isElementOf
493 rdf:resource="http://www.ukoln.ac.uk/metadata/education/regproj/reg/elementS
494 et/dcterms"/>
495   </rdf:Description>
496
497 <rdf:Description
498 rdf:about="http://www.ukoln.ac.uk/metadata/education/regproj/reg/elementSet/
499 dcterms">
500   <rdf:type
501 rdf:resource="http://www.ukoln.ac.uk/metadata/education/regproj/reg/ElementS
502 et"/>
503   <dc:title>The Dublin Core Terms Element Set</dc:title>
504   <dcterms:created>2000-07-11</dcterms:created>
505   <reg:status>DCMI recommendation</reg:status>
506   <dc:description>
507
508
509 The Dublin Core metadata vocabulary is a simple vocabulary intended to facilitate discovery
510 of resources.
511
512   </dc:description>
513   <reg:responsibleAgency
514 rdf:resource="http://www.ukoln.ac.uk/metadata/education/regproj/reg/agency/d
515 cmi"/>
516   <reg:xmlNamespacePrefix>dcterms:</reg:xmlNamespacePrefix>
517   <reg:specification
518 rdf:resource="http://dublincore.org/usage/terms/terms-latest.html"/>
519   </rdf:Description>
520
521 Example 3:A browse request for the whole of the agency category (no Resource URI is
522 given)
523
524 (
525   (action
526     (agent-identifier :name UKOLNServer@solo.ukoln.ac.uk:1099/JADE)
527     (ReturnBrowseResults
528       (Browse :Scope agency :Resource ""))

```

```

529         )
530     )
531 )
532
533 Returns a list of all the agencies (descriptions encoded in RDF)
534
535 <rdf:Description
536 rdf:about="http://www.ukoln.ac.uk/metadata/education/regproj/reg/agency/iso"
537 >
538     <rdf:type
539 rdf:resource="http://www.ukoln.ac.uk/metadata/education/regproj/reg/Agency"/
540 >
541     <reg:agencyName>International Standards Organisation</reg:agencyName>
542 </rdf:Description>
543
544 <rdf:Description
545 rdf:about="http://www.ukoln.ac.uk/metadata/education/regproj/reg/agency/lc">
546     <rdf:type
547 rdf:resource="http://www.ukoln.ac.uk/metadata/education/regproj/reg/Agency"/
548 >
549     <reg:agencyName>Library of Congress</reg:agencyName>
550 </rdf:Description>
551
552     <rdf:Description
553 rdf:about="http://www.ukoln.ac.uk/metadata/education/regproj/reg/agency/meg"
554 >
555     <rdf:type
556 rdf:resource="http://www.ukoln.ac.uk/metadata/education/regproj/reg/Agency"/
557 >
558     <reg:agencyName>Metadata for Education Group</reg:agencyName>
559     <reg:agencyHomepage
560 rdf:resource="http://www.ukoln.ac.uk/metadata/education"/>
561 </rdf:Description>
562
563     <rdf:Description
564 rdf:about="http://www.ukoln.ac.uk/metadata/education/regproj/reg/agency/oclc"
565 ">
566     <rdf:type
567 rdf:resource="http://www.ukoln.ac.uk/metadata/education/regproj/reg/Agency"/
568 >
569     <reg:agencyName>OCLC</reg:agencyName>
570 </rdf:Description>
571
572 <rdf:Description rdf:about="http://purl.org/rdn/RDN/">
573     <rdf:type
574 rdf:resource="http://www.ukoln.ac.uk/metadata/education/regproj/reg/Agency"/
575 >
576     <reg:agencyName>Resource Discovery Network</reg:agencyName>
577     <reg:agencyHomepage rdf:resource="http://www.rdn.ac.uk/" />
578 </rdf:Description>
579
580 (elided)
581
582     <rdf:Description
583 rdf:about="http://www.ukoln.ac.uk/metadata/education/regproj/reg/agency/dcmi"
584 ">
585     <rdf:type
586 rdf:resource="http://www.ukoln.ac.uk/metadata/education/regproj/reg/Agency"/
587 >
588     <reg:agencyName>The Dublin Core Metadata Initiative</reg:agencyName>
589     <reg:agencyHomepage rdf:resource="http://dublincore.org/" />
590 </rdf:Description>
591
592     <rdf:Description
593 rdf:about="http://www.ukoln.ac.uk/metadata/education/regproj/reg/agency/w3">

```

```

594     <rdf:type
595     rdf:resource="http://www.ukoln.ac.uk/metadata/education/regproj/reg/Agency"/
596     >
597     <reg:agencyName>World Wide Web Consortium</reg:agencyName>
598     </rdf:Description>
599
600

```

601 **Example 4:** A browse request for a specific resource (<http://purl.org/dc/terms/MESH/>) from the
602 encoding scheme category.

```

603
604 (      (action
605         (agent-identifier :name UKOLNServer@solo.ukoln.ac.uk:1099/JADE)
606         (ReturnBrowseResults
607          (Browse           :Scope encodingscheme           :Resource
608           http://purl.org/dc/terms/MESH/)
609          )
610        )
611 )

```

612 4.4 Interrogating the Server Agent

613 We have implemented two examples of Requester Agents, both of which are driven by a human user
614 and make requests to the Server Agent. These two agents use the ServerSearchOntology to
615 communicate requests to the Server Agent, and display the response returned by the Server. Results to
616 queries are contained within the content slot of an INFORM message from the Server Agent, and
617 consist of RDFS descriptions. Thus the ontology is only used to communicate requests; responses are
618 simply wrapped up in the content slot of the message.

619 4.4.1 The GUI Agent

620 This agent presents the user with a graphical interface implemented with Java Swing. This is realized
621 through two classes:

622 ServerAgentGui class extends the Swing JFrame class, and defines the appearance
623 of the interface;
624 ServerGuiAgent class extends the Jade GuiAgent class, and defines the behaviours
625 that are instantiated in response to user actions at the interface.

626 Each instance of the Agent class is associated with one instance of the Gui class (and vice versa).
627

628 The appearance of the interface is shown in Figure 5. It contains the following main
629 components:

- 630 • a pull-down list of categories
- 631 • a button for triggering the display of a whole category (the latter obscured in the first
632 screenshot)
- 633 • a text entry for resource URIs, and an associated button for displaying;
- 634 • a text entry for search terms, with an associated search button
- 635 • a display area for results

636
637 After selecting a category, the user can then choose to browse the whole category, or to enter a resource
638 URI for a known resource. Alternatively, the search box can be used to interrogate the Server. The
639 three tasks that the interface supports reflect the kinds of requests that can be expressed in the
640 ServerSearchOntology:

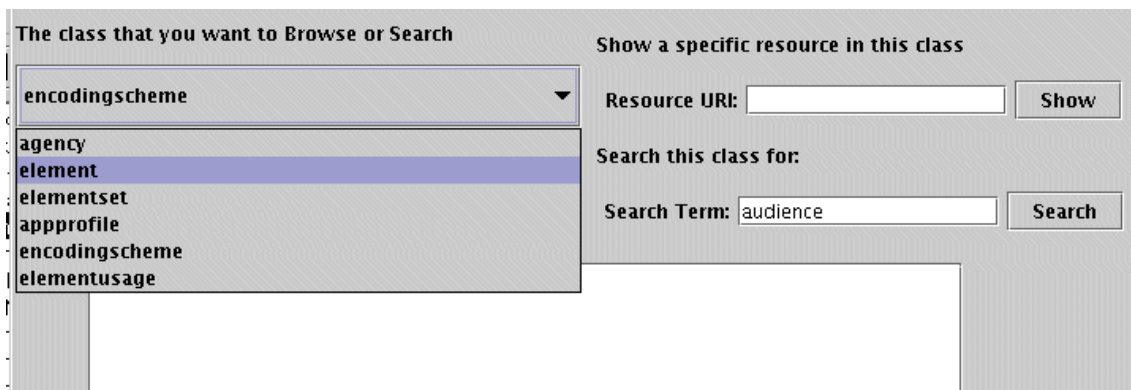
641
642 The ServerAgentGui class implements ActionListener; on Action events, the handler
643 (ActionPerformed) invokes the JADE postGUIEvent method to communicate with the
644 ServerGuiAgent class; this is the path by which user actions on the interface trigger
645 behaviours in the agent.

646
647 Within the agent, the onGuiEvent method handles the events from the interface (invoked
648 through postGuiEvent). A message is built (using the ServerSearchOntology) corresponding
649 to the action invoked; the message is sent using a SenderBehaviour (which extends
650 OneShotBehaviour). A cyclic behaviour listens for response messages from the ServerAgent

651 and when an INFORM message arrives, it invokes a displayResults method in the gui, so that
 652 the content of the message (containing RDF-encoded descriptions) is displayed (Figure 6).
 653

654 The interface has been design to support one outstanding request at a time. In theory
 655 multiple requests could be launched before the first response arrives, and at present there is
 656 no control to prevent this. In practice the system response is sufficiently fast that no major
 657 control is required at present to synchronise requests and responses. If such control were
 658 required, this could best be implemented through the Gui by disabling the sending controls
 659 until a response is received. An alternative would be an interface that supported multiple
 660 outstanding requests, but this would require a more complicated design that is beyond the
 661 scope of the present project. This also requires a more complicated coordination model
 662 between the interface and the agent(s) for managing requests.
 663

664 The link between the ServerGUIAgent and the Server Agent is hardwired and the Server
 665 Agent is assumed to be running locally.



666
 667
 668
 669
 670
 671

Figure 5. Using the interactive GUI of the ServerGuiAgent to enter requests

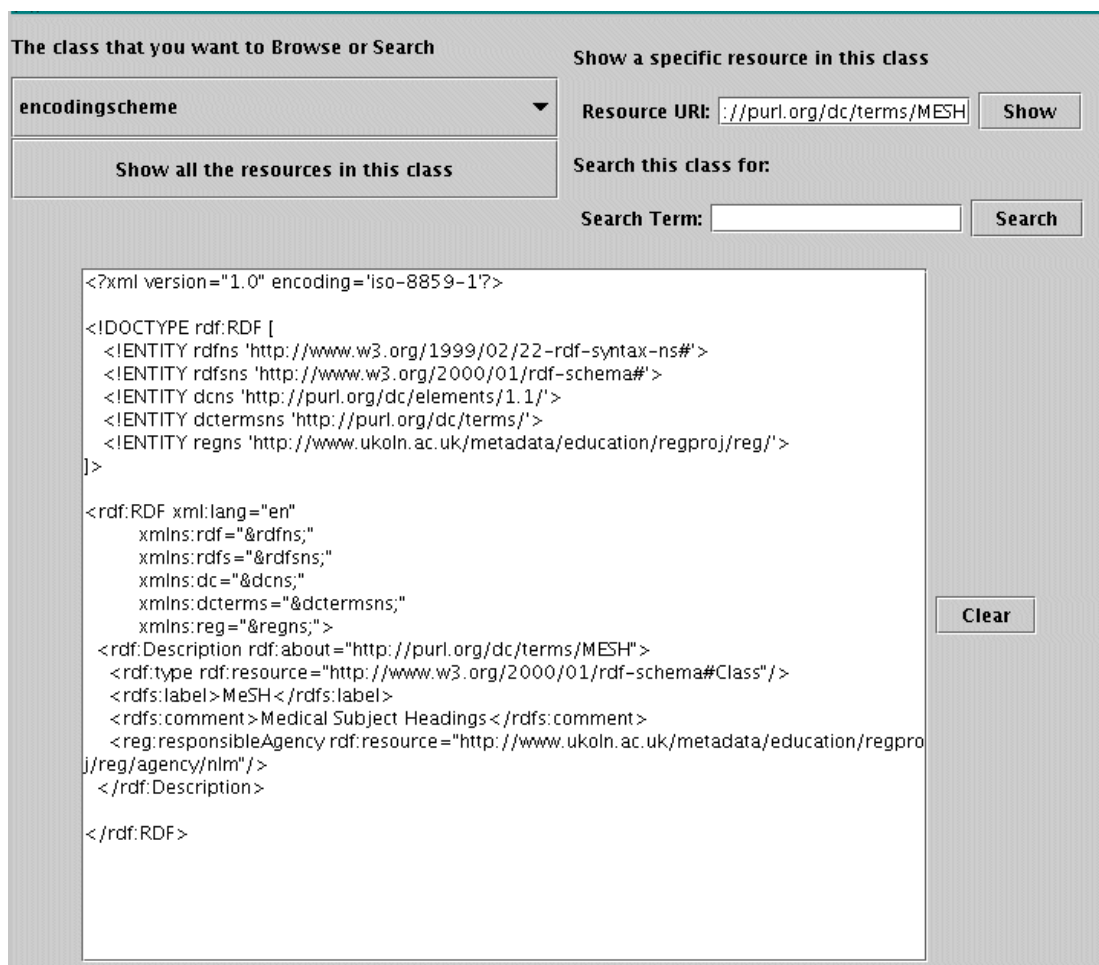


Figure 6. Results are displayed in a window in the GUI.

672
673
674

675 **4.4.2 The Command Line agent**

676 A second Agent Class, ServerRequesterAgent, has been provided to interact with the user
677 through the command line. On setup() this agent first establishes which Server the user
678 would like to use, with a choice of either the UKOLN Server, or a local one.

679 **4.4.3 Behaviours**

680 The Agent then instantiates a main sequential behaviour (HandleRequestsBehaviour) which
681 prompts for and reads input from the terminal. The onStart() method of the main behaviour
682 interacts with the user to define what kind of transaction the user is performing (browse or
683 search) and its parameters: scope, search term or resource URI:

```
684
685 ENTER the local name of the Server agent or press enter to use the
686 UKOLN Server-->
687 ENTER s for search or b for browse -->
688 s
689 Class to Search ---> element
690 Enter a SearchTerm ---> audience
691
```

692 A suitable message is then built and a Sender Behaviour is scheduled (as a sub behaviour) to
693 send the message to the Server Agent. The next subBehaviour added then handles the
694 response from the Server Agent and displays the result to the user.

695 The onEnd() method then checks if the user would like to carry out another transaction. If the
696 user stops, the agent is terminated; if the user wishes to continue, all the behaviours are
697 reset.

698 **5 Conclusions**

699 We have successfully deployed an ontology server onto the Agencities.NET network, where it is
700 available for either browsing over the Web or querying by agents. It should be noted that the server
701 accepts metadata vocabularies encoded in RDF Schema. Further, the vocabularies need to adhere to
702 the model described in the Appendix. The work presented has advanced the work begun in previous
703 projects to investigate an approach based on automated querying and processing of simple ontologies
704 by software agents rather than through human interaction.

705 **6 Acknowledgements**

706 The software used in this project for the ontology server was originally developed in the MEG Registry
707 project which was funded by JISC and BECTa. The ideas in this project have been developed from
708 work in the DESIRE, SCHEMAS and MEG Registry Projects. Thanks to Pete Johnston, UKOLN,
709 University of Bath, for help with the MEG Registry and associated software. Thanks also to Owen
710 Cliff, Department of Computer Science, University of Bath for assistance with setting up the UKOLN
711 server.

712
713 UKOLN is funded by Resource: The Council for Museums, Archives & Libraries, the Joint
714 Information Systems Committee (JISC) of the Higher and Further Education Funding Councils, as well
715 as by project funding from the JISC and the European Union. UKOLN also receives support from the
716 University of Bath where it is based.

717 7 References

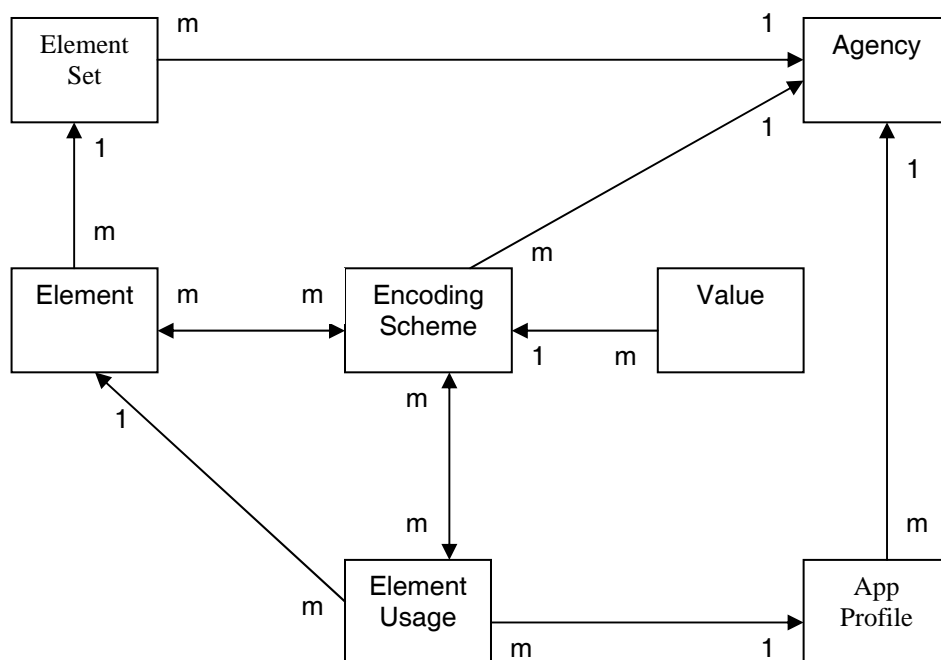
- 718 [1] UKOLN web-page for Agentcities.NET deployment project
719 <http://www.ukoln.ac.uk/metadata/agentcities/>
720
- 721 [2] UKOLN Ontology Server for the Agentcities.NET network
722 <http://solo.ukoln.ac.uk/agents/server/>
723
- 724 [3] UKOLN
725 <http://www.ukoln.ac.uk/>
726
- 727 [4] Agentcities.NET
728 <http://www.agentcities.org/EUNET/>
729
- 730 [5] Dublin Core Metadata Initiative (DCMI)
731 <http://www.dublincore.org/>
732
- 733 [6] DESIRE
734 <http://www.ukoln.ac.uk/metadata/desire/>
735
- 736 [7] SCHEMAS Project
737 <http://www.schemas-forum.org/>
738
- 739 [8] Thomas Baker, Makx Dekkers, Rachel Heery, Manjula Patel, Gauri Salokhe, *What Terms*
740 *Does Your Metadata Use? Application Profiles as Machine Understandable Narratives*,
741 *Journal of Digital Information* Vol 2 (2), November 2001
742 <http://jodi.ecs.soton.ac.uk/Articles/v02/i02/Baker/baker-final.pdf>
743
- 744 [9] Heery H and Patel M., *Application Profiles: mixing and matching metadata schemas*,
745 *Ariadne* Issue 25, September 2000
746 <http://www.ariadne.ac.uk/issue25/app-profiles/intro.html>
747
- 748 [10] OntoWeb Technical Roadmap v 1.0
749 <http://babage.dia.fi.upm.es/ontoweb/wp1/OntoRoadMap/index.html>
750
- 751 [11] Numbered Hypernotes in J.Hendler
752 [http://www.sciencemag.org/cgi/content/full/299/5606/520?ijkey=1BUgJQXW4nU7Q&keytype=](http://www.sciencemag.org/cgi/content/full/299/5606/520?ijkey=1BUgJQXW4nU7Q&keytype=ref&siteid=sci)
753 [ref&siteid=sci](http://www.sciencemag.org/cgi/content/full/299/5606/520?ijkey=1BUgJQXW4nU7Q&keytype=ref&siteid=sci)
754 [http://www.sciencemag.org/cgi/content/full/299/5606/520?ijkey=1BUgJQXW4nU7Q&keytype=](http://www.sciencemag.org/cgi/content/full/299/5606/520?ijkey=1BUgJQXW4nU7Q&keytype=ref&siteid=sci#note9)
755 [ref&siteid=sci#note9](http://www.sciencemag.org/cgi/content/full/299/5606/520?ijkey=1BUgJQXW4nU7Q&keytype=ref&siteid=sci#note9)
756
- 757 [12] Sowa, J.F. Building, Sharing and Merging Ontologies: Glossary
758 <http://www.jfsowa.com/ontology/ontoshar.htm#s6>
759
- 760 [13] SCHEMAS Project Glossary
761 <http://www.schemas-forum.org/info-services/d74.htm>
762
- 763 [14] MEG Website
764 <http://www.ukoln.ac.uk/metadata/education/>
765
- 766 [15] MEG Registry Project
767 <http://www.ukoln.ac.uk/metadata/education/regproj/>
768
- 769 [16] Beckett D., *The Design and Implementation of the Redland RDF Application Framework*
770 *Proceedings of WWW10, Hong Kong, May 2-5 2001*
771 <http://www10.org/cdrom/papers/frame.html>
772
- 773 [17] *The MEG Registry and SCART: complementary tools for creation, discovery and re-use*
774 *of metadata schemas*. Rachel Heery, Pete Johnston, Dave Beckett (ILRT, University of

- 775 Bristol) & Damian Steer (ILRT, University of Bristol) - October 2002
776 In: Proceedings of the International Conference on Dublin Core and Metadata for e-
777 Communities, 2002. Florence: Firenze University Press, 2002, pp. 125-132.
778 <http://www.bncf.net/dc2002/program/ft/paper14.pdf>
779
780 [18] RDF Schemas
781 <http://www.w3.org/TR/rdf-schema/>
782 [Note: this is the latest version of the W3C Working Draft, released on 12 November 2002,
783 which is a work in progress; The registry development took place before the release of this
784 draft.]
785
786 [19] W3C Web Ontology Working Group
787 <http://www.w3.org/2001/sw/WebOnt/>
788
789 [20] SWAD-Europe: Scalability and Storage: Survey of Free Software /Open Source RDF
790 storage systems
791 http://www.w3.org/2001/sw/Europe/reports/rdf_scalable_storage_report/
792
793 [21] DAML+OIL
794 <http://www.w3.org/TR/daml+oil-reference>
795
796 [22] OWL
797 <http://www.w3.org/TR/owl-ref/>
798
799 [23] OWL2
800 <http://www.w3.org/TR/2003/WD-webont-req-20030203/>
801
802 [24] Wilson, Michael
803 http://www.w3c.rl.ac.uk/pasttalks/slidemaker/EPS_DTI/Overview.html
804
805 [25] SemanticWeb.org
806 <http://www.semanticweb.org/knowmarkup.html#ontologies>
807
808 [26] Hendler Web version of IEEE article
809 <http://www.cs.umd.edu/~hendler/AgentWeb.html>
810
811 [27] WordNet
812 <http://www.semanticweb.org/library/>
813
814 [28] RDFS(FA)
815 <http://dl-web.man.ac.uk/rdfsfa/>
816
817 [29] The Meg Registry Client Software (SCART)
818 <http://www.ukoln.ac.uk/metadata/education/regproj/scart/>
819
820 [30] DAML Repository
821 <http://www.daml.org/ontologies/>
822
823 [31] BT Ontology Server
824 <http://193.113.27.14/ontology-server-demo/index.jsp>
825
826 [32] BT Ontology Server Service Description
827 <http://193.113.27.14/services/OntologyService/ServiceDescription.htm>
828
829 [33] Another Ontology Page
830 http://burningluggage.com/another_ontology_page/aop.htm
831
832 [34] SCHEMAS: Best practice guidelines for managing a registry
833 <http://www.schemas-forum.org/info-services/d52.htm>
834

- 835 [35] DCMI Registry Working Group
836 <http://uk.dublincore.org/groups/registry/>
837
838 [36] OWL Overview
839 <http://www.w3.org/TR/2002/WD-owl-features-20020729/>
840
841 [37] EOR (Extensible Open RDF) Toolkit
842 <http://eor.dublincore.org/>
843

844 **Appendix: The MEG Registry Data Model**

845



846

847

848 **Agency:** An organisation or individual responsible for managing one or more Element Sets,
 849 Application Profiles or Encoding Schemes

850

851 Relationships

852

853 Element Set → is-Managed-By (m-1) → **Agency**

854 Encoding Scheme → is-Managed-By (m-1) → **Agency**

855 Application Profile → is-Managed-By (m-1) → **Agency**

856

857 Agency Properties

858

Identifier (URI)

Name The name or title of the Agency

Home Page URL A source of further info about the Agency

859

860 **Element Set:** A set of metadata Elements that is managed as a coherent unit by an Agency.

861 The Elements of an Element Set are “functionally” related, by virtue of having been defined

862 for the purpose of usefully describing the characteristics of a resource

863

864 Relationships

865

866 **Element Set** → is-Managed-By (m-1) → Agency

867 Element → is-Element-Of (m-1) → **Element Set**

868

869

870

871

872 Element Set Properties

873

Identifier (URI)

Title The name or title of the Element Set

Version	The version of the Element St
Date created	Date this version created
Status	Draft/recommendation etc
Description	Including any notes of scope/purpose
Classification	
Specification	Prose description of/guidelines for use of Element Set

874

875 **Element:** *A formally defined term that is used to describe a characteristic or attribute of a*
 876 *resource*

877

878 Relationships

879

880 **Element** → is-Element-Of (m-1) → Element Set

881 **Element** → associated-Encoding-Scheme (m-m) → Encoding Scheme

882 **Element** → refines (m-1) → **Element**

883 Element Usage → uses (m-1) → **Element**

884

885 Element Properties

886

Identifier (URI)	
Name	A human-readable version of the property name
Definition	A statement that clearly represents the concept and essential nature of the Element
Comment	A remark concerning the application/use of the data element
Data type	Indicates the type of data that can be represented in the value of the data element
Obligation	Indicates whether the Element is always or sometimes required to be present
Maximum occurrence	Indicates any limit to the repeatability of the Element

887

888 **Encoding Scheme:** *A set of contextual information or parsing rules that aids in the*
 889 *interpretation of the value of a metadata Element. Encoding Schemes include*

- 890 • *controlled vocabularies, which enumerate a list of values, and;*
- 891 • *formal notations or parsing rules, which define precisely how a lexical representation of a*
 892 *value is to be interpreted*

893

894 Relationships

895 **Encoding Scheme** → is-Managed-By Agency (m-1) → Agency

896 Element → associated-Encoding-Scheme (m-m) → **Encoding Scheme**

897 Element Usage → associated-Encoding-Scheme (m-m) → **Encoding Scheme**

898 Value –type (m-1) → **Encoding Scheme**

899

900 Encoding Scheme Properties

901

Identifier (URI)	
Name	The name or title of the Encoding Scheme
Version	The version of the Encoding Scheme
Date created	Date this version created
Status	Draft/recommendation etc
Description	Including any notes of scope/purpose
Classification	
Specification	Prose description of/guidelines for use of Encoding Scheme

902

903 **Controlled Vocabulary Value:** *An individual value or term in a controlled vocabulary*

904

905 Relationships

906

907 **Value** → type (m-1) → Encoding Scheme

908

Identifier (URI)

Value	Value
Label	Human-readable form of value
Description	Explanation or definition of value

909

910 **Application Profile:** *A set of Element Usages that is managed as a coherent unit by an*
 911 *Agency. An Application Profile is optimised for the resource description requirements of a*
 912 *particular application or context.*
 913 *Like the Elements of an Element Set, the Element Usages within an Application Profile are*
 914 *“functionally” related, by virtue of having been defined for the purpose of usefully describing a*
 915 *resource.*
 916 *Within an Application Profile, the Element Usages may reference Elements from multiple*
 917 *Element Sets*

918 Relationships

920

921 **Application Profile** → is-Managed-By Agency (m-1) → Agency
 922 Element Usage → is-Usage-In (m-1) → **Application Profile**

923

924 Application Profile Properties

925

Identifier (URI)	
Title	The name or title of the Application Profile
Version	The version of the Application Profile
Date created	Date this version created
Status	Draft/recommendation etc
Description	Including any notes of scope/purpose
Classification	
Associated XML Schema Specification	Prose description of/guidelines for use of Application Profile

926

927 **Element Usage:** *A deployment of a (previously defined) metadata Element in the context of a*
 928 *particular domain or application. The used Element may be tailored for the context by:*
 929

- *a narrowing of its semantic definition;*
- *association with specified datatypes or Encoding Schemes;*
- *specification of obligation/occurrence constraints*

932

933 Relationships

934

935 **Element Usage** → is-Usage-In (m-1) → Application Profile
 936 **Element Usage** → uses (m-1) → Element
 937 **Element Usage** → associated-Encoding-Scheme (m-m) → Encoding Scheme

938

939 Element Usage Properties

940

Identifier (URI)	
Name	A human-readable version of the Element name.
Definition	A statement that clearly represents the concept and essential nature of the Element
Comment	A remark concerning the application/use of the Element.
Data type	Indicates the type of data that can be represented in the value of the Element
Obligation	Indicates whether the Element is always or sometimes required to be present
Maximum occurrence	Indicates any limit to the repeatability of the Element

941

942