ECHO DEPOSITORY: OVERVIEW OF SEMANTIC ARCHIVING RESEARCH

CHAD CURTIS

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN
GRADUATE SCHOOL OF LIBRARY AND INFORMATION SCIENCE

Date: December 2007.

Contents

Part 1. BECHAMEL, Digital Repositories, and RDF Databases 4
1. Introduction: Bechamel and ECHO DEPository 4
2. Relevant Projects 5
2.1. Fedora Repository 5
2.2. Kowari Metastore and Mulgara 7
2.3. RDF Database connectors/abstraction layers 8
2.3.1. Trippi 8
2.3.2. Tupelo 8
2.4. SWI-Prolog 9
2.4.1. JPL 10

Part 2. BECHAMEL Implementation 10
3. General Goal 10
3.1. Specific Goals 10
4. Testing Environment 11
4.1. Overview of of OS, Users, and Directories 11
4.2. Overview of Applications and Languages 12
5. Instructions to Replicate Environment 13
5.1. Installation of J2SDk jdk1.5.0_12 13
5.2. Installation of Kowari 1.0.5 13
5.3. Installation of SWI-Prolog 5.6.37 13
5.4. Installation of KowariClient.java 14
5.5. Installation of PostgresSQL 14
5.6. Installation of Fedora Commons 15
5.7. Ingest demos 16
5.8. Editing of .bash_profile and .bashrc 18
6. Instructions to Maintain Environment 19
6.1. Kowari 19
6.2. PostgreSQL 20
6.3. Tomcat 20
6.4. Kowari Client.pl 20
6.5. Fedora Commons 20
7. Technical Issues and Solutions 21
8. Testing/Findings 24
8.1. Non-Root Environment 24
8.2. Solitary Kowari 24
8.3. Embedded Kowari 26
8.4. Remote Kowari and Fedora/Trippi 27
8.5. JPL memory handling 30
9. Glossary 31

9.1. BECHAMEL 31
9.2. Fedora Commons 31
9.3. Kowari Metastore 32
9.4. JRDF 32
9.5. N3 (Notation 3) 32
9.6. N-Triples 32
9.7. RDF 32
9.8. SWI-Prolog 33
9.9. Trippi 33
9.10. Tupelo 33
References 33

Part 1. BECHAMEL, Digital Repositories, and RDF Databases

1. Introduction: Bechamel and ECHO DEPository

The ECHO DEPository at the Graduate School of Library and Information Science is a test environment for proof of concept of a BECHAMEL application that is able to access and modify the RDF metadata of a Fedora Commons digital object in order to form inferences with ontologies. The justification for the need of a formal BECHAMEL application has been explored in Renear et al. (2002), Sperberg-McQueen et al. (2002), Renear et al. (2003), Dubin et al. (2003), Dubin (2003), and Dubin & Birnbaum (2004). One should refer to these documents, but a more succinct description of current goals of BECHAMEL was recently presented by David Dubin:

> The BECHAMEL system is a knowledge representation and inference environment, originally created for automated interpretation of conventional markup languages. Written in Prolog, the system provides predicates for processing the syntactic structures that emerge from an SGML/XML parser, defining object classes, instantiating objects, assigning values to properties, and establishing relationships between or among object instances. BECHAMEL uses Prolog's built-in capabilities to derive inferences from these facts. BECHAMEL emerged from the joint work of researchers at the University of Illinois, the University of Bergen, and the Worldwide Web Consortium to create a knowledge-based workbench on which precise theories of the semantics of document markup could be expressed and tested. Recent improvements to BECHAMEL include a change from its native Prolog-based knowledge structures to an RDF basis. BECHAMEL can process a subset of the OWL ontology language, compute inferences over a collection of RDF triples, and express the new facts that it discovers using RDF. Ongoing development work includes interfaces for interrogating and updating a remote RDF database and processing RDF descriptions that are retrieved over the web.
>
> We propose a deductive system for reasoning about digital resources, their properties, and the machine-readable descriptions that imperfectly express those properties. The idea is not to automatically discover missing semantic information or correct malformed records, but to call human attention to descriptions that are anomalous, incomplete, or suspicious. Such a system will reason over a large knowledge base of simple assertions about digital resources. Alerts will be triggered by such anomalies as:
> 1. A property value assertion, where the identity of the property taking the value cannot be deduced.

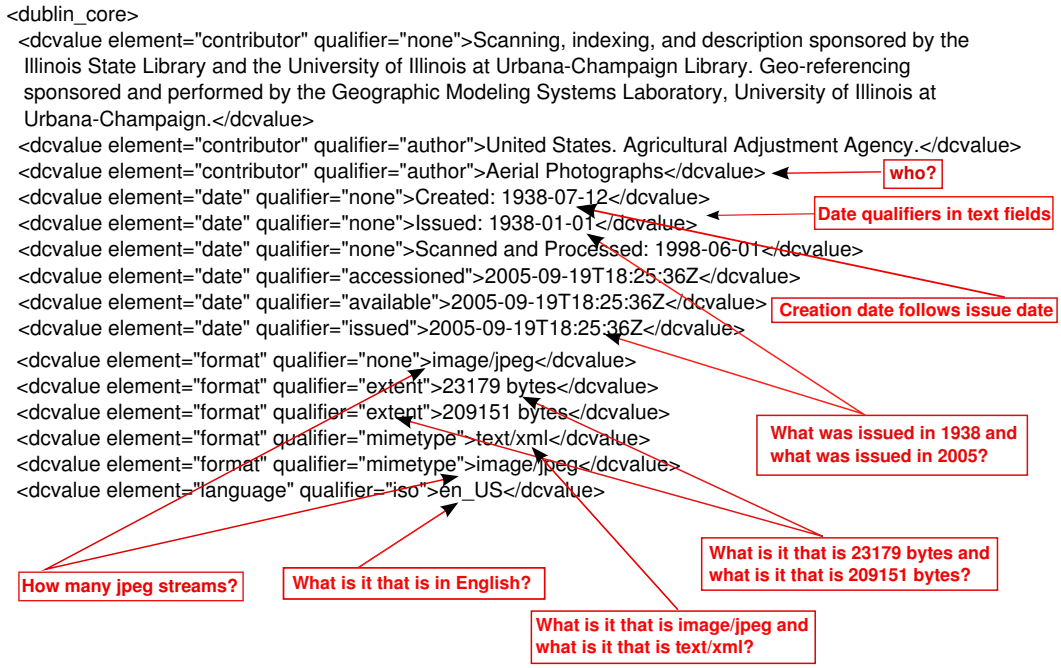
> 2. A property value assertion, where the identity of the resource exhibiting the property cannot be deduced.
> 3. Deduction that a stream or file exists without a pointer to locate it.
> 4. Ascription of a property to an instance of an inappropriate class (e.g., MIME classification to a bit sequence, rather than to the resource expressed by the bit sequence).
> 5. Apparently contradictory assertions (e.g., a resource that is both a word processor document and an executable application).
> Dubin (2007)

Due to inadequacies in proposed metadata preservation models and unreliable human implementation of metadata standards a formal framework is needed. Dubin states three problems with standard metadata encoding technologies:

> 1. Preservation metadata formats typically overload a simple syntax with multiple competing semantic interpretations. Typical examples include XML applications where a small number of syntactic relationships (e.g., the parent/child relationship between elements) represents any number of semantic relationships (whole/part, property name/value, etc.) that are context dependent. Often a precise interpretation (in formal rather than natural language) of these semantics can be found only in the execution of application software that consumes the file (and, presumably, in the mind of the programmer who wrote the application).
>
> 2. Metadata descriptions of digital resources typically confound different levels of abstraction, making it difficult to verify precisely what properties and relationships are being asserted. For example, a single description of a digital image may include assertions about the image itself (e.g., its height and width in centimeters), the image content (e.g., that it depicts a specific person), and the file that encodes the image (e.g., its size in bytes).
>
> 3. The information in resource descriptions may only be incompletely available for machine processing and verification. Crucial contextual data may exist only as natural language annotations or as unstructured information in the content of metadata fields.
> Dubin (2007)

At the 2007 ALISE Annual Conference Dubin presented a poster session, which contains Figure 1, a visual summary of issues at play:

What's Wrong with this Metadata?



Provenance

1. A paper description accompanied the original photograph, which had been taken in 1938.
2. In 1998 the photograph was scanned for inclusion in an image database. A metadata record for the photograph was entered in a relational database. The fields were derived from the FGDC Geospatial Metadata Standard.
3. In May 2005 an OAI 2.0 metadata record was derived via a mapping from the database fields into DC.
4. Months later the OAI record was transformed via XSLT for ingestion into a DSpace installation.
5. When the record was exported from DSpace, additonal DC statements were automatically added.

Preservation Risks

1. Records like this one result from many transactions, separated in time.
2. Many people contribute, and they're not all able to consult with each other.
3. The distinctions that get muddled require inferences that human minds make without conscious effort.
4. Natural language annotations may resolve the ambiguities.
5. But without machine-readable expressions for validation and constraint enforcement, each transformation compunds the errors.

FIGURE 1. A visual representation of what can go wrong with metadata.

2. Relevant Projects

2.1. **Fedora Repository.** Currently Fedora Commons is positioned as a non-profit organization contributing to a new knowledge paradigm:

> Fedora Commons will embrace the dual focus of enabling the creation of innovative, collaborative information spaces and attending to the longevity and integrity of information that results from collaboration. In doing so, it will work to ensure that change is both revolutionary and sustainable. Fedora Commons builds on the foundation of the highly successful and internationally recognized Fedora Project. Payette (2007)

Fedora Commons expanded from the initial digital repository software: Flexible Extensible Digital Object Repository Architecture (FEDORA). Fedora started development at Cornell University in 1997 and although development has spread out, project administration is based out of Cornell University and the University of Virginia. The Fedora White Paper, Fedora Development Team (2005), lists many features that make it ideal for digital object management and preservation: a powerful digital object model; extensible metadata management; ability to express inter-object relationships; and OAI-PMH conformance.

Fedora is written in Java and in the past has had three primary web service APIs: Management, Access, and Search Fedora Commons System Documentation. The management API is for administration of the repository through creation and manipulation of digital objects through a SOAP-enabled web service. The Access API is the primary interface to accessing digital objects. The Search API is the interface for searching and browsing the repository.

Fedora 2.0 added the Resource Index Search API. The following is a description of Resource Index Search API from Fedora Commons documentation:

> The Resource Index is an RDF-based index of the Fedora repository that includes the following for each digital object:
> 1. object-to-object relationships (from the RELS-EXT datastream of a digital object)
> 2. custom user-defined properties (from the RELS-EXT datastream of a digital object)
> 3. object properties (derived from the FOXML digital object) 4. metadata about datastreams and disseminations (derived from the FOXML digital object)
> 5. Dublin Core metadata (obtained from the default DC datastream of a digital object)
> Fedora Commons System Documentation

The Resource Index Search API is the most relevant API to the BECHAMEL project, as it provides access to the expression of inter-object relationships, which is made possible through Kowari Metastore.

2.2. **Kowari Metastore and Mulgara.** Within the RDF database community there are divisions between developers who think a scalable RDF Datastore can be a modeled as a relational database and those who think specialized databases must be developed. David Wood and Paul Singleton from the Kowari and Mulgara projects believe the latter:

> Kowari is an Open Source, massively scalable, transaction-safe, purpose-built database for the storage and retrieval of metadata.
>
> Much like a relational database, one stores information in Kowari and retrieves it via a query language. Unlike a relational database, Kowari is optimized for the storage and retrieval of many short statements (in the form of subject-predicate-object, like "Kowari is fun" or "Kowari imports RDF"). Kowari is not based on a relational database due to the large numbers of table joins encountered by relational systems when dealing with metadata. Instead, Kowari is a completely new database optimized for metadata management. Kowari MetaStore

Kowari Metastore is the result of opening source from a commercial product created by Tucana, Tucana Knowledge Server. When Tucana was purchased by Northrop Grumman disagreements on the future of an open source version of TKS made developers leave the Kowari branding and produce the fork, Mulgara Wood (2007). It has been confirmed through correspondence to Chris Wilper that Mulgara will be used in Fedora Commons 3.0 in place of Kowari. Wilper (2007)

2.3. **RDF Database connectors/abstraction layers.**

2.3.1. *Trippi.* The programmatic connector of RDF metadata from the Fedora digital object to the modified triplestore is Trippi: "a Java library providing a consistent, thread-safe access point for updating and querying a triplestore." Trippi Connectors exist for Sesame, Kowari, Oracle Spatial, and MPTStore, but one can create a connector to other triplestores with the API. The administrator and lead developer of the project is Chris Wilper, who also plays a large role in the development of Fedora Commons.

2.3.2. *Tupelo.* NCSA's Tupelo is similar to Trippi, but aims to be more a semantic framework than just a datastore connector:

> Tupelo is a Semantic Content Repository framework. It provides means of storing, retrieving, annotating, and accessing information using Semantic Web technologies such as RDF (`http://www.w3.org/RDF/`), backed with standard storage technologies such as filesystems and databases, as well as RDF stores such as Sesame (`http://www.openrdf.org/`) and Mulgara (`http://www.mulgara.org/`).

RDF is a generic metadata framework capable of describing digital objects, real-world entities, and abstract concepts at multiple levels of specificity and granularity. Because of its use of global ID's (URI's) and named-link architecture for expressing both relationships and attributes, it provides a very simple means of assembling composite descriptions from independently-generated parts, a common issue in distributed data management.

However RDF API's and technologies have tended to be monolithic, closely tying API's and query languages to specific storage architectures. Tupelo solves this problem by providing a common abstract subsuming RDF descriptions, storage, retrieval, and basic stream operations, sort of like a JDBC for the semantic web (similar to Trippi (`http://trippi.sourceforge.net/`)). In addition to providing means of interacting with metadata, Tupelo also provides a uniform means of storing, annotating, and accessing streams in a variety of heterogeneous storage architectures.

For example, Tupelo can act as a WebDAV client, extract RDF metadata from WebDAV resources, and copy them into a Sesame repository with a small set of generic operations. Tupelo can also aggregate multiple heterogeneous stores, so that a filesystem could hold stream information, but stream annotations could be stored in Mulgara. Tupelo

2.4. **SWI-Prolog.** SWI-Prolog is currently one of the more popular open source implementations of Prolog. The official website explains the origin of SWI-Prolog:

> Better interaction between XPCE and Prolog required a powerful bi-directional interface between Prolog and C which was not provided by any Prolog system back then. Some rainy sundays triggered the birth of SWI-Prolog. At some stage this tiny system became big enough to (barely) run the Shelley Knowledge Engineering Workbench under development. In some aspects it was much better than Quintus thanks to its more flexible C interface, much faster compiler and make facility for quickly re-loading of modified files.
>
> Probably the best move was to promise a bottle of cognac for anyone finding ten bugs. There is no better way to make a system more popular, even if it has a few bugs! SWI-Prolog was born and would remain the central piece of the home-brew Software Infrastructure at HCS/SWI.
>
> In use with a growing number of projects, HCS/SWI decided to distribute SWI-Prolog at no charge using the emerging file-exchange through direct ftp-connections. It was distributed free for personal

and academic usage as well as in combination with the non-free PCE environment for academic and commercial usage.

Over the years, the (X)PCE and SWI-Prolog development was guided by the needs of our own applications as well as requirements from external users. PCE became XPCE after porting to X11 and turning it into a full-blown object-extension to Prolog. At a later stage MS-Windows was added to the supported platforms, providing a cross-platform GUI development toolkit. Growing applications made SWI-Prolog loose most of its limitations. Garbage collection and last-call optimisation was added to deal with memory-hungry applications. Atom-garbage collection and exception-handling was added to facilitate 24x7 running servers. Recent development include network (TCP/IP, CGI, HTTP), HTML/XML/SGML and RDF facilities, UNICODE support as well as constraint handling (CHR, clp(R) together with Tom Schrijvers from the University of Leuven). SWI-Prolog

2.4.1. *JPL.* The Java-Prolog Language package is now included in standard SWI-Prolog packages and binaries. JPL enables "Prolog applications to exploit any Java classes, instances, methods etc. (without requiring any wrappers, metadata etc. to be set up first)" and "Java applications to manipulate any Standard Prolog libraries, predicates, etc." JPL. More specifically:

> JPL is a set of Java classes and C functions providing an interface between Java and Prolog. JPL uses the Java Native Interface (JNI) to connect to a Prolog engine through the Prolog Foreign Language Interface (FLI), which is more or less in the process of being standardized in various implementations of Prolog. JPL is not a pure Java implementation of Prolog; it makes extensive use of native implementations of Prolog on supported platforms. The current version of JPL only works with SWI-Prolog. JPL

Part 2. BECHAMEL Implementation

3. General Goal

Maintain a Linux environment, including documentation, with the goal to support the research in semantic archiving conducted by David Dubin. The installation environment needs to be reproducible, by a non-root user, with the primary aide of the provided documentation.

3.1. **Specific Goals.** (taken from Janet Eke, "semantic_archive_notes_may2007.rtf')

B.1.1. Get Prolog to work with Kowari: build Prolog app using JPL that allows other Prolog apps to modify and inspect contents of a Kowari repos using ITQL queries

B.1.2. BECH to RDF: enhancing BECH to work with RDF and thus with Kowari

B.1.3. Prolog to Tupelo: develop Prolog interface to Tupelo APIs with aim to provide generalized access to RDF databases, including but not limited to Kowari

B.1.4.1 BECH to Tupelo: (here BECH takes advantage of above); develop capacity to store BECHAMEL-generated knowledge persistently in Kowari and retrieve for BECHAMEL using Prolog

B.1.4.2.BECH to Tupelo: implement Prolog/Tupelo interface (B.1.3.) for BECHAMEL

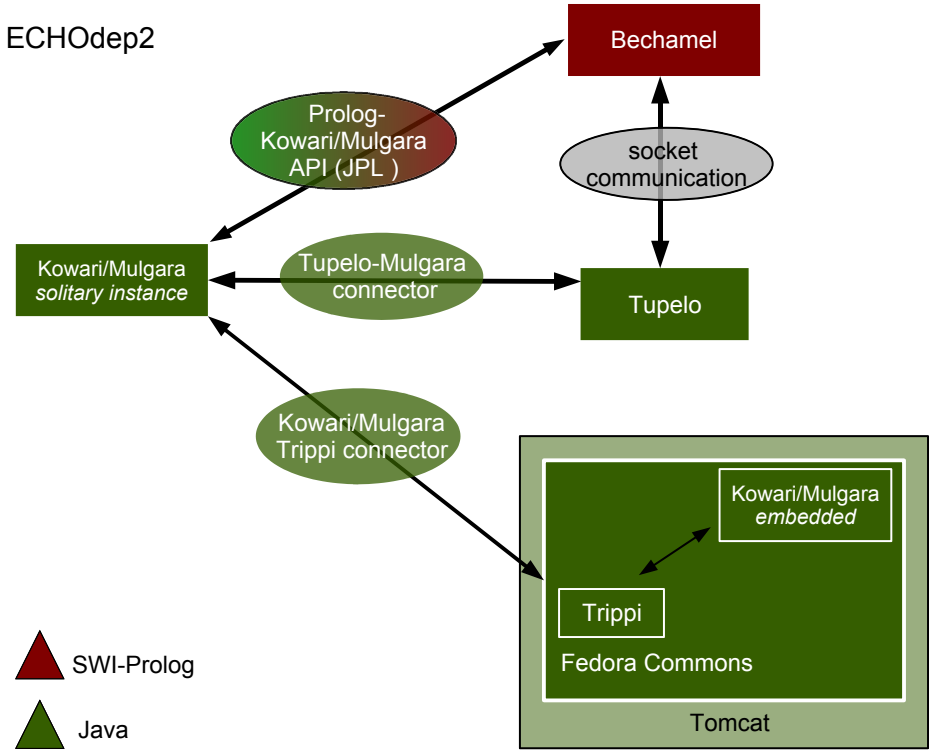


FIGURE 2. The system as implemented on the GSLIS ECHO DEPository

4. TESTING ENVIRONMENT

This is a compilation of separate text files, which contain more details, but are less user-friendly. The directory containing the ECHO-DEP text files, including

this document, can be found on the 'echodep2' server: /content/echodep/doc. Initial testing was done on echodep1, see main_doc_echodep1.pdf. Many of the latter text files refer to echodep1 directories, so this should be used as the primary source for echodep2.

4.1. **Overview of of OS, Users, and Directories.** Echodep2 (`http://echodep2.lis.uiuc.edu:8086`) is running Debian Etch 4.1.1-21. All software is installed under user, 'ccurtis3' and group 'echodep'. For the most part, the only assistance of a system administrator is needed for the creation of a user for PostgresSQL. GLSIS system administrator Brynnen Owen was contacted to create the user after supplying user name, home directory location, and password.

Directories created in /content/echodep/, not including folders created during application installation:
/doc: Documentation
/source: Compressed and uncompressed source code.
/users: Created to contain the home directory for user 'postgres'
/chad_files: misc. working scripts and notes

4.2. **Overview of Applications and Languages.** Complete list of applications installed and absolute paths for binaries of interest, as user 'ccurtis3,' group 'echodep,' in the directory, /content/echodep/. In cases where multiple versions are used, the paths are stacked:

```
Apache Ant = /content/echodep/apache-ant-1.7.0/bin/ant

Maven = /content/echodep/maven-2.0.7/bin/mvn

Apache Web Server = /content/echodep/httpd_2.2.6/bin/httpd

SWI-Prolog = /content/echodep/prolog5.6.37/lib/pl-5.6.37/
             bin/x86_64-linux/pl

Java Dev Kit = /content/echodep/jdk1.5.0_12/bin/java
               /content/echodep/j2sdk1.4.2_15/bin

Kowari = /content/echodep/kowari-1.0.5/dist/kowari-1.0.5.jar
         /content/echodep/kowari-1.0.5/dist/kowari-1.0.5.jar
         /content/echodep/kowari-1.1.0-pre2/dist/kowari-1.1.0.jar

Mulgara = /content/echodep/mulgara_1.1.0/dist/mulgara-1.1.0.jar

PostgresSQL = /content/echodep/pgsql/bin/pgsql
```

```
Fedora Commons = /content/echodep/fedora-2.2.1/
               /content/echodep/apache-tomcat-5.5.23/webapps/fedora

Apache Tomcat = /content/echodep/apache-tomcat-5.5.23/bin/startup.sh
        /content/echodep/jakarta-tomcat-5.0.28/bin/startup.sh

Tupelo = /content/echodep/t2/
```

5. Instructions to Replicate Environment

To create an environment limited to non-root users the following was used: user 'ccurtis3' in group 'echodep' ccurtis3 has read, write, and execute permissions to /content/echodep/. Only applications and languages directly involved in installing software and managing applications were installed. Otherwise existing binaries (make, etc.) installed by root were used.

5.1. **Installation of J2SDk jdk1.5.0_12.** This is the simplest install due to it being a binary file. Agree to the license agreement and you're done.

```
wget -O jdk-1_5_0_12-linux-amd64.bin http://192.18.108.239/ECom/
EComTicketServlet/BEGIND4C23EDCA5BA6553AB880D7B9B7038EA/
-2147483648/2248771647/1/832394/832358/2248771647/2ts+/
westCoastFSEND/jdk-1.5.0_12-oth-JPR/jdk-1.5.0_12-oth-JPR:15/
jdk-1_5_0_12-linux-amd64.bin

gserve102:source 507 \$ chmod +x jdk-1_5_0_12-linux-amd64.bin
gserve102:source 508 \$ ./jdk-1_5_0_12-linux-amd64.bin


j2sdk1.4.2_15:
wget -O j2sdk-1_4_2_15-linux-i586.bin http://192.18.108.146/ECom/
EComTicketServlet/BEGIN628C6744AB66DB130618BE0CFA2CED43/
-2147483648/2329856739/1/837230/837110/2329856739/2ts+/
westCoastFSEND/j2sdk-1.4.2_15-oth-JPR/j2sdk-1.4.2_15-oth-JPR:4/
j2sdk-1_4_2_15-linux-i586.bin

chmod +x j2sdk-1_4_2_15-linux-i586.bin
./j2sdk-1_4_2_15-linux-i586.bin
```

5.2. **Installation of Kowari 1.0.5.**

```
mv ./kowari-1.0.5.tgz /content/echodep/kowari-1.0.5.tgz
tar -zxvf kowari-1.0.5.tgz
cd kowari-1.0.5
```

```
./build.sh dist
```

5.3. **Installation of SWI-Prolog 5.6.37.**

```
cd /content/echodep/source
tar -zxvf pl-5.6.37.tar.gz
cd pl-5.6.37/src
./configure --prefix=/content/echodep/prolog_5.6.34 --enable-mt
make
make install

To install JPL package:
cd pl-5.6.34/packages
./configure
make
make install
```

jpl.jar must be found by any Java VMs (and compilers) used with JPL; one possibility is to put it on your global CLASSPATH Please add directory holding libjvm.so to $LD_LIBRARY_PATH

Additional JPL instructions can be found at: `http://www.swi-prolog.org/packages/jpl/installation.html`

5.4. **Installation of KowariClient.java.** If KowariClient.java must be edited (example: RMIs):

On line 109:

```
String modelURI = "rmi://128.174.154.103.lis.uiuc.edu/server1/#sampledata";
String query = "select $s $p $o from <"+modelURI+"> where $s $p $o ;";
```

On line 178:

```
String modelURI = "rmi://128.174.154.103.lis.uiuc.edu/server1/#sampledata";
String query = "select $s $p $o from <"+modelURI+"> where $s $p $o ;";
```

Then compile the java source:

```
javac KowariClient.java
```

jk.jar supplied by NCSA can be uncompressed and the new compiled file, KowariClient.class, replaces the existing version of the file. A new version of jk.jar can formed with the new class and the two existing files from the initial jk.jar:

```
jar cf jk.jar KowariClient.class META-INF readOwl.class
```

5.5. **Installation of PostgresSQL.** A user needs to be created to install PostgresSQL. I contacted the System Administrator Brynnen Owen and he created the user after I gave him the user name, home directory location, and password.
username =
home = /content/echodep/users/postgres

password = [password]

```
tar -zxvf postgresql-8.2.4.tar.gz
./configure --prefix=/content/echodep/pgsql --enable-debug
make
make install
su - postgres
/content/echodep/pgsql/bin/initdb -D /content/echodep/pgsql/data
/content/echodep/pgsql/bin/postgres -D /content/echodep/pgsql
/data >logfile 2>&1 &
/content/echodep/pgsql/bin/createdb test
/content/echodep/pgsql/bin/psql test
Success. You can now start the database server using:

 /content/echodep/pgsql/bin/postgres -D /content/echodep/pgsql/data
or
 /content/echodep/pgsql/bin/pg_ctl -D /content/echodep/pgsql
/data -l logfile start
```

I actually use:

```
/content/echodep/pgsql/bin/pg_ctl -D /content/echodep/pgsql/data
-l postgres.log start

/content/echodep/pgsql/bin/pg_ctl -D /content/echodep/pgsql/data
-l postgres.log stop
```

Postgresql.config must now be edited (change ports, etc.).

To overcome non-root installation and the fact that I could not chown the directory, data, after creating it I did the following:

As ccurtis3 I issued the following command:

```
chmod o+w pgsql
```

This gave write permission the the directory pgsql.

I issued the command to log in as the user 'postgres':

```
su postgre
```

As postgres I issued the command:

```
mkdir /content/echodep/pgsql/data
```

I logged back in as ccurtis3:

```
su ccurtis3
```

I then removed write access from 'others':

```
chmod o-w pgsql
```

My final result is the same as:

```
root$ mkdir /content/echodep/pgsql/data
root$ chown postgres /content/echodep/pgsql/data
```

Lastly, /content/echodep/pgsql must be added to the .bashrc PATH

5.6. **Installation of Fedora Commons.** Installation of Fedora Commons 2.2.1, adapted from official Fedora Commons documentation:

Step 1: Prepare Environment Variables

The following environment variables must be correctly defined: JAVA_HOME

This should point to the base directory of your Java installation. For UNIX derivatives, this might be something like /usr/local/java-1.5.0-sun FEDORA_HOME

This is the directory where Fedora will be installed, for example, /usr/local/fedora.

Step 2: PATH

This must include the Java and Fedora bin directories. For UNIX derivatives, this will be $FEDORA_HOME/server/bin, $FEDORA_HOME/client/bin and usually $JAVA_HOME/bin.

If you will be building from source, Ant should also also be on your path. JAVA_OPTS

If Fedora is configured to use SSL, JAVA_OPTS must include the javax.net.ssl.trustStore and javax.net.ssl.trustStorePassword properties. See the SSL section below for more information. CATALINA_HOME

If Fedora is configured to use Tomcat, CATALINA_HOME must be set before starting Fedora. If using the quick install option, CATALINA_HOME should be set to $FEDORA_HOME/tomcat (or %FEDORA_HOME%
tomcat in Windows).

Configure Fedora Commons to work with PostgresSQL installation:

Please consult the documentation at `http://www.postgresql.org/docs/` for more detailed information about configuring PostgreSQL.

Launch the PostgreSQL interactive terminal, psql, (optionally appending the -U argument to connect as a different user). psql -d postgres

To create a user 'fedoraAdmin' with password 'fedoraAdmin' and database named 'fedora22', enter the following:

```
CREATE ROLE "fedoraAdmin" LOGIN PASSWORD 'fedoraAdmin'; CREATE DATABASE
"fedora22" WITH ENCODING='UTF8' OWNER="fedoraAdmin";
```

Run the Fedora Commons installation jar:

```
echodep2:/content/echodep/source 540 \$ java -jar fedora-2.2.1-installer.jar
```

I performed a custom install to point to PostgreSQL and"existingTomcat"

5.7. **Ingest demos.**

```
echodep2:/content/echodep/fedora-2.2.1/client/bin 581$
sh fedora-ingest-demos.sh echodep2.lis.uiuc.edu 8080 fedoraAdmin
[password] http
```

```
Starting Fedora DemoIngester...
Ingesting Demonstration Objects...

SUCCESS: All 33 objects were ingested.

A detailed log is at /content/echodep/fedora-2.2.1/client/logs/
ingest-from-dir-1187192238003.xml

SUCCESS: All 9 objects were ingested.

A detailed log is at /content/echodep/fedora-2.2.1/client/logs/
ingest-from-dir-1187192254472.xml

Finished.
```

Demos can be viewed at: `http://echodep2.lis.uiuc.edu:8080/fedora/get/demo:5` Search service is at at: `http://echodep2.lis.uiuc.edu:8080/fedora/search`

In Fedora Commons 2.2.1 installer the Resource Index (Kowari is the default) is turned off by default. I edited fedora.fcfg to set the Resource Index level at 2:

```
<module role="fedora.server.resourceIndex.ResourceIndex"
class="fedora.server.resourceIndex.ResourceIndexModule">
 <comment>Supports the ResourceIndex.</comment>
 <param name="level" value="2">
 <comment>(required)
 Index level. Currently, only 0, 1 and 2 are supported
 levels. 0 = off: do not load the ResourceIndex 1 = basic: system
 metadata, RELS-EXT, disseminations 2 = basic + method permutations
 WARNING: changing the level (except to 0) requires
 running the Resource Index Rebuilder.</comment>
 </param>
```

From Fedora Commons 2.2.1 documentation:

> * level
> Sets the operating level of the Resource Index. 0 Off: the Resource Index will not load at server startup.
> 1 On: the Resource Index will index system properties, inter & intra-object relationships, and user-defined relationships.
> 2 Same as 1, but adds indexing of parameterized methods, where the parameters have a finite domain (e.g., the method getImage with parameter size, whose domain is small, large). Because calculating method parameters may result in a combinatorial explosion of statements in the Resource Index (depending on the design of

a particular repository's Behavior Definition Objects), this level of indexing must be explicitly set.
* datastore
The id of the datastore to use with the Resource Index. The referenced datastore must assert a connectorClassName parameter with a valid Trippi Connector class.
* syncUpdates
Whether to flush the triple buffer before returning from object modification operations. This defaults to false. Specifying this as true will ensure that RI queries immediately reflect the latest triples. Specifying false will not provide this guarantee, but can reduce roundtrip time for API-M operations (especially when using Kowari).
Fedora Commons offical website

I then had to run the Resource Index Rebuilder:
I stopped Fedora by stopping Tomcat:

```
$CATALINA_HOME/bin/shutdown.sh
```

I then continued by using the rebuild binary in the Fedora Commons /server directory:

```
echodep2:/content/echodep/fedora-2.2.1/server/bin 515 $ ls
fedora-rebuild.bat fedora-reload-policies.bat validate-policy.bat
fedora-rebuild.sh fedora-reload-policies.sh validate-policy.sh
echodep2:/content/echodep/fedora-2.2.1/server/bin 516 $
sh fedora-rebuild.sh
```

I then restarted Tomcat:

```
$CATALINA_HOME/bin/startup.sh
```

The Fedora Resource Index Search interface is available at: `http://echodep2.lis.uiuc.edu:8080/fedora/risearch`

Fedora Commons 2.2.1 can be remotely administered through the use of the shell script:

```
$FEDORA_HOME/client/bin/fedora-admin.sh
```

5.8. **Editing of .bash_profile and .bashrc.** In order for the installed applications to be correctly called at BASH two files in the home directory of 'ccurtis3'and 'postgres': .bash_profile and .bashrc:

```
export KOWARI_HOME=/content/echodep/kowari-1.0.5
export JAVA_HOME=/content/echodep/jdk1.5.0_12
export JAVADOC=/content/echodep/jdk1.5.0_12/bin
export JRE_HOME=/content/echodep/jdk1.5.0_12/jre
export JAVA_VM=/content/echodep/jdk1.5.0_12/jre/bin
export JAVAC=/content/echodep/jdk1.5.0_12/bin/javac
export FEDORA_HOME=/content/echodep/fedora-2.2.1
```

```
export CATALINA_HOME=/content/echodep/apache-tomcat-5.5.23
export TERM=xterm

export PATH=${FEDORA_HOME}/bin:/content/echodep/pgsql/bin:${JAVA_HOME}/bin:
${JAVA_VM}:${JRE_HOME}:/content/echodep/prolog5.6.37/bin:/usr/local/bin:
/usr/bin:/bin

export CLASSPATH=/content/echodep/chad_files/dev/java_dev/echodep/classes:
${JAVAC}:${KOWARI_HOME}:/content/echodep/dev/java_dev/echodep/classes
:${JRE_HOME}/lib/jpl.jar:${KOWARI_HOME}/dist/driver-1.0.5.jar:
${JAVA_HOME}/lib/jpl.jar:${KOWARI_HOME}/dist/jk.jar
:${KOWARI_HOME}/dist/kowari-1.0.5.jar:
${KOWARI_HOME}/dist/client-jena-base-1.0.5.jar:
${KOWARI_HOME}/lib/saaj-1.1.jar:${KOWARI_HOME}/lib/log4j-1.2.8.jar:
${KOWARI_HOME}/lib/emory-util.jar:${JAVA_HOME}/lib/tools.jar:
${KOWARI_HOME}/dist/itql-1.0.5.jar:${KOWARI_HOME}/lib/apache-soap-2.2.jar:
${KOWARI_HOME}/lib//jrdf-0.3.3.jar:${KOWARI_HOME}/lib/jargs-0.2.jar:
usr/local/encap/ccurtis3_install/kowari-1.0.5/conf:${KOWARI_HOME}/dist/:.

export LD_LIBRARY_PATH=${JRE_HOME}/lib/amd64:${JRE_HOME}/lib/amd64/server
```

6. Instructions to Maintain Environment

List all processes:

```
$ ps axu
```

Kill process:

```
$ kill [PID]
```

Port overview:
8080 Tomcat 5.5, Fedora
8081 RMI Server for Kowari 1.0.5
8082 Kowari 1.0.5
8083 Postgres
8084 Mulgara
8085 RMI server for Mulgara
8086 Apache
1099 Fedora/Kowari RMI server

6.1. **Kowari.** Kowari uses port 8082

Start server:

```
cd /content/echodep/kowari-1.0.5/dist
java -jar kowari-1.0.5.jar -p 8082 &
```

(One usually can add the space and ampersand '&' at the end of a command to run a process in the background. This was not working for Kowari, so I issued the following command in order to keep the Kowari up after logging out:

```
$ nohup java -jar kowari-1.0.5.jar -p 8082 &
```

Start ITQL console through X11 server:

```
in Apple X11.app: ssh -X -Y ccurtis3@echodep2.lis.uiuc.edu
$ cd /content/echodep/kowari-1.0.5/dist
$ java -jar itql-1.0.5.jar
```

OR

Use Kowari Viewer, an HTTP interface: `http://echodep2.lis.uiuc.edu:8082/webui/GetViewerScreen.event`

6.2. **PostgreSQL.** PostgreSQL must be run as user 'postgres' It uses port 8081.

```
/content/echodep/pgsql/bin/pg_ctl -D /content/echodep/pgsql/data
-l postgres.log start

/content/echodep/pgsql/bin/pg_ctl -D /content/echodep/pgsql/data
-l postgres.log stop
```

6.3. **Tomcat.** must be run as ccurtis3 uses port 8080

```
$CATALINA_HOME/bin/startup.sh
$CATALINA_HOME/bin/shutdown.sh
```

6.4. **Kowari Client.pl.** See

```
kowari_answer([],R).


create <rmi://128.174.154.103.lis.uiuc.edu:8081/server1#semweb>;

load <http://www.arches.uga.edu/\~vstaub/GlobalInfoSys/project/ontology/
Could_have_been.rdf> into
<rmi://128.174.154.103.lis.uiuc.edu:8081/server1#semweb>;

kowari([],R).

select $subject $predicate $object from
<rmi://128.174.154.103.lis.uiuc.edu:8081/server1#semweb>
where $subject $predicate $object;

drop <rmi://128.174.154.103.lis.uiuc.edu:8081/server1#semweb>;
```

6.5. **Fedora Commons.** Fedora Remote Graphical Administration

Fedora Core has several graphical administration applications, such as system-config-users, system-config-time, and system-config-services. In order to forward these applications over an ssh connection, you must use the -Y flag. The -Y enables trusted X11 forwarding over ssh.

Terminal.app is fine, provided the DISPLAY variable is set in the environment appropriately (try 127.0.0.1:0.0 for default install), before you do ssh -X

Add to .profile:

```
export DISPLAY=:0.0
```

in Apple Tiger X11.app:

```
ssh -X -Y ccurtis3@echodep2.lis.uiuc.edu

$ cd /content/echodep/fedora-2.2.1/client/bin
$ sh fedora-admin.sh
```

X11.app doesn't even need to be open until you wish to launch an X11 application on your fedora box. Then it'll need to be open - great if you like Terminal.app more than xterm. (This is all under OSX 10.4.x Leopard 10.5 changed the X11 code-base, so this may differ in the future)

On Linux use xhost +

The Fedora Resource Index Search interface is available at: `http://echodep2.lis.uiuc.edu:8080/fedora/risearch`

7. Technical Issues and Solutions

(1) Lack of Root Access: I was restricted to having rwx access to /usr/local/encap. This made installation more difficult.

Solution: (1) restricting installation into /usr/local/encap/ccurtis3_install and duplicating applications only when necessary, e.g. Java. (2) edit .bash_profile and .bashrc in home directory (/homea/ccurtis3) to include the paths to each binary.

(2) JPL make errors: The make file for JPL would not form correctly.

Solution: I edited the following lines in the make file located in /usr/local/encap/ccurtis3_install/source/pl-5.6.34/packages/jpl/src/java/Makefile:

```
####/content/echodep/source/pl-5.6.37/packages/jpl/src/java####

JAVAC=/usr/local/encap/ccurtis3_install/j2sdk1.4.2_14/bin/javac
JAR=jar
JUNIT=/usr/share/java/junit.jar
JAVADOC=/usr/local/encap/ccurtis3_install/j2sdk1.4.2_14/bin/javadoc
```

(3) Kowari Gmake error: I installed Java 1.6.0 for my installation environment. But through failed installation attempts I found out that Kowari will not build on Java later 1.4.2. Development for Kowari stopped in 2006 and a fork, Mulgara, took on development. I confirmed the fact through the Kowari mailing list which

unresolvable Problems with Java post-1.4.x caused the Kowari development team to stick to 1.4.2:

> Re: [Kowari-developers] building problems From: Paul Gearon ¡gearon@ie...¿ - 2006-06-21 16:06 Hi Alessandro,
>
> On 6/21/06, Alessandro Di Bella ¡aldib@fu...¿ wrote: Hi, I tried to build the CVS HEAD for kowari 1.1 and 1.2 and I encountered the following few problems. I'll detailed below:
>
> Kowari doesn't support Java 1.5. It won't compile. This has been well known for some time. Unfortunately, I note that the README file says: Note. You must use J2SE 1.4.2 or above for compiling and running Kowari. Obviously this was written before 1.5 came out. The problem is that the modifications which make Kowari work on Java 1.5 also break it on Java 1.4. The decision was made some time ago to keep 1.4 compatibility.

Solution: Install an older version of Java (1.4.x) and try to maintain the rest of the environment on the same version. I kept Java 1.6.0 installed for later use.

(4) SWI-Prolog JPL segfaults: The following appears when JPL is called:

```
ERROR: (/usr/local/encap/ccurtis3_install/prolog_5.6.34/lib/pl-5.6.34/
library/jpl.pl:3733):
 read_clause/2: Caught signal 11 (segv)
ERROR: (/usr/local/encap/ccurtis3_install/prolog_5.6.34/lib/pl-5.6.34/
library/jpl.pl:4119):
 read_clause/2: Caught signal 11 (segv)
```

Solution: It is apparently common right now and there has not been a resolution on the mailing-lists.

Regarding the "Caught signal 11 (segv)" error: segmentation fault - An error in which a running program attempts to access memory not allocated to it and core dumps with a segmentation violation error. This is often caused by improper usage of pointers in the source code, dereferencing a null pointer, or (in C) inadvertently using a non-pointer variable as a pointer.

(5) Incorrect Java parameters: The following describes the problem with running kowari.pl:

```
kowari(Query,Result) :- /* relation between iTQL query and result */
 jpl_new( 'KowariClient', [], F ),
 jpl_call( F, executeQuery, Query, Answer),
 consume_answer(Answer,Result).
```

kowari(Q,R) The kowari method uses two arguments: the Query variable (bound to an iTQL query string), and the unbound variable Result. The kowari method produces Result, a table form of a Kowari answer.

jpl_new(+Class,+Params,-Ref) jpl_new creates an instance of the java class, KowariClient, which is set in the CLASSPATH, a list of actual parameters for the constructor, which is empty here, which binds F to a new object reference.

jpl_call(+Ref, +Method, +Params, -Result) jpl_call calls the executeQuery method of the object to which F refers (KowariClienbt class), effectively passing it the Java value (Query variable).

consume_answer(Answer,Result)

This predicate continues the process to the next method.

I cannot get to that method due to the error when using example1(R). and example2(R).:

```
jpl_call/4: Type error: ‘method_params’ expected, found
‘select $sampledata subquery(select $prevn ’1’ from
<rmi://gserve102.lis.uiuc.edu/server1\#sampledata>
where walk($sampledata <http://sampledata\#prev> $o and $s
<http://sampledata\#prev> $prevn) and $prevn
<http://sampledata\#terms> ’1’) from
<rmi://gserve102.lis.uiuc.edu/server1\#sampledata>
where $sampledata $p $o;’ (3rd arg must be a proper list
of actual parameters for the named method)
```

In other words, kowari.pl wants a proper list for the Query variable, while the executeQuery method, in the Java class KowariClient, requires a string. If the diagnosis is correct there needs to be a ’list to string’ conversion somewhere. I thought about the Prolog predicate, string_to_list(String, List), but with a modified kowari.pl with the mentioned predicate I get an error: (not all actual parameters are convertible to Java values or references).

Solution: Joe Futrelle experienced the same error and revised the code.

(6) log4 errors:

```
kowari(Query,Result) :-
 jpl_new( ’edu.uiuc.kowari_client.KowariClient’, [], F ),
 jpl_call( F, executeQuery, [Query], Answer),
 consume_answer(Answer,Result).

Executing ’select $s $p $o from <rmi://gserve102.lis.uiuc.edu/
server1\#sampledata> where $s $p $o;’
log4j:WARN No appenders could be found for logger
(org.kowari.itql.ItqlInterpreter).
log4j:WARN Please initialize the log4j system properly.
ERROR: jpl_call/4: Type error: ‘method_params’ expected,
found ‘[2]’ (not all actual parameters are convertible
 to Java values or references)
Exception: (13) consume_column
```

```
(@'J\#00000000000137080236', 2, _G2181) ?
```

Due to the low priority of the logging error for the ItqlInterpreter this error was not resolved. Superficial analysis implies the wrong file/directory is being pointed at.

```
<TucanaConfig>

 <!-- Paths to external component configuration, relative to JAR file -->
 <ExternalConfigPaths>
 <TucanaLogging>conf/log4j-kowari.xml</TucanaLogging>
 <WebDefault>conf/webdefault.xml</WebDefault>
 </ExternalConfigPaths>
```

8. Testing/Findings

8.1. **Non-Root Environment.** It is completely feasible to create a functional non-root environment for testing research software in semantic archiving. Due to an ever increasing awareness of security vulnerabilities of systems, it is highly probable that further projects will have students and faculty who do not have root access or have requests denied by system administrators. Therefore it is vital to be able to perform tasks under limited permission. The documentation of the testing environment at the Graduate School of Library and Information Science demonstrates the feasibility and reproducibility of such an environment.

8.2. **Solitary Kowari.** Through the contribution of the Tupelo project, a BECHAMEL application can issue iTQL queries to a solitary Kowari instance. The KowariClient written in SWI-Prolog, using the JPL package, can call methods from a Java class that uses the ItqlInterperter.

The methods mirror iTQL query actions:
Create model
Load RDF into model
Query Model
Dump RDF out of model
Delete model

The following is an example of a session with output:

```
echodep2:/content/echodep/chad_files/kowari_pl_fix_20070723 503 $ pl
% /homea/ccurtis3/.plrc compiled 0.00 sec, 264 bytes

Welcome to SWI-Prolog (Multi-threaded, Version 5.6.37)
Copyright (c) 1990-2007 University of Amsterdam.
```

```
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic). or ?- apropos(Word).

?- [kowari].
% kowari compiled 0.00 sec, 8,672 bytes

Yes
?- kowari_answer('create <rmi://128.174.154.103.lis.uiuc.edu:
8081/server1#semweb>;',R).
Executing 'create <rmi://128.174.154.103.lis.uiuc.edu:
8081/server1#semweb>;'
log4j:WARN No appenders could be found for logger
(org.kowari.itql.ItqlInterpreter).
log4j:WARN Please initialize the log4j system properly.

R = @null

Yes
?- kowari_answer('load <http://www.arches.uga.edu/\~vstaub/
GlobalInfoSys/project/ontology/Could_have_been.rdf>
into <rmi://128.174.154.103.lis.uiuc.edu:8081/server1#semweb>;',R).
Executing 'load <http://www.arches.uga.edu/\~vstaub/
GlobalInfoSys/project/ontology/Could_have_been.rdf>
into <rmi://128.174.154.103.lis.uiuc.edu:8081/server1#semweb>;'

R = @null

Yes
?- kowari('select $subject $predicate $object
from <rmi://128.174.154.103.lis.uiuc.edu:8081/server1#semweb>
where $subject $predicate $object;',R).
Executing 'select $subject $predicate $object from
<rmi://128.174.154.103.lis.uiuc.edu:8081/server1#semweb>
where $subject $predicate $object;'

R = [['http://ainge.cs.uga.edu/gis/publications#Could_have_been_10059',
'http://ainge.cs.uga.edu/gis/publications#author_Name', shahLink],
['http://ainge.cs.uga.edu/gis/publications#Could_have_been_10059',
'http://www.w3.org/TR/1999/PR-rdf-schema-19990303#label', shahLink],
```

```
['http://ainge.cs.uga.edu/gis/publications#Could_have_been_10059',
'http://www.w3.org/1999/02/22-rdf-syntax-ns#type',
'http://ainge.cs.uga.edu/gis/publications#Author_Link'],
['http://ainge.cs.uga.edu/gis/publications#Could_have_been_10058',
'http://ainge.cs.uga.edu/gis/publications#has_downloads',
'http://www.arches.uga.edu/\~vstaub/
GlobalInfoSys/project/ontology/ns1;Could_have_been_10029'],
['http://ainge.cs.uga.edu/gis/publications#Could_have_been_10058',
'http://ainge.cs.uga.edu/gis/publications#from_project',
'http://ainge.cs.uga.edu/gis/publications#Could_have_been_00043'],
['http://ainge.cs.uga.edu/gis/publications#Could_have_been_10058',
'http://ainge.cs.uga.edu/gis/publications#has_citations',
'http://www.arches.uga.edu/\~vstaub/
GlobalInfoSys/project/ontology/ns1;Could_have_been_10013'],
['http://ainge.cs.uga.edu/gis/publications#Could_have_been_10058',
'http://ainge.cs.uga.edu/gis/publications#has_authors'|...],
['http://ainge.cs.uga.edu/gis/publications#Could_have_been_10058'|...],
[...|...]|...]

Yes
?-
% halt
```

8.3. **Embedded Kowari.** Fedora uses Kowari, but it is misleading in that it makes it seem as though a near full installation of Kowari exists. Instead it is highly embedded and extremely difficult to work with anything outside of the standard configuration. Fedora's perspective is as follows:

> The combination of representing explicit relationships as RDF/XML in a datastream of a digital object and then mapping them to the Kowari triple store offers the best of both worlds. The explicit representation provides the basis for exporting, transporting, and archiving of the digital objects with their asserted relationships to other objects. The mapping to Kowari provides a graph-based index of an entire repository and the basis for high-performance queries over the relationships.
>
> Our preliminary report on query performance of various triple store technologies is available at `http://tripletest.sourceforge.net/2005-06-08/index.html` We are planning a future publication that explores these results and their implications. The added advantage of the dual representation is that the entire triple store can be rebuilt by importing and parsing the XML-based digital objects. Lagoze et al. (2006)

A project in the United Kingdom, RepoMMan released "D-D8 Report on experiences with Fedora during the first year" and commented that

> Fedora has an integrated Kowari datastore to provide a searchable graph of relationships between objects, the so-called resource index. The Kowari datastore is not easily amenable to being replaced Green (2006)

More specifics of the use of the Fedora digital object as a carrier of metadata is written in the 2005 whitepaper, "Fedora Open Source Repository Software":

> As stated earlier, each Fedora digital object has a slot for metadata that expressing the relationship of the object to other digital objects, and in fact to information entities outside of Fedora (for example, external web pages). This feature makes use of state- of-the-art semantic web technologies. Relationships are stored within a special datastream in a digital object as statements encoded in the Resource Description Format (RDF) XML syntax. These relationships may be derived from any ontology, including a basic relationship ontology supplied with Fedora. The Fedora system automatically indexes the relationship metadata from all digital objects in a special database. This database can be queried using a query language specialized for extracting information from a relationship graph. This query interface is exposed as a service in the Fedora API, and in fact can be used for dynamic disseminations like any web service. Fedora Development Team (2005)

The interface to query these relationships is limited If one wants to edit the relationships, perform complex queries (inferencing)

> This relationship metadata is automatically indexed into the relationship store. As a result, as shown in Figure 5, demo:10 can have a URL-accessible dissemination that queries the relationship store to return its list of members. Note that these queries can traverse the entire relationship graph, not just information local to the relationship metadata in an individual digital object. Fedora Development Team (2005)

8.4. **Remote Kowari and Fedora/Trippi.** Due to the fact that Kowari is embedded in Fedora and uses Trippi for writing triples, and the SWI-Prolog application connects to an RMI server, it seemed optimal to use the solitary instance of Kowari used in prior testings. Fedora documentation explains how to configure Fedora in order to use a remote instance of Kowari. Unfortunately, due to the fact that Fedora uses Kowari, which is no longer is development, the classes in Trippi and Kowari are different and incompatible. Due to current development on Fedora 3.0, the focus is on creating compatibility with Mulgara, therefore Kowari-specific issues are not being addressed.

The following was submitted to the Fedora Commons mailing-list on Sourceforge. There were no replies:

From: Chad Curtis ¡ccurtis3@ui...¿ - 2007-09-13 18:53

After extensive searching and troubleshooting I'm seeking help for triplestore rebuilding issues.

BACKGROUND: My first goal is to send an iTQL query to the embedded Kowari in Fedora Commons 2.2.1, using an application developed at my university. When I use the application to query a stand-alone Kowari installation, I have no problems regardless the version of Kowari. My

first goal was unfruitful as I could not find enough information on what is missing/patched in the embedded Kowari, except on the Trippi side of things. I put the first goal to the side. My second goal is to set up a remote Kowari installation as the Resource Index for Fedora Commons, then query the remote Kowari through my application. I need to emphasize that the purpose of the project it to bypass the risearch interface/Trippi. Eventually the same application will query Mulgara in Fedora Commons 3.0.

Environment: Debian Etch Kowari 1.0.5 Fedora 2.2.1 Java 1.5.0_12 Tomcat 5.0.28

ISSUE: When rebuilding a remote Kowari the following exceptions, pasted below, are thrown. After searching mailing list archives I found someone who had the exact same errors in Jan. 2007, but the issue was never addressed, at least publicly. I tried contacting the SourceForge user, but did not receive an answer. At first I thought it would be a firewall/permissions/policy issue, but it seems to be related to Fedora/Trippi code. The fact that fedora.common.rdf.FedoraViewNamespace does not implement serialization looks like a starting point:

```
INFO [main] (ConcurrentTriplestoreWriter.java:327) - Closing...
org.kowari.query.QueryException: Java RMI failure
at
org.kowari.server.rmi.RemoteSessionWrapperSession.testRetry
(RemoteSessionWrapperSession.java:735)
at org.kowari.server.rmi.RemoteSessionWrapperSession.insert
(RemoteSessionWrapperSession.java:233)
at org.trippi.impl.kowari.KowariSession.doTriples
(KowariSession.java:142)
at org.trippi.impl.kowari.KowariSession.add
(KowariSession.java:128)
at org.trippi.impl.base.MemUpdateBuffer.writeBatch
(MemUpdateBuffer.java:199)
at org.trippi.impl.base.MemUpdateBuffer.flush
(MemUpdateBuffer.java:123)
at
```

```
org.trippi.impl.base.ConcurrentTriplestoreWriter.flushBuffer
(ConcurrentTriplestoreWriter.java:272)
at org.trippi.impl.base.ConcurrentTriplestoreWriter.close
(ConcurrentTriplestoreWriter.java:330)
at org.trippi.impl.kowari.KowariConnector.close
(KowariConnector.java:287)
at fedora.server.resourceIndex.ResourceIndexImpl.close
(ResourceIndexImpl.java:379)
at fedora.server.resourceIndex.ResourceIndexRebuilder.finish
(ResourceIndexRebuilder.java:193)
at fedora.server.utilities.rebuild.Rebuild.<init>
(Rebuild.java:116)
at fedora.server.utilities.rebuild.Rebuild.main(Rebuild.java:

367)
Caused by: java.rmi.MarshalException: error marshalling arguments;
nested exception is:
java.io.NotSerializableException:
fedora.common.rdf.FedoraViewNamespace
at sun.rmi.server.UnicastRef.invoke(UnicastRef.java:122)
at org.kowari.server.rmi.RemoteSessionImpl_Stub.insert
(Unknown Source)
at org.kowari.server.rmi.RemoteSessionWrapperSession.insert
(RemoteSessionWrapperSession.java:228)
... 11 more
Caused by: java.io.NotSerializableException:
fedora.common.rdf.FedoraViewNamespace
at java.io.ObjectOutputStream.writeObject0
(ObjectOutputStream.java:1081)
at java.io.ObjectOutputStream.defaultWriteFields
(ObjectOutputStream.java:1375)
at java.io.ObjectOutputStream.writeSerialData
(ObjectOutputStream.java:1347)
at java.io.ObjectOutputStream.writeOrdinaryObject
(ObjectOutputStream.java:1290)
at java.io.ObjectOutputStream.writeObject0
(ObjectOutputStream.java:1079)
at java.io.ObjectOutputStream.defaultWriteFields
(ObjectOutputStream.java:1375)
at java.io.ObjectOutputStream.writeSerialData
(ObjectOutputStream.java:1347)
at java.io.ObjectOutputStream.writeOrdinaryObject
```

```
(ObjectOutputStream.java:1290)
at java.io.ObjectOutputStream.writeObject0
(ObjectOutputStream.java:1079)
at java.io.ObjectOutputStream.writeObject
(ObjectOutputStream.java:302)
at java.util.HashSet.writeObject(HashSet.java:254)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke
(NativeMethodAccessorImpl.java:39)
at sun.reflect.DelegatingMethodAccessorImpl.invoke
(DelegatingMethodAccessorImpl.java:25)
at java.lang.reflect.Method.invoke(Method.java:585)
at java.io.ObjectStreamClass.invokeWriteObject
(ObjectStreamClass.java:917)
at java.io.ObjectOutputStream.writeSerialData
(ObjectOutputStream.java:1339)
at java.io.ObjectOutputStream.writeOrdinaryObject
(ObjectOutputStream.java:1290)
at java.io.ObjectOutputStream.writeObject0
(ObjectOutputStream.java:1079)
at java.io.ObjectOutputStream.writeObject
(ObjectOutputStream.java:302)
at sun.rmi.server.UnicastRef.marshalValue(UnicastRef.java:258)
at sun.rmi.server.UnicastRef.invoke(UnicastRef.java:117)
... 13 more
WARN [main] (ConcurrentTriplestoreWriter.java:334) - Error flushing

update buffer while closing Triplestore: org.trippi.TrippiException:
Error adding triples: org.kowari.query.QueryException: Java RMI failure
INFO [main] (ConfigurableSessionPool.java:218) - Closing all
sessions...
INFO [main] (KowariSession.java:310) - Closed session.
INFO [main] (KowariSession.java:310) - Closed session.
INFO [main] (KowariSession.java:310) - Closed session.
INFO [main] (KowariSessionFactory.java:253) - Closing underlying
SessionFactory...
Finished.
```

8.5. **JPL memory handling.** JPL has threading issues with JVM and is unstable and not scalable. Initially the JPL-based Prolog code used on the Prolog to Kowari interface was also going to be used for an interface of BECHAMEL to Tupelo. Memory management issues in JPL has halted further development on a

JPL-based BECHAMEL to Tupelo interface. Plans for the interface are forthcoming.

9. Glossary

Information retrieved and cited from websites, wikis, and papers of projects.

9.1. **BECHAMEL.** The BECHAMEL system is a knowledge representation and inference environment for expressing and testing semantic rules and constraints for markup languages. Written in Prolog, the system provides predicates for processing the syntactic structures that emerge from a SGML/XML parser, defining object classes, instantiating object instances, assigning values to properties, and establishing relationships between or among object instances. BECHAMEL uses Prologs built-in capabilities to derive inferences from these facts.

Part of the ongoing development of BECHAMEL involves experimenting with strategies for mapping syntactic relations to object relations and properties. The paper Öbject mapping for markup semanticsdescribes the current strategy, based on a blackboard model. Advantages of this approach include context free rules and the potential to exploit parallel processing for scalability. It has the drawback, however, of not permitting evidence to be described in ways people are likely to find natural or familiar. By using the current approach to produce formal accounts of the semantics of popular markup languages, we hope to learn a great deal about the ways markup syntax typically cues semantic relationships. That advance in our understanding will inform the development of more usable languages for object mapping. Dubin (2003)

9.2. **Fedora Commons.** Fedora Commons is an open-source, digital repository service that provides the foundation for many types of information management systems. The Fedora platform is logically divided into four major functional areas that reflect its first principles:

1. repository services 2. preservation services 3. semantic services 4. enterprise services

These are critical services that should be offered by any platform whose purpose is to enable collaborative applications while attending to the challenges of information management and preservation. Using a standards-based, service-oriented architecture, the Fedora platform provides an extensible framework of service components to support features such as OIA-PMH, search engine integration, messaging, workflow, format conversion, bulk ingest, and others. In addition, features such as authentication, fine-grained access control, content versioning, replication, integrity checking, dynamic views of digital objects, and more are incorporated into the Fedora repository service. Fedora provided services can be seamlessly integrated into an organization's existing infrastructure, protecting and enhancing prior investments.

Many of Fedora's features exploit its flexible and extensible digital objects, which are containers for metadata, one or more representations of the content and relationships to other information resources. Fedora's digital objects provide "Lego-like" building blocks to support uniform management and access to heterogeneous content including books, images, articles, datasets, multi-media, and more. Access to the digital object is provided by disseminators, which can simply deliver a desired portion of the digital object or can deliver a customized view. Fedora's digital objects are self-describing and self-delivering-key features that enable preservation.

9.3. **Kowari Metastore.** An open source, massively scalable, transaction-safe, purpose-built database for the storage and retrieval of metadata.

Much like a relational database, one stores information in Kowari and retrieves it via a query language. Unlike a relational database, Kowari is optimized for the storage and retrieval of many short statements (in the form of subject-predicate-object, like "Kowari is fun" or "Kowari imports RDF"). Kowari is not based on a relational database due to the large numbers of table joins encountered by relational systems when dealing with metadata. Instead, Kowari is a completely new database optimized for metadata management. `http://www.kowari.org/oldsite/1061.htm`

9.4. **JRDF.** JRDF is an attempt to create a standard set of APIs and base implementations to RDF (Resource Description Framework) using the latest version of the Java language.

9.5. **N3 (Notation 3).** Notation-3 is more user-firendly serialization of RDF.

9.6. **N-Triples.** N-Triples is a line-based, plain text format for encoding an RDF graph. It was designed to be a fixed subset of N3[N3] [N3-Primer] and hence N3 tools such as cwm [CWM], n-triples2kif [N-TRIPLES2KIF], and Euler [EULER] can be used to read and process it. cwm can output this format when invoked as "cwm -ntriples". `http://www.w3.org/TR/rdf-testcases/`

9.7. **RDF.** The Resource Description Framework (RDF) is a language for representing information about resources in the World Wide Web. It is particularly intended for representing metadata about Web resources, such as the title, author, and modification date of a Web page, copyright and licensing information about a Web document, or the availability schedule for some shared resource. However, by generalizing the concept of a "Web resource", RDF can also be used to represent information about things that can be identified on the Web, even when they cannot be directly retrieved on the Web. Examples include information about items available from on-line shopping facilities (e.g., information about specifications, prices, and availability), or the description of a Web user's preferences for information delivery.

RDF is intended for situations in which this information needs to be processed by applications, rather than being only displayed to people. RDF provides a

common framework for expressing this information so it can be exchanged between applications without loss of meaning. Since it is a common framework, application designers can leverage the availability of common RDF parsers and processing tools. The ability to exchange information between different applications means that the information may be made available to applications other than those for which it was originally created.

RDF is based on the idea of identifying things using Web identifiers (called Uniform Resource Identifiers, or URIs), and describing resources in terms of simple properties and property values. This enables RDF to represent simple statements about resources as a graph of nodes and arcs representing the resources, and their properties and values.

9.8. **SWI-Prolog.** SWI-Prolog is an open source implementation of the programming language Prolog, commonly used for teaching. SWI-Prolog has been under continuous development since 1987. Its main author is Jan Wielemaker. It has a rich set of features, libraries (including its own GUI library, XPCE), developer tools (including an IDE supporting a GUI debugger and a GUI profiler), and extensive documentation. SWI-Prolog runs on Unix, Windows and Macintosh platforms. `http://en.wikipedia.org/wiki/SWI-Prolog`

9.9. **Trippi.** Trippi (pronounced, 'tri-pE) is a Java library providing a consistent, thread-safe access point for updating and querying a triplestore. It is similar in spirit to JDBC, but for RDF databases.

In addition to the API, the Trippi distribution comes with two higher-level applications for working with triplestores: A console and a web service.

Trippi connectors currently exist for Sesame, Kowari, Oracle Spatial, and MPT-Store. See the Trippi Implementation Guide for details on how to write a Trippi adapter for your own triplestore software.

9.10. **Tupelo.** Tupelo is a data and metadata archiving system based on semantic web technologies. Tupelo provides a variety of generic utilities for managing large RDF graphs using best-of-breed RDF database implementations such as Kowari. (We plan to support Mulgara.)

Tupelo is designed for archiving large-scale, complex scientific data and metadata collections. It is also suitable for more conventional digital libraries containing Dublin Core or other standard digital library metadata schemas. Its RDF-based metadata framework can support a wide variety of schemas, from simple, flat-namespace schemas such as Dublin Core, to hierarchical models derived from XML Schema, to more web-like models derived from RDF variants such as RSS. If you can describe it using entity-relation graphs, you can store it in Tupelo. `http://dlt.ncsa.uiuc.edu/wiki/index.php/Main_Page`

References

Dubin, D. (2003). Object mapping for markup semantics. In B. T. Usdin (Ed.) *Proceedings of Extreme Markup Languages 2003*. Montreal, Quebec.
URL http://www.mulberrytech.com/Extreme/Proceedings/html/2003/Dubin01/EML2003Dubin01-toc.html

Dubin, D. (2007). Metadata analysis for digital preservation (poster session). In *Association for Library and Information Science Education 2007*. Association for Library and Information Science Education.

Dubin, D., & Birnbaum, D. (2004). Interpretation beyond markup. In B. T. Usdin (Ed.) *Proceedings of Extreme Markup Languages 2004*. Montreal, Quebec.
URL http://www.mulberrytech.com/Extreme/Proceedings/html/2004/Dubin01/EML2004Dubin01.html

Dubin, D., Sperberg-McQueen, C. M., Renear, A., & Huitfeldt, C. (2003). A logic programming environment for document semantics and inference. *Literary and Linguistic Computing*, *18*(2), 225–233. (This is a corrected version of an article that appeared in 18:1 pp. 39-47).
URL http://llc.oxfordjournals.org/cgi/reprint/18/2/225

Fedora Commons offical website (2007). http://www.fedora-commons.org/about/.
URL http://www.fedora-commons.org/about/

Fedora Commons System Documentation (2007). http://www.fedora.info/download/2.2.1/userdocs/.
URL http://www.fedora.info/download/2.2.1/userdocs/

Fedora Development Team (2005). Fedora open source repository software. Tech. rep., Cornell University and University of Virginia.
URL http://fedora.info/documents/WhitePaper/FedoraWhitePaper.pdf

Green, R. (2006). Repomman project: D-d8 report on experiences with fedora during the first year. Tech. rep., The University of Hull.
URL http://www.hull.ac.uk/esig/repomman/downloads/D-D8-fedora-exp-v10.pdf

JPL (2007).
URL http://www.swi-prolog.org/packages/jpl/

Kowari MetaStore (2005). http://www.kowari.org/oldsite/1061.htm.
URL http://www.kowari.org/oldsite/1061.htm

Lagoze, C., Payette, S., Shin, E., & Wilper, C. (2006). Fedora: an architecture for complex objects and their relationships. *International Journal on Digital Libraries*, *V6*(2), 124–138.
URL http://portal.acm.org/citation.cfm?id=1124652

Payette, S. (2007). Fedora commons proposal to the gordon and betty moore foundation. Tech. rep., Cornell University.

Renear, A., Dubin, D., Sperberg-McQueen, C. M., & Huitfeldt, C. (2002). Towards a semantics for XML markup. In R. Furuta, J. I. Maletic, & E. Munson (Eds.)

Proceedings of the 2002 ACM Symposium on Document Engineering, (pp. 119–126). McLean, VA: Association for Computing Machinery.
URL http://doi.acm.org/10.1145/585058.585081

Renear, A., Dubin, D., Sperberg-McQueen, C. M., & Huitfeldt, C. (2003). XML semantics and digital libraries. In C. C. Marshall, G. Henry, & L. Delcambre (Eds.) *Proceedings of the third ACM/IEEE-CS joint conference on Digital libraries*, (pp. 303 – 305). Los Alamitos, CA: IEEE.
URL http://portal.acm.org/citation.cfm?id=827192

Sperberg-McQueen, C. M., Dubin, D., Huitfeldt, C., & Renear, A. (2002). Drawing inferences on the basis of markup. In B. T. Usdin, & S. R. Newcomb (Eds.) *Proceedings of Extreme Markup Languages 2002*. Montreal, Quebec.
URL http://www.w3.org/People/cmsmcq/2002/EML2002Sper0518.final

SWI-Prolog (2007). http://www.swi-prolog.org/hcs.html.
URL http://www.swi-prolog.org/hcs.html

Trippi (2007). http://trippi.sourceforge.net/.
URL http://trippi.sourceforge.net/

Tupelo (2007). http://dlt.ncsa.uiuc.edu/wiki/index.php/overview.
URL http://dlt.ncsa.uiuc.edu/wiki/index.php/Overview

Wilper, C. (2007). Email. Personal communication.

Wood, D. (2007). David wood talks with talis about mulgara and semantic web databases. Interview.
URL http://talk.talis.com/archives/2007/05/david_wood_talk.html, http://talis-podcasts.s3.amazonaws.com/twt20070430-David_Wood.mp3