

**ON
knowledge
TO**

Ontology Middleware Implementation

(OMM Development)

Atanas Kiryakov, Damyan Ognyanov, Borislav Popov

OntoText Lab.

Identifier	39
Class	Deliverable
Version	1.1
Version date	September 2002
Status	Final
Distribution	Public
Responsible Partner	OntoText Lab., Sirma AI Ltd.

On-To-Knowledge Consortium

This document is part of a research project funded by the IST Programme of the Commission of the European Communities as project number IST-1999-10132. The partners in this project are: Vrije Universiteit Amsterdam (VU) (co-ordinator), NL; the University of Searchrlsrue, Germany; Schweizerische Lebensversicherungs- und Rentenanstalt / Swiss Life, Switzerland; British Telecommunications plc, UK; CognIT a.s, Norway; EnerSearch AB, Sweden; AIdministrator Nederland BV, NL; OntoText Lab., Sirma AI EOOD, Bulgaria.

Vrije Universiteit Amsterdam (VU)

Faculty of Sciences, Division of Mathematics and
Computer Science
De Boelelaan 1081a
1081 HV Amsterdam, the Netherlands
Fax and Answering machine: +31-(0)20-872 27 22
Mobil phone: +31-(0)6-51850619
Contactperson: Dieter Fensel
E-mail: dieter@cs.vu.nl

University of Karlsruhe

Institute AIFB
Searchiserstr. 12
D-76128 Searchrlsrue, Germany
Tel: +49-721-608392
Fax: +49-721-693717
Contactperson: R. Studer
E-mail: studer@aifb.uni-searchrlsrue.de

Schweizerische Lebensversicherungs- und Rentenanstalt / Swiss Life

Swiss Life Information Systems Research Group
General Guisan-Quai 40
8022 Zürich, Switzerland
Tel: (41 1) 284 4061, Fax: (41 1)284 6913
Contactperson: Ulrich Reimer
E-mail: Ulrich.Reimer@swisslife.ch

British Telecommunications plc

BT Adastral Park
Martlesham Heath
IP5 3RE Ipswich, UK
Tel: (44 1473)605536, Fax: (44 1473)642459
Contactperson: John Davies
E-mail: John.nj.Davies@bt.com

CognIT a.s

Busterudgt 1.
N-1754 Halden, Norway
Tel: +47 69 1770 44, Fax: +47 669 006 12
Contactperson: Bernt. A. Bremdal
E-mail: bernt@cognit.no

EnerSearch AB

SE 205 09 Malmö, Sweden
Tel: +46 40 25 58 25; Fax: +46 40 611 51 84
Contactperson: Hans Ottosson
E-mail: hans.ottosson@enersearch.se

AIdministrator Nederland BV

Julianaplein 14B
3817CS Amersfoort, NL
Tel: (31-33)4659987, Fax: (31-33)4659987
Contactperson: Jos van der Meer
E-mail: Jos.van.der.Meer@aidministrator.nl

OntoText Lab.

Sirma AI EOOD - Artificial Intelligence Labs
38A Chr. Botev blvd, 1000 Sofia, Bulgaria
Tel: +359 2 981 23 38; Fax: +359 2 981 90 58
Contactperson: Atanas Kiryakov
E-mail: Atanas.Kiryakov@sirma.bg

Abstract

This document presents the implementation of the Ontology Middleware Module (OMM) after the analysis and design presented in Deliverable 38, [Kiryakov et al, 2002]. It provides short overview of the principle definition of the OMM and the Knowledge Control System, which is a substantial part of it. The most part of the document provides information for the installation, sample knowledge bases, test cases and scenarios, implementation details and performance considerations.

Acknowledgements

The research reported here was carried out in the course of the On-To-Knowledge project. This project is partially funded by the IST Programme of the Commission of the European Communities as project number IST-1999-10132. The partners in this project are: Vrije Universiteit Amsterdam VUA (coordinator, NL), University of Karlsruhe (Germany), Swiss Life (Switzerland), BT plc (UK), CognIT a.s. (Norway), EnerSearch AB (Sweden), AIdministrator Nederland BV (NL), OntoText Lab. (BG). We wish to particularly thank to the Arjohn Kampman, Jeen Broekstra for the instant support and most specially for the timely implementation of the truth maintenance system of Sesame.

Contents

1. Introduction	5
1.1. Ontology Middleware Overview	5
1.2. Architecture and Interfaces	6
1.2.1. Overview of the SESAME Architecture	6
1.2.2. How OMM Fits in the Picture?	7
2. Installation.....	9
3. Skills Example.....	10
4. Major Test Cases	10
4.1. Work with a Previous Version.....	10
4.2. Play with the Security	11
4.2.1. Demo Security Setup.....	11
4.2.2. Repository Restriction Demo	12
4.2.3. Schema Restriction Demo	12
4.2.4. Classes Restriction Demo.....	12
4.2.5. Classes-Over-Schema Restriction Demo.....	14
4.2.6. Properties Restriction Demo.....	14
4.2.7. Pattern Restriction Demo.....	14
5. Implementation Details.....	15
5.1. Tracking Changes Implementation.....	15
5.2. Access Control Implementation.....	17
6. Performance comments	18
7. Conclusion	18
8. References	18
Appendix A: Limitations of the Implementation	20
A.1. Tracking of changes vs. Versioning and Temporality.....	20
A.2. Meta Knowledge about Ontologies	20

1. Introduction

This document presents the implementation of the Ontology Middleware Module (OMM) – the appropriate analysis and design issues are presented in Deliverable 38, [Kiryakov et al, 2002], which is a necessary basis for proper understanding of this report.

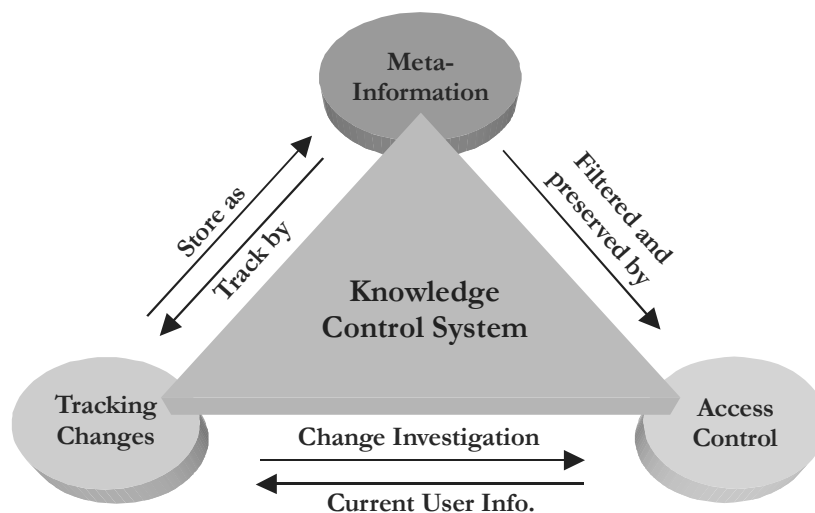
A short overview of the definition and design of OMM is presented in this section followed by presentation of the architecture on a technical level. The second section provides information about the installation of OMM, the third section presents the example used for testing and evaluation purposes. The test cases and demonstration scenarios which demonstrate the most important features of OMM are presented in section 4. Comments on the implementation are available in section 5 followed by performance information in section 6. The last section provides a short conclusion.

1.1. Ontology Middleware Overview

The middleware can be seen as „administrative“ software infrastructure that makes the results of the On-To-Knowledge project easier for integration in real-world applications. The central issue is to make the methodology and modules available to the society in a shape that allows easier development, management, maintenance, and use of middle-size and big knowledge bases¹. In the light of these objectives the following main features are supported:

- Versioning (tracking changes) of knowledge bases;
- Access control (security) system;
- Meta-information for knowledge bases.

These three aspects are tightly interrelated among each other as depicted on the following scheme:



The composition of the three functions above represents a *Knowledge Control System (KCS)* that provides the knowledge engineers with the same level of control and manageability of the knowledge in the process of its development and maintenance as the source control systems (such as CVS) provide for the software. However, KCS is not only limited to support the knowledge engineers or developers – from the perspective of the end-user applications, KCS can be seen as

¹ A knowledge base can consist of ontology and/or instance data and application specific knowledge.

equivalent to the database security, change tracking (often called cataloguing) and auditing systems. The KCS is be carefully designed so to support these two distinct use cases.

A fully-functional *Ontology Middleware* system should serve as a flexible and extendable platform for knowledge management solutions. It has to provide infrastructure with at least the following features:

- A repository providing the basic storage services in a scalable and reliable fashion. This role is already fulfilled by SEASME.
- Knowledge control – the KCS introduced above.
- Multi-protocol client access to allow different users and applications to use the system via the most efficient “transportation” media. This aspect is discussed in the next subsection.
- Support for plugable reasoning modules suitable for various domains and applications. This ensures that within a single enterprise or computing environment one and the same system may be used for various purposes (that require different reasoning services and expressivity) so enabling easy integration, interoperability between applications, knowledge maintenance and reuse.

The design of the ontology middleware module presented here is just an extension of the SESAME architecture (see [Broekstra and Kampman, 2001b]).

1.2. Architecture and Interfaces

Ontology Middleware Module (OMM) is designed to extend the existing functionality of the SESAME RDF(S) repository enriching it with support for versioning (tracking changes), meta-information, access control (security), and DL reasoning.

Here we present an overview of the enhancements to the Sesame – how the architecture was extended, which are the new and the modified modules.

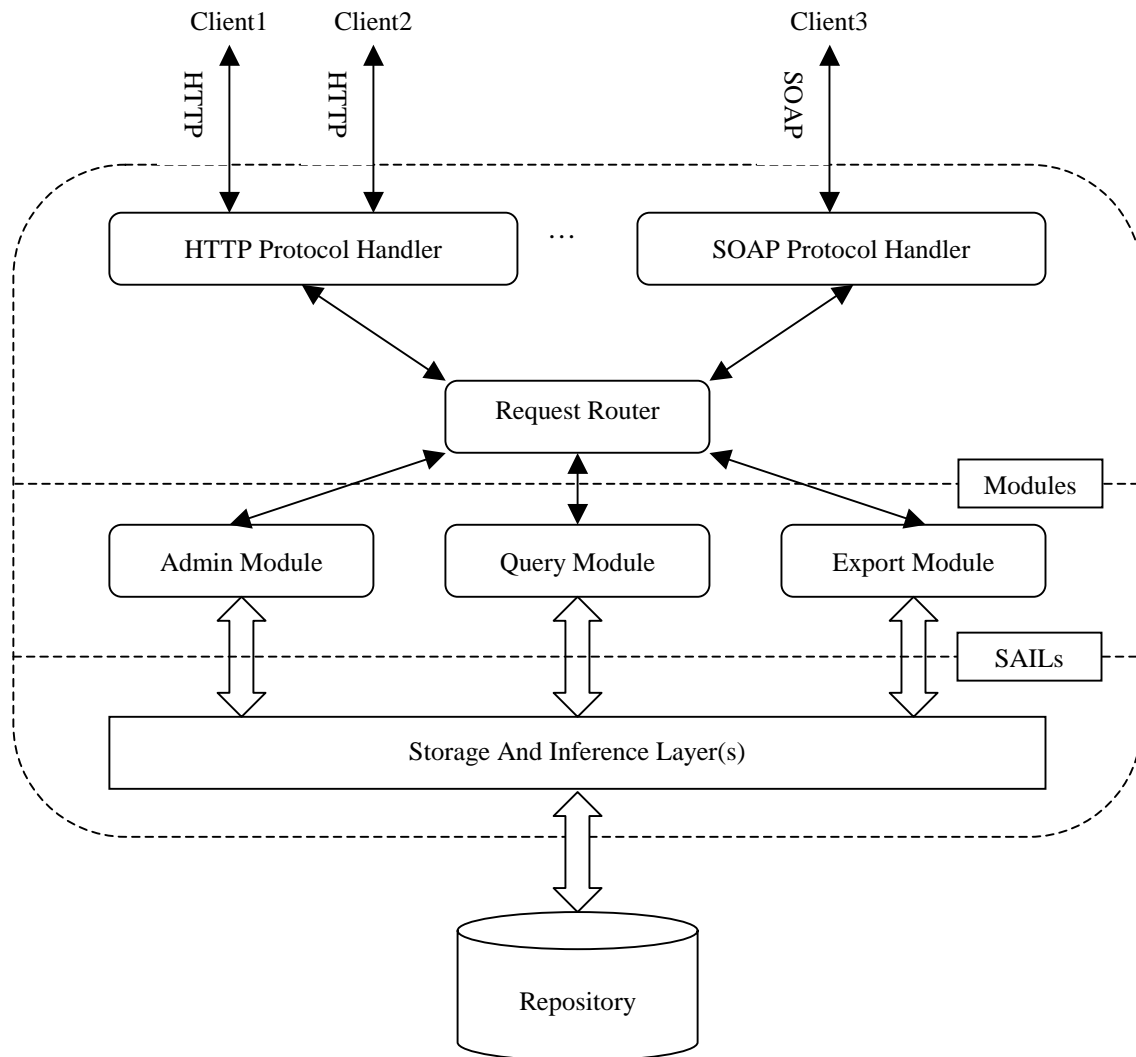
1.2.1. Overview of the SESAME Architecture

The SESAME architecture is composed of several layers. The access layer is responsible to provide all the necessary functionality managing the relations with the users, i.e. the front-end applications.

The Request Router manages the necessary marshalling of the calls from and to the appropriate protocols and routes the calls to the modules that provide the appropriate functionality – it was extended to meet the multi-protocol access requirements of OMM as well as to provide session handling in a manner desired by the versioning and security features of OMM.

Follows a layer consisting of various modules each of which implements the basic functionality via calls to the storage and inference layers (SAIL).

The SAIL interface hides the specific implementation dependant to the underlying physical storage, features and formal semantics supported. A number of SAILS can be stacked on top of each other, each of them handling some of calls and transmitting the rest to the underlying one. The access to the SAILS as well as the communication between goes through the SAIL interfaces. More about the SESAME architecture and features can be found in [Broekstra and Kampman, 2001b].



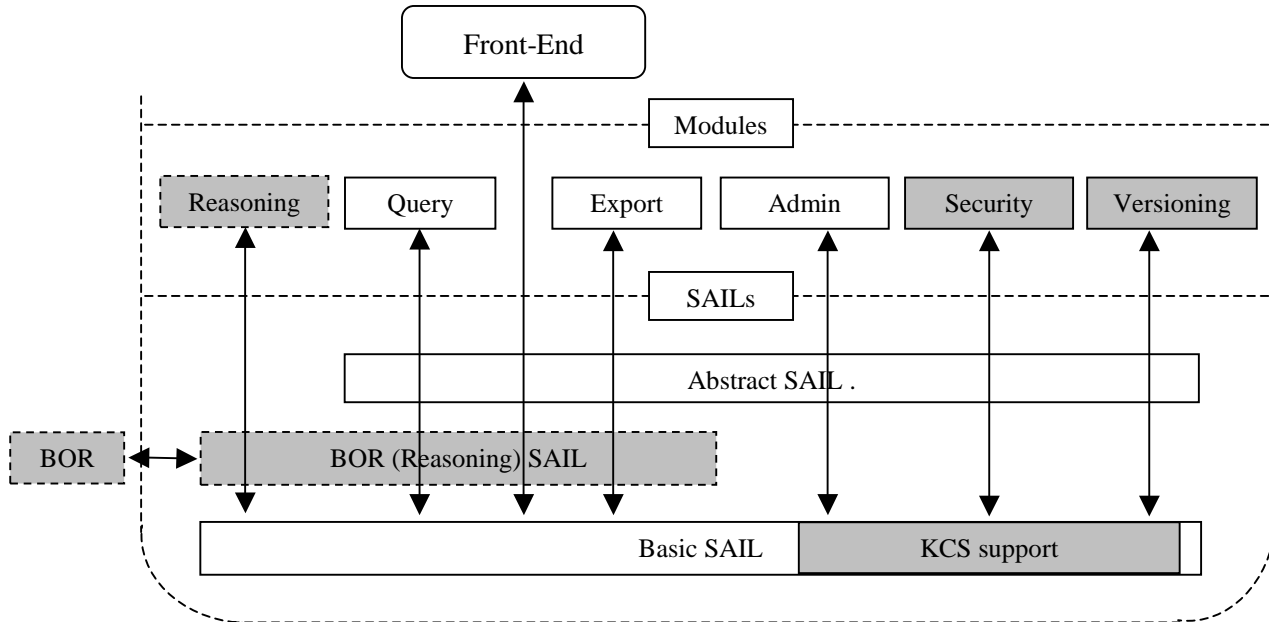
1.2.2. How OMM Fits in the Picture?

The interfaces of the OMM are designed to fit in the stacked SAIL architecture by extending it with their specific functionality. Some of the SESAME functional modules were modified so to benefit directly from the interfaces in the OMM SAIL extension. The implementation of BOR (the DAML+OIL reasoner, [Simov and Jordanov, 2002]) was integrated through an additional SAIL. Its full potential is accessible via a separate Reasoning module responsible for routing the appropriate calls to the reasoner. On the other hand the reasoner supports the SAIL interface so the rest of the modules can also interoperate with it like with the standard RDF(S) supporting SAIL. For instance, the Query module will be able to perform queries against DAML+OIL repositories without changes to its interfaces.

It is also the case that the existing basic SAIL (that implements the physical storage and the RDF(S) reasoning on top of a relational database) have been modified so to take care at the lowest level for some of the kinds of meta-information, especially those necessary for the knowledge control system (KCS). The reason for this design is that it is extremely important the information related to the access rights and "history" of the statements in the repository to be stored in the most efficient way.

This architecture allows transparent manipulation of the repository for the existing tools. Each application can either work with the repository through the existing SESAME modules or by changing the repository behavior working directly with the new reasoning, versioning, and security modules or gaining the access to the underlying low-level programming interfaces.

The scheme below represents how the lower part of the current SESAME architecture was changed. As mentioned earlier, the Request Router also undertook some modifications so to be able to (i) route the new interfaces and (ii) support more protocols. There were also important changes that take place in the router in order to enable the most convenient access control mechanism for each of the specific protocols.



The gray boxes denote the new modules or extensions developed. The arrows depict how the requests of the different modules are passing through the different SAILs. The boxes with dashed borders (Reasoning and BOR) are optional, i.e. not an obligatory part of the OMM/SESAME architecture. Those are only necessary for DAML+OIL reasoning.

Another new feature is that front-end applications (such as editors and viewers) are now able to communicate directly with the SAILs without using any of the functional modules. The requests of such application (as all the requests coming outside SESAME) are handled through the request router. It also means that such requests are subject of the standard access control and they are possible through all the supported protocols. This feature is still under development and testing.

2. Installation

The Ontology Middleware Module (OMM) is an open-source extension of SESAME. It is available together with its source code, examples, and documentation from the SESAME project at SourceForge². Most up to date installation instructions and binary files can always be found at <http://www.ontotext.com/OMMInstall.html>, we strongly recommend these instructions to be used – the remaining part of this section is only intended to give a rough idea about the procedure.

The Ontology Middleware Module can be easily installed on top of a SESAME installation working with MySQL database (SQL92SAIL). Here are the major steps during the installation:

- Get a version of `sesame.jar` which contains all the `com.ontotext.omm` classes. Alternatively, one may just add these class files to an existing `sesame.jar`. The later one usually resides in folder `<TOMCAT_HOME>/webapps/sesame/WEB-INF`.
- Extend the `web.xml` file (again in `<TOMCAT_HOME>/webapps/sesame/WEB-INF`) with registrations and mappings for two more servlets: `createVersion` and `selectBranch`. This requires the following text to be added for the first one:

```
<servlet>
  <servlet-name>createVersion</servlet-name>
  <servlet-class>com.ontotext.omm.http.CreateVersion
</servlet-class>
...
</servlet>
```

and

```
<servlet-mapping>
  <servlet-name>createVersion</servlet-name>
  <url-pattern>/servlets/createVersion/*</url-pattern>
</servlet-mapping>
```

Analogous sections for the `selectBranch` servlet are also necessary.

- Create a new repository in the `system.conf` file (the same directory), with the following `sail` elements in the `sailstack`:

```
<sail class="com.ontotext.omm.security.SecuritySail">
  <param name="jdbcDriver" value="org.gjt.mm.mysql.Driver"/>
...
</sail>
<sail class="com.ontotext.omm.versioning.VersioningTmsMySQLSail">
  <param name="jdbcDriver" value="org.gjt.mm.mysql.Driver"/>
  <param name="jdbcDriver" value="org.gjt.mm.mysql.Driver"/>
...
</sail>
```

Setting up the JDBC parameters according to your database installation (should the same as for the `SQL92SAIL`).

Having the installation finished, OMM services can be configured on a per-repository basis – within single installation there could be one repository with access control, another just with tracking and third one using both or none of this services.

Making use of OMM depends on the exact feature. For instance, for a repository which was set up to use the `versioningTmsMySQLSail` the changes will be tracked without any action on the user or application side. As another example, for repositories using the `SecuritySail`, the security policy³ will be enforced, again without any change in the APIs – every regular request to OMM/SESAME will be controlled by the security sail. For instance, in an RQL query, only those statements that fit

² <http://www.sourceforge.net>

³ Or just the default security policy if there is no such defined.

into the users permissions will be returned as a result.

3. Skills Example

For the purpose of testing and verification of OMM we extended the skills management schema from the case study presented in Deliverable 20, [Novotny and Lau, 2001]. It is a toy Skills Management knowledge base structured as follows:

- **Enterprise.rdfs** – a general organizational ontology, covering the basic types of organizations, various employment and management relations, as well, as structural relations in complex organizations. No ownership issues modeled. This model is derived from the Upper Cyc Ontology, [Cycorp 1997]. It includes classes such as `Organization` and `Position` and properties such as `hasPosition`;
- **Skills.rdfs** – a general ontology about personal skills, training courses, and competence requirements for certain positions, expressed via the required skills. Includes classes such as `Skill` and properties such as `hasSkill`;
- **Sirma_enter_kb.rdf** – description of the companies, people and positions within Sirma Group. The knowledge is simplified but correct view of a real organization;
- **Sirma_skills_hierarchy.rdfs** – non-exhaustive hierarchy of skills relevant for Sirma Group. The top skills are `BusinessSkill`, `TechnicalSkill`, and `LanguageSkill`;
- **Sirma_skills_kb.rdf** – information about the skills of specific people within Sirma Group.

Overall, the sample knowledge base has about 140 classes and 600-700 statements. The files are available at <http://www.ontotext.com/otk/2002/05/>.

In order to be able to reproduce the test cases, the files should be uploaded in this order in a clear repository with base URIs like http://www.ontotext.com/otk/2002/05/<file_name>. Both `VersioningTmsMySQLSail` and `SecuritySail` should be included in the stack. The user account used should have the appropriate rights – in order to avoid mistakes and confusion, we recommend the default administrator account (admin/admin) to be used for the upload.

4. Major Test Cases

Test cases based on the examples from the previous section are presented here for the purpose of demonstration of the key features of the Ontology Middleware Module.

The steps in each of the case descriptions presuppose that you are logged into SESAME the system, often with requirements for the user account being used. The most basic user interface to SESAME currently is a web-interface that can be started just putting a URL such as http://your_server:a_port/Sesame - the exact details depend on the SESAME and Tomcat installations. Once getting to the entry page of SESAME, you can log in as a specific user if you follow the [Log In](#) link.

4.1. Work with a Previous Version

After loading the sample into the database, you can check how the uploading transaction were tracked. The first six updates (UIDs 2 to 7) correspond to the upload of each of the files. In order to see one of them it has to be first labeled as a version (this is due to the specifics of the current GUI – there is no such requirement in principle.)

1. Select `Set Working Version` from the `Available Actions` page. In the `Update ID` combo-box, you see the valid Update Ids – this is actually the list of the states of the repository.

2. Select an `update ID` (say 2), put a version name in the `Label` field (say “ver2”), and press `create`. In case of successful labeling/creation of a version you will be sent back to the `Available Actions` page.
3. Select `set working version` again. The new version should appear in the `Repository Versions` list.
4. Follow the link for the newly created version, say “ver2”. In this moment, a dummy branch of the current repository is being created and populated with the state of the repository at the selected version. This process depending on the size of the repository can take few seconds or a minute. You are automatically getting switched to work with the newly-created branch in **read-only mode**. After successful branching, you are sent to page, where you can see text like “Available actions on **branch of 2 repos_name** [select other] are:...”. This is an indication that you are currently working with the branch of your initially selected repository, populated with its state at the selected version – any requests and queries will be interpreted with respect to this state.
5. From the `Available Actions` page, select Evaluate an RQL Query, and evaluate the “class” query – the classes introduced after this update (after the upload of the corresponding file) will not appear in the result. For instance, if you set as a working version, Update ID 2, this means that all the specific skill classes (such as `http://.../sirma_skills_hier.rdfs#Java`) should be missing.

4.2. Play with the Security

This set of demos concerns the fine-grained Security and Access Control functionality over RDF(S) repositories. Its aim is to present (via reasonable examples) the types of security restrictions covered by the current implementation. In order to obtain better understanding on the formal representation of the security policy, one should read section 3 in [Kiryakov et al, 2002].

4.2.1. Demo Security Setup

This section describes the *security setup* used in this demo by presenting the permissions given to each user used in the demo.

The **Admin** is granted full permissions (*read, add, remove, admin, history*) over the whole repository via a *repository restriction*, which is used in a do-everything-with-the-repository rule.

The **Anonymous** user is not given any permissions, which means unable to perform any of the access actions, even read.

The **testuser** is allowed to read the entire *schema* via a *Schema Restriction* used in a rule, specifying *Read* right.

The **HRManager** is allowed to *add, read, remove* all the subclasses of *Skill*. This part of the repository is restricted via a *Classes Over Schema Restriction*. *Class Restriction* is used to allow the user to *read, add, remove* all instances of Skills.

The **RandDManager** has the *read, add, remove* rights for the *TechnicalSkill* hierarchy and its instances. These permissions are set in the same manner as for the *HRManager* : via *Class Restriction* and a *Class Over Schema Restriction* combined in a security rule.

The **Employee** has the most complex security setup in the demo. Its permissions are set via three *Pattern Restrictions* formed by three *Classes Restrictions*, three *Properties Restrictions* and one *Classes Over Schema Restriction*. The security rules formed on the basis of the *Pattern Restrictions* grant *Read* right to the *Employee*.

- Pattern restriction `< iPerson, pHasPosition, _ >`, where: *iPerson* is a *Classes Restriction* restricting the instances of *Perso*. *pHasPosition* is a *Properties Restriction* restricting the sub-properties of *hasPosition*. This restriction specifies the triples that state

the position which a particular person has.

- Pattern restriction $\langle \text{scSkill}, \text{pSubClassOf}, _ \rangle$, where: *scSkill* is a *Classes Over Schema Restriction* restricting the subclasses of *Skill*; *pSubClassOf* is a *Properties Restriction* restricting the sub-properties of *subClassOf*. This restriction specifies all triples that define the Skills hierarchy.
- Pattern restriction $\langle \text{iPerson}, \text{pHasSkill}, \text{iTechnicalSkill} \rangle$, where: *iPerson* is a *Classes Restriction* restricting the instances of *Person*; *pHasSkill* is a *Properties Restriction* restricting the sub-properties of *hasSkill*; and *iTechnicalSkill* is a *Classes Restriction* restricting the instances of *TechnicalSkill*. This restriction specifies all triples that state the belonging of a certain technical skill to a particular person.

4.2.2. Repository Restriction Demo

Restricts the whole repository. Certain rights are applicable only for this restriction type (such as Admin and History).

This section will present the access to the repository when there is (or isn't) present a repository restriction for the user being used. The **admin** user has a **do-everything-with-the-repository** rule associated. This rule is based on a **repository restriction**. The **anonymous** user does not have assigned a rule based on repository restriction, neither any other rules.

1. Log in⁴ to Sesame as [**admin** | **admin**].
2. Choose **Evaluate an RQL Query**.
3. Type in the query : **select * from {X} @p {Y}**. All the triples in the repository are displayed.
4. Log out.
5. Log in as [**anonymous** |] .
6. Choose **Evaluate an RQL Query**.
7. Type in the query : **select * from {X} @p {Y}**. No triples are displayed.
8. Log out.

4.2.3. Schema Restriction Demo

Restricts all the resources and statements that constitute the schema of the repository.

The **testuser** does have a rule that allows *reading* and is restricted by a schema restriction. On the other hand, the **anonymous** is not allowed to view the schema.

1. Log in to Sesame as [**testuser** | **opensesame**].
2. Choose **Evaluate an RQL Query**.
3. Type in the query : **select * from {X} @p {Y}**. All the triples forming **the schema** are displayed (e.g. 476 results found, while the whole repository consists of 608 triples.)
4. Log out.
5. Log in as [**anonymous** |] .
6. Choose **Evaluate an RQL Query**.
7. Type in the query : **select * from {X} @p {Y}**. No triples are displayed.
8. Log out.

4.2.4. Classes Restriction Demo

Restricts all the resources (instances) of specific classes, including the statements

⁴ Each time "Log in" is suggested, please also select the database with the Skills Hierarchy Example loaded and with the Security Sail set for it as one of the stacked sails.

where those are subjects. Disjunction logic applicable in case of multiple classes. Resources that are instances of sub-classes also considered. Defined via set of classes.

This example demonstrates that the **HRManager** is able to *add/remove instances* under the *Skills* hierarchy (Instances of <http://www.ontotext.com/otk/2002/05/skills.rdfs#Skill> and its subclasses). While the **Employee** is able only to *read* these.

For this purpose a new **DemoSwingSkill** class will be added as an instance of the **Swing** skill. It will be associated with an existing employee and the level of this skill for this employee will be specified (<http://www.ontotext.com/otk/2002/05/SecurityDemo/newSwingSkill.rdf>).

A. Add

1. Log in as [**Employee** | **E**].
2. Choose *Add data from the world wide web*.
3. As a *URL of the data set* :
<http://www.ontotext.com/otk/2002/05/SecurityDemo/newSwingSkill.rdf>. Press **Add Data**. The status should display that the addition of statements is impossible since access is denied. To check this:
4. Go back and choose *Explore the Repository*.
5. Paste this URI: http://www.ontotext.com/otk/2002/05/sirma_enter_kb.rdf#DemoSwingSkill and select *Explore*. No results should be displayed.
6. Log out.
7. Log in as [**HRManager** | **HR**].
8. Choose **Add data from the world wide web**.
9. As a *URL of the data set* :
<http://www.ontotext.com/otk/2002/05/SecurityDemo/newSwingSkill.rdf>
10. Press **Add Data**.
11. The status should display the number of added statements.
12. In order to list all the triples which subject is *DemoSwinSkill* go back and select *Evaluate and RQL Query*. Paste the following query:
13. select * from {S} @p {O}
14. where S = http://www.ontotext.com/otk/2002/05/sirma_enter_kb.rdf#DemoSwingSkill
15. Three (3) results should be displayed, where one of the triples states that *DemoSwingSkill* is instances (rdf:type) of the *Swing* skill.
16. Log out.

B. Remove

1. Log in as [**Employee** | **E**].
2. Choose **Remove statements**. We will try to remove the triple that defines *DemoSwingSkill* as instance of the *Swing* skill.
3. Specify *subject, predicate, object* as follows:
subject : http://www.ontotext.com/otk/2002/05/sirma_enter_kb.rdf#DemoSwingSkill
predicate : <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
object : http://www.ontotext.com/otk/2002/05/sirma_skills_hier.rdfs#Swing
4. Proceed by pressing the *Remove statements* button. The notification should say that *no statements have been removed* since the *Employee* has no permission to remove instances under the *Skills Hierarchy*.
5. Now the *remove attempt* will be held under the *HRManager* login.
6. Log in as [**HRManager** | **HR**].
7. Choose **Remove statements**. We'll try to remove the triple that defines *DemoSwingSkill* as instance of the *Swing* skill.
8. Specify only the *subject* :
http://www.ontotext.com/otk/2002/05/sirma_enter_kb.rdf#DemoSwingSkill
9. Proceed by pressing the *Remove statements* button. The notification should say that the statement (one statement) *has been removed*.

Note: Using **RandDManager** instead of **HRManager** and **TechnicalSkill** instead of **Skill** could additionally demonstrate the scenario demonstrated in this section.

4.2.5. Classes-Over-Schema Restriction Demo

Restricts all the classes and properties that are sub (class/property) of the resource stated in the restriction's definition. Is a restriction over the schema.

This section presents the granted permission of the HRManager to edit the Skills Hierarchy via a security rule using Classes Over Schema Restriction.

A. Add

1. Log in as **[Employee|E]**.
2. Choose *Add data from the world wide web* and set the following URL : <http://www.ontotext.com/otk/2002/05/SecurityDemo/skillClass.rdfs> - it contains a new subclass of *BusinessSkill* - *BusinessEspionage*. The employee should not be able to add such a statement.
3. Press *Add Data*.
4. Go back and choose *Evaluate an RQL query*.
5. Paste the following query:

```
select * from {X} @p {Y}
where X = http://www.ontotext.com/otk/2002/05/sirma_skills_hier.rdfs#BusinessEspionage
```

No results should be found which indicates that indeed the statements have *not been added*.
6. Log out.
7. Log in as **[HRManager|HR]**.
8. Repeat steps 2-5 above, three statements will be returned after step 5.

B. Remove

1. Log in as **[Employee|E]**.
2. Choose *Remove Statements* and Set the *subject* to:
http://www.ontotext.com/otk/2002/05/sirma_skills_hier.rdfs#BusinessEspionage. This should remove all statements which subject is *BusinessEspionage*.
3. Press *Remove Statements*. The status should report that *no statements have been removed*.
4. Log out.
5. Log in as **[HRManager|HR]**.
6. Repeat steps 2 and 3. After step 3 the status should report that 3 statements have been removed, which demonstrates the *remove* permissions granted to the *HRManager* under the *Skills Hierarchy*.
7. Log out.

Note: Using **RandDManager** instead of **HRManager** and **TechnicalSkill** instead of **Skill** could additionally demonstrate the scenario demonstrated in this section.

4.2.6. Properties Restriction Demo

Restricts all the statements with specific properties as predicates. Disjunction logic applicable in case of multiple properties specified. The sub-properties are also considered. Defined via set of properties.

The *Properties Restrictions* are demonstrated in the following section as part of the *Pattern Restrictions* demo, since each of the presented *Pattern Restrictions* has a *Properties Restriction* as a constituent.

4.2.7. Pattern Restriction Demo

Restricts all the statements that conform to patterns defined via restrictions of type Classes or Instances over the subject and/or object and restriction of type Properties over the predicate;

The **Employee** user will be used to demonstrate its permissions defined via three Pattern Restrictions.

< iPerson, pHasPosition, >

1. Log in as [**Employee|E**].
2. Choose *Explore the Repository*.
3. Set the following URI: <http://www.ontotext.com/otk/2002/05/enterprise.rdfs#hasPosition> and *Press Explore*. Only the 5 statements that comply with this restriction should be displayed, the others are filtered.
4. Log out.

< scSkill, pSubClassOf, >

1. Log in as [**Employee|E**].
2. Choose *Explore the Repository*
3. Set the following URI: <http://www.w3.org/2000/01/rdf-schema#subClassOf> and *Press Explore*. Only the triples from the Skills hierarchy definition are displayed having *subClassOf* as predicate, the others are filtered. Compare via the same query using the admin user (follows).
4. Log out.
5. Log in as [**admin|admin**].
6. Repeat steps 2 and 3, now it displays the triples that contain as a constituent *subClassOf*, either as subject, predicate or object.
7. Log out.

< iPerson, pHasSkill, iTechanicalSkill >

1. Log in as [**Employee|E**].
2. Choose *Explore the Repository*
3. Set the following URI: <http://www.ontotext.com/otk/2002/05/skills.rdfs#hasSkill>. Only the triples with *TechnicalSkill* as an object are displayed, the others are filtered. Compare via the same query using the admin user (follows).
4. Log out.
5. Log in as [**admin|admin**].
6. Repeat steps 2 and 3, now it displays all the triples having *hasSkill* as predicate. The result should contain at least one more triple than the previous result, because there is at least one *aPerson,hasSkill, aSkill* triple, in which *aSkill* is not a *TechnicalSkill*, but is a *LanguageSkill*, for example.
7. Log out.

5. Implementation Details

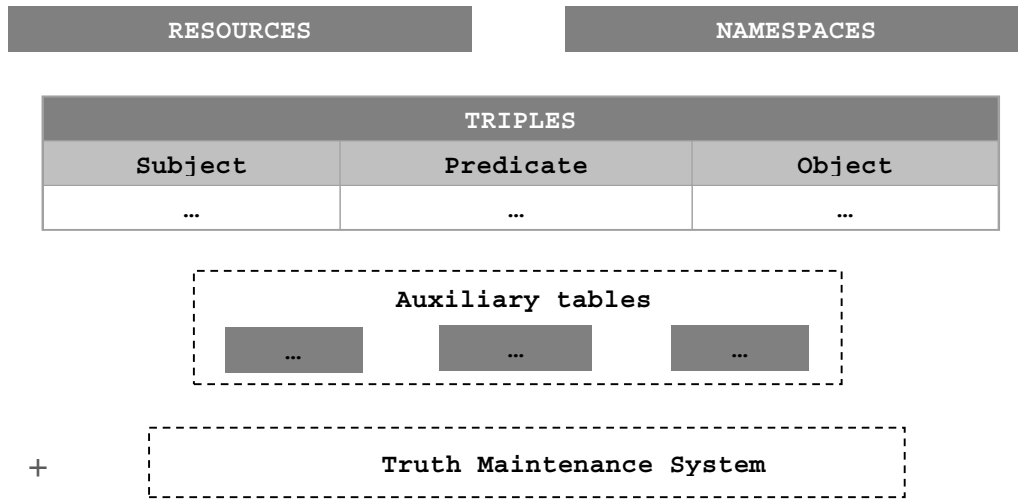
More important implementation details and alternations in the design (with respect to Deliverable 38, [Kiryakov et al, 2002]) are presented here for the major sub-systems of the ontology middleware – tracking changes and access control. For good understanding of the following subsections understanding respectively of sections 2 and 3 of Deliverable 38 is required. The conceptual model for track of changes is also presented in [Kiryakov and Ognyanov, 2002a] and [Kiryakov and Ognyanov, 2002b].

5.1. Tracking Changes Implementation

The only significant change in the implementation plans was related to the necessary extension of the DB schema of SESAME for the purposes of the ontology middleware. However, first it is important to acknowledge that after the requirements for a truth maintenance system (TMS) in

SESAME, outlined during the analysis and design for the OMM, such TMS was developed by administrator b.v. on a fairly short time scale, taking into account its complexity.

The database schema used from the SQL92SAIL (and respectively the MySQL implementation)



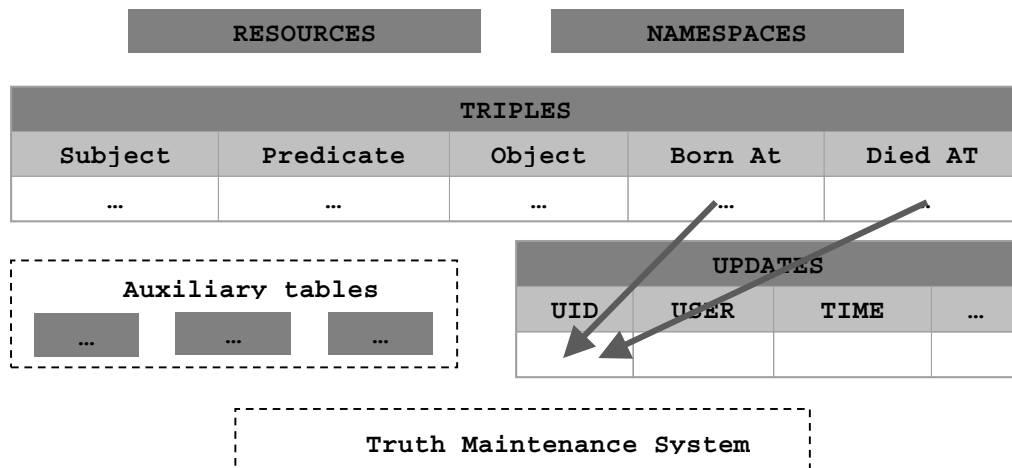
can be depicted as follows.

The original plan and the first implementation of the ontology middleware extensions of SESAME was build on the following database schema:

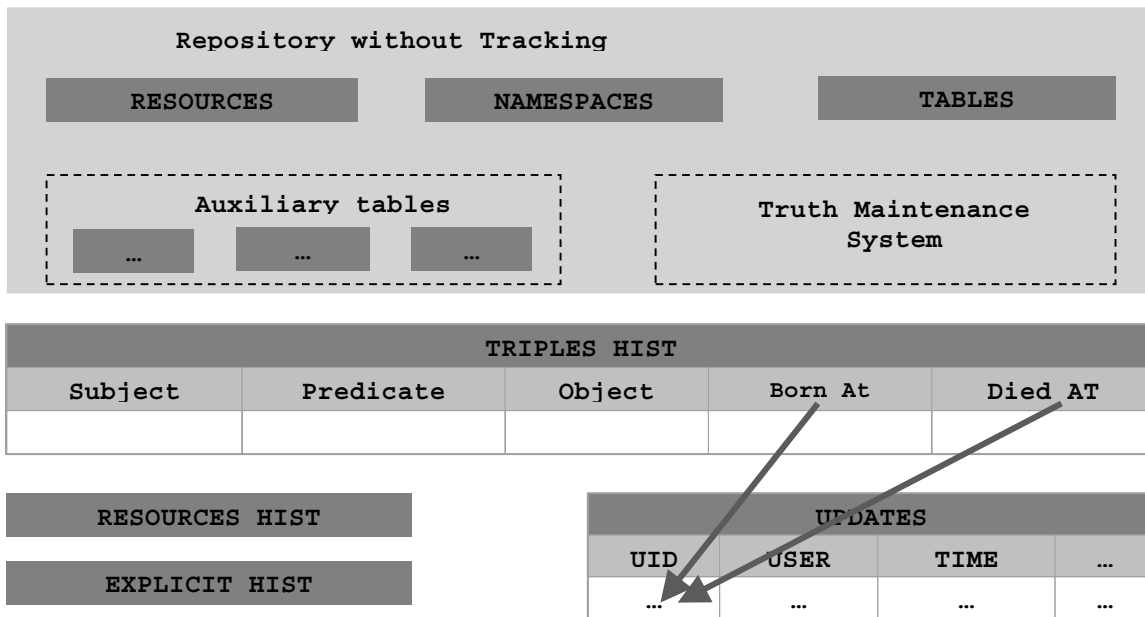
The above schema was based on the idea to extend the existing TRIPLES table with two more columns with references to the updates which caused the adding and deletion of the corresponding statement. Under this schema, the triples that are “valid” in certain state of the repository can be queried via the appropriate restrictions on the **BornAt** and **DiedAt** columns. This idea appeared to be problematic dew to a number of reasons:

- There was a need for re-implementation of the basic SAIL in order to add methods that can work with previous versions of the repository without making use of the auxiliary tables (that always correspond to the current state.)
- This schema fails to account for the transitions between implicit and explicit states of the statement during its lifetime. This is a major problem as far as many methods in the SAIL interface the query module count on this distinction.
- Finally, the performance when working with the current version was affected dew to need of additional conditions on each query including the TRIPLES table.

After analysis of different design alternatives and their impact on the performance, we



implemented the following schema:



The core idea is that the original tables remain unchanged and additional (in a sense parallel) tables are used to preserve the tracking information. When a user needs to be able to work with a previous version, the appropriate state is branched into a separate auxiliary repository which is read-only and has no support for tracking changes. This schema has the following advantages and disadvantages:

- some information is duplicated (-);
- to start work with old version a dummy branch is created, which takes both considerable time and space (-);
- working with the current version is as fast as without tracking, the implementation is much simpler (+);
- working with old version is as fast as working with the current one – the same optimizations work (+);

5.2. Access Control Implementation

There was a single major modification necessary for the implementation of the security subsystem. After more careful analysis of different use cases and sample security policies, we uncovered that one additional restriction type is necessary, namely **classes-over-schema** – it is specified by a set of classes and determines the set of all resources which represent their subclasses⁵. In contrast, the **classes** restriction type determines set of resources which are instances of the specified classes.

An example for usage of both the Classes and Classes-over-schema restriction types are available in sub-section 4.2.1. above.

⁵ As well as the statements which has such resources as subject – no change with respect to the Classes restriction type in this.

6. Performance comments

During the current phase of the project we concentrated on implementation of the basic functionality of the OMM leaving more serious optimization and performance evaluation plans for the next phase – task 11.4, Integration. Here follow some preliminary comments on various aspects of the OMM. The Security sub-system is still under optimization and performance evaluation.

Among the most important issues about the performance of the OMM is time and space overhead caused by the mechanisms for tracking of the changes. Few measurements shared below:

- The time overhead when adding statements (uploading ontologies) within a repository with tracking changes support (as compared to such without) is below 5%. This means that the price for supporting all this additional information related to the tracking is fairly low.
- The space overhead (in the database) caused by the additional tracking information depends on the number of updates performed over each of the statements in the repository. In a hypothetical situation, when 100 resources and 500 statements are initially added, than half of them are deleted and another 50 resources and 250 statements were uploaded, the overhead (in terms of volume in the database) is some additional 150%. It can be explained as follows: the historical data for each statement or resource that ever appeared in the repository is roughly as much as the “basic”/“regular” information preserved for it. This for instance, means that immediately after upload in an empty repository with tracking support, the space overhead will be something like additional 100%.
- The time for branching of (certain state of) a repository is comparable with the time for uploading the explicit statements in a new repository.
- As it was expected, there is no impact on the non-modifying transactions with the repository – all queries on the current and previous versions work without any delay as compared to repository without tracking support.

7. Conclusion

The core functionality of the Ontology Middleware Module were implemented following the analysis and design reported in [Kiryakov et al, 2002]. During the next phase of the project, the goal will be optimize and measure the performance of the system and to work on the integration with the rest of the tools developed under the On-To-Knowledge.

8. References

- [Brickley and Guha, 2000] W3C; Dan Brickley, R.V. Guha, eds.
Resource Description Framework (RDF) Schemas.
<http://www.w3.org/TR/2000/CR-rdf-schema-20000327/>
- [Broekstra and Kampman, 2001a] Jeen Broekstra, Arjohn Kampman
Query Language Definition.
 Deliverable 10, On-To-Knowledge project, May 2001.
<http://www.ontoknowledge.org/download/del10.pdf>
- [Broekstra and Kampman, 2001b] Jeen Broekstra, Arjohn Kampman
Sesame: A generic Architecture for Storing and Querying RDF and RDF Schema.
 Deliverable 9, On-To-Knowledge project, October 2001.
<http://www.ontoknowledge.org/download/del10.pdf>
- [Cycorp 1997] Cycorp. Cyc Public Ontology, 1997.
<http://www.cyc.com/cyc-2-1/>
- [Ding et al, 2001] Ying Ding, Dieter Fensel, Michel Klein, Borys Omelayenko

Ontology management: survey, requirements and directions.

Deliverable 4, On-To-Knowledge project, June 2001.

<http://www.ontoknowledge.org/download/del4.pdf>

[Kiryakov and Ognyanov, 2002a] Atanas Kiryakov and Damyan Ognyanov.

Tracking Changes in RDF(S) Repositories.

In the Proceedings of Workshop on Knowledge Transformation for the Semantic Web, at the 15th European Conference on Artificial Intelligence, pages 27-25. Lyon, France, July 23, 2002.

[Kiryakov and Ognyanov, 2002a] Atanas Kiryakov and Damyan Ognyanov.

Tracking Changes in RDF(S) Repositories.

In the Proceedings of 13th International Conference on Knowledge Engineering and Knowledge Management EKAW02, Sigüenza, Spain, 1-4 October 2002. To Appear.

[Kiryakov et al, 2002] Kiryakov A., Simov K. Iv., Ognyanov D. *Ontology Middleware: Analysis and Design.* Deliverable 38, On-To-Knowledge project, March 2002.

<http://www.ontoknowledge.org/download/del38.pdf>

[Lassila and Swick, 1999] W3C; Ora Lassila, Ralph R. Swick, eds.

Resource Description Framework (RDF) Model and Syntax Specification

<http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/>

[Lenat, 1998] Lenat, Doug. *The dimension of context-space.* Working paper of the Cycorp Corp. Austin, Texas (1998).

[Simov and Jordanov, 2002] Simov K. Iv., Jordanov S. *BOR: a Pragmatic DAML+OIL*

Reasoner. Deliverable 40, On-To-Knowledge project, June 2002.

<http://www.ontoknowledge.org/download/del40.pdf>

[Novotny and Lau, 2001] Novotny B., Lau T. *Organizational Memory Description of Case Study Prototypes.* Deliverable 20, On-To-Knowledge project, June 2002.

<http://www.ontoknowledge.org/download/del20.pdf>

Appendix A: Limitations of the Implementation

This appendix provides discussion on issues raised from the reviewers at the 3rd project review.

A.1. Tracking of changes vs. Versioning and Temporality

The problem of handling and managing the temporal aspect of the information is widely studied in the Philosophy as well as in the fields of the Mathematical logic and Database management. It is often considered also as one of the toughest issues related to context modeling (see [Lenat, 1998]) and process ontologies. In order to avoid confusion about the sort of temporality OMM supports, we provide here some clarifications:

- **What OMM is about?** – OMM has two major goals (i) to support collaborative Knowledge Engineering and (ii) to provide platform for Semantic Web and Knowledge Management enterprise applications. The analysis, design, and implementation were led by these goals. Temporal reasoning and temporal databases were never subject of discussion, even more they were consciously avoided because their complexity can put into serious risk other more relevant tasks;
- **Versioning vs. Update Tracking** – OMM provides tracking of changes suitable for the following purposes (i) to be able to serve the knowledge engineers in the same way the “source control systems” serve the software engineers and (ii) to allow for change investigation and logging. Tasks such as automatic detection of the differences and degree of compatibility between two version of one and the same ontology were never an issue;
- **Changes in the Real-world vs. Changes in the Knowledge about it** – OMM is only handling at a basic level the changes in the formally represented knowledge disregarding are those somehow related to changes in the real world or not.

A.2. Meta Knowledge about Ontologies

The way in which OMM support meta-information about ontologies is not presented well enough. This feature was mostly designed and available for OIL and our days DAML+OIL ontologies. It is the case there that the ontology is described itself as resource⁶:

```
<rdfs:Class rdf:ID="Ontology">
  <rdfs:label>Ontology</rdfs:label>
  <rdfs:comment>
    An Ontology is a document that describes
    a vocabulary of terms for communication between
    (human and) automated agents.
  </rdfs:comment>
</rdfs:Class>
```

In this context, any information, which can be assigned to concepts and instances, can also be assigned to ontologies, which are nothing more but resources from the above-described class. Any sort of features can be assigned to the ontologies and those can be retrieved afterwards with the regular RDF(S) and DAML+OIL means. There are two technical issues to be resolved so to allow meta-information for ontologies to be efficiently used within the above framework:

- To enable cross-repository search on meta-information for ontologies so to avoid the need of iteration through all the repositories in case of meta-information based lookup for ontologies. Although a minor engineering issue, it was unpleasant to admit that it was overlooked during the design phase. Currently, it was scheduled for addition to the OMM

⁶ Taken from <http://www.daml.org/2001/03/daml+oil>

API and implementation.

- To be able to distinguish between information/knowledge and meta-information. This problem is already resolved for any sort of meta-information including such about ontologies.