1. Introduction

Download the PDF version of this Installation Manual by clicking here.

1. Background

All FuGE Database STKs utilize Version 1.0 of the FuGE Standard. Each database toolkit provides a database and persistence layer based on the FuGE Object Model (FuGE-OM), together with other helper classes. These are ideal for community developers wishing to store their FuGE-related data in a more structured way than XML files provide, and as a foundation to build FuGE tools or a FuGE-based system. These databases may also be extended by specific communities to produce community database implementations. There are currently two types of database STK built on Version 1, and a single legacy Milestone 3 STK:

- Hibernate + Spring Database STK
- EJB 3.0 Database STK (not yet available)
- Legacy FuGE Milestone 3, Hibernate + Spring Database STK: Utilizes Milestone 3 of the FuGE Standard.

Now that FuGE Version 1.0 has been released, we suggest that all new developers begin with the Version 1.0 STKs. However, the Milestone 3 STK is retained for legacy reasons, as there may still be some developers utilizing that version.

All current STKs are built with Apache Maven 2, a software project management and build system similar to, but more comprehensive than other build tools such as Apache Ant or GNU Make. They also utilize AndroMDA, a model-driven architecture (MDA) and code generation system, to build a variety of Java classes, configuration files, documents and database scripts. AndroMDA plugs into Maven and during the build process parses a developer's UML Object Model (OM) and generates the appropriate output. It is the core tool used within all FuGE toolkits, and is highly configurable. The type of output can be tailored using different AndroMDA conversion tools, called 'cartridges'. These encapsulated cartridges each do a separate task: for instance, there is an XML Schema cartridge that builds XSDs from UML, and there are Hibernate and Spring or EJB3 cartridges that build a database and persistence layer based on the provided UML.

The Hibernate STK provides:

- a FuGE-structured relational database
- a object-relational persistence and query layer
- a set of Plain Old Java Object (POJO) entity classes representing FuGE UML entities
- a set of Data Access Object (DAO) POJO classes that facilitate and encapsulate access to entity classes

A persistence layer bridges the gap between a relational database and POJOs, abstracting the low-level database code by providing the programmer with an object-oriented interface. Using AndroMDA-generated classes in conjunction with Hibernate + Spring allows the developer to work only with POJOs, and, when ready to populate, query or modify the database, the persistence layer handles the underlying database commands.

2. Purpose of the Hibernate STK and its relation to the greater FuGE Toolkit Project

The FuGE software tools are separated based on their development goals. The FuGE Hibernate Project has been specifically created to aid

- those who wish to be able re-create or modify the FuGE Object Model in database form for their own purposes
- those who wish to use a Hibernate persistence layer to perform database functions

The FuGE Hibernate STK makes use of the jars created by the FuGE XSD STK to perform a number of tasks. Shortly, a Maven 2 repository specifically holding these jars will be available, but until then, please visit the STK pages, and install the FuGE XSD STK prior to starting work with this STK. You will need the jars created by that project.

When you check-out the hibernate-stk subversion repository, you will get a Maven2 project. As provided, this project builds a FuGE database plus a Hibernate and Spring persistence layer. Full instructions for compiling this project and generating and accessing the FuGE database are included in this documentation.

3. What is the relationship between the FuGE Hibernate STK and the FuGE-OM?

In the FuGE development process, the master version of the FuGE structure is the Object Model, written with UML and, using Maven 2 and AndroMDA, translated into a variety of tools. The FuGE XSD STK is available as a separate project, whose purpose is to create either a plain FuGE XSD or aid in the creation of specific community extensions of the FuGE-OM. This Hibernate STK depends upon jars created within the FuGE XSD STK and has been developed for those who wish to work with a FuGE RDBMS with a Hibernate and Spring persistence layer. Additionally, there is a EJB2 STK for FuGE, which uses EJB3 rather than Hibernate and Spring.

The Hibernate STK has, in addition to the auto-generated code for the persistence layer, a set of manually-created mapping code that links the JAXB2 objects present in the FuGE XSD STK to the Hibernate FuGE classes. This allows the user to read FuGE-ML directly into the database.

There are a number of changes made to the officially-sanctioned FuGE-OM as provided in the main download pages. These changes are generally to low-level AndroMDA tags and are required to make sure that access to the database is straightforward and clear. A full list of the changes can be found on the page within this documentation entitled FuGE-OM Modifications for the

Hibernate Implementation .

4. First Steps

If you're interested in FuGE, you can download the FuGE-OM and view it in MagicDraw 15.0 . The Community Edition can be downloaded for free, if you are an academic.

If you wish to develop a database based on FuGE, then this Hibernate Software ToolKit (STK) may be right for you.

2. What You'll Need

1. Recommendations

1.1. Operating Systems

The FuGE Hibernate STK can be installed on either Windows or Linux, but we highly recommend a Linux system, specifically Ubuntu Gutsy Gibbon and higher.

If you use Ubuntu, it is simple to apt-get the components required. Alternatively, you could create a Virtual Machine and install the latest Ubuntu Server (or Standard) release (8.04, Hardy Heron) on it. This has the advantage that it has postgresql 8.3 and maven 2.0.8 in its package repository. During the installation it even asks whether you want to install a database server.

1.2. Database Choice

We also recommend PostgreSQL or MySQL for the database choice, as these are the only two that have been tested with this STK. Of these, PostgreSQL is the preferred choice as it is the one used by the main developer of this STK.

These instructions were originally and loosely based on those found in the Credits Section , but have been modified to suit the needs of this latest version of the STK.

2. Installing SSH

While the STK does NOT require SSH, if your database server will be running on a different machine from where you have this project installed, you may wish to be able to log on to that remote machine.

- Ubuntu. In Ubuntu, you can just run the following command to install ssh if you don't have it yet on the appropriate two machines:

  ```
  sudo apt-get install ssh
  ```

- Other OSs. You may already have ssh installed on the appropriate machines. You can use either commercial SSH or the open alternative, OpenSSH (this is what Ubuntu installs using the above command).

- Windows. There are a couple of options for installing an OpenSSH server on your windows machine, though neither are particularly easy. (This is another reason we recommend running the webapp from a *nix environment.)
    - One option for OpenSSH is to install OpenSSH for Windows , which installs a minimal cygwin package.
    - Fully install cygwin, and then install OpenSSH within cygwin. Might be overkill.

3. Installing Sun Java 5 or Java 6

The FuGE STK has been tested on both Sun Java 5 and Java 6, so feel free to use whichever version suits you.

3.1. Install Sun J2SE Development Kit 6.0 (JDK 6.0)

Otherwise, if you are using Ubuntu 7.04 or higher (or similar), you can install Sun Java 6 simply with an apt-get command. The following commands show the installation of the binaries, and of the JDK (the latter is required for Tomcat):

```
sudo apt-get install sun-java6-bin
sudo apt-get install sun-java6-jdk
```

and then optionally set the default version in your system with

```
sudo update-java-alternatives
```

Full details of the installation of Sun Java on Ubuntu 7.04 or higher is available at https://help.ubuntu.com/community/Java .

Alternatively, you can download the latest Java from Sun directly .

3.2. Install Sun J2SE Development Kit 5.0 (JDK 5.0)

If you are using Ubuntu 7.04 or higher, you can install Sun Java 5 simply with an apt-get command. The following commands show the installation of the binaries, and of the JDK (the latter is required for Tomcat):

```
sudo apt-get install sun-java5-bin
sudo apt-get install sun-java5-jdk
```

and then optionally set the default version in your system with

```
sudo update-java-alternatives
```

Full details of the installation of Sun Java on Ubuntu 7.04 or higher is available at https://help.ubuntu.com/community/Java . These instructions are for java 6, so just replace any instance of the number "6" in the instructions with the number "5".

Alternatively, you can download Java 1.5.x from Sun directly .

3.3. Check Your Installation

Make sure that the JAVA_HOME environment variable is pointing to the directory where you installed the JDK.

If using the "apt-get install" method for Ubuntu described above combined with the update-java-alternatives command, you should already have this variable set.

4. Download and Install Maven 2.0.7 or Later

Download and install Maven 2.0.7 or later from this site: http://maven.apache.org/download.html

5. Maven Setup

Create a directory in your home directory called .m2 with a single file inside called settings.xml . This is what my settings.xml looks like:

```
            <settings>
<localRepository>/media/share/synched/Documents/.m2/repository/</localRepository>

            <proxies>
```

```
<proxy>
<active>true</active>

<protocol>http</protocol>


<host>my.proxy.host</host>
<port>8080</port>
</proxy>

</proxies>
<mirrors>
<mirror>

<id>ibiblio.org</id>

<name>ibiblio Mirror of http://repo1.maven.org/maven2/
</name>

<url>http://mirrors.ibiblio.org/pub/mirrors/maven2</url>


<mirrorOf>central</mirrorOf>
</mirror>

</mirrors>
</settings>
```

All sections are optional. However, depending on your circumstances, you may wish to use one or more of these settings. The "localRepository" element names a location separate from the default home directory for the Maven2 repository. This may be beneficial if you have limited space on your home directory, as this repository directory can grow quite large. The "proxies" element should only be used for those developers who must access the internet via a proxy. The "mirrors" section is useful to have in case the primary central Maven2 server is offline for some reason.

6. Environment Variables and Settings for Maven

Set up the environment variable M2_HOME to point to your maven installation directory, and then ensure that both $M2_HOME/bin and $JAVA_HOME/bin are present in your PATH. Also set M2_REPO, which is the location of your Maven2 repository. Not only is it recommended by Maven, it is also a variable used by this project later on.

7. Test Maven (Part One)

Test that you've instaled maven correctly by running mvn --version . You should see something similar (but not necessarily identical) to the following:

```
$ mvn --version
Maven version: 2.0.7
Java version: 1.5.0_08
OS name: "linux" version: "2.6.17-12-386" arch: "i386"
```

8. Test Maven (Part Two)

Check that Maven2 is working properly by creating a temporary, empty project with the following command:

```
mvn archetype:create -DgroupId=testapp -DartifactId=testapp
```

Check for the BUILD SUCCESSFUL message and, once you have received this message, please delete the created testapp folder.

9. Introduction

This is the only AndroMDA artifact that we will install explicitly. All other artifacts, such as AndroMDA cartridges, will be automatically downloaded by the Maven2 scripts generated by the plugin. Install the plugin by following the steps below.

10. AndroMDA Installation

10.1. Download and Install

- Click here to download the the AndroMDA plugin installer.
- Unzip the contents of the installer into your Maven repository at $M2_REPO (or whatever you have set "localRepository" to be, or by default, it resides in your-home-dir/.m2/repository).
- Verify that the following directory was created (switch the slashes around for Windows):

```
$M2_REPO/org/andromda/maven/plugins/andromda-maven-plugin>>>
```

10.2. Test Installation

- Create a temporary directory, e.g. C:\andromda-temp or $HOME/andromda-temp .
- Create a file called pom.xml in this directory with the following content:

```
<project>
<modelVersion>4.0.0</modelVersion>
<groupId>samples.test</groupId>
<artifactId>test</artifactId>
<version>1.0</version>
<packaging>jar</packaging>
<name>test</name>
<build>
<defaultGoal>compile</defaultGoal>
<plugins>
<plugin>
<groupId>org.andromda.maven.plugins</groupId>
<artifactId>andromdapp-maven-plugin</artifactId>
<version>3.2</version>
</plugin>
</plugins>
</build>
<repositories>
<repository>
```

```xml
<id>andromda</id>
<name>AndroMDA Repository</name>
<url>http://team.andromda.org/maven2</url>
</repository>
</repositories>
<pluginRepositories>
<pluginRepository>
<id>andromda</id>
<name>AndroMDA Repository</name>
<url>http://team.andromda.org/maven2</url>
</pluginRepository>
</pluginRepositories></project>
```

- Open a Command Prompt in the directory where you created this pom.xml and run the command mvn without any arguments. Make sure the command completes successfully by displaying the BUILD SUCCESSFUL message.
- You can now delete the temporary directory you just created.

11. Choosing a UML Tool

You will probably want a UML tool to use if you wish to manipulate the UML at all. If this is important to you, AndroMDA has been tested with both ArgoUML and MagicDraw , though the FuGE STK only recommends the MagicDraw line of editors. You must use MagicDraw 15.0 or higher. A free version of the Community Edition of this UML editor is available.

12. Installing a Supported Database

You must choose a database supported by Hibernate. Databases used in the creation of the Hibernate STK include PostgreSQL and MySQL. Hibernate STK has been extensively tested on PostgreSQL and less extensively tested on MySQL. In fact, many users (especially Windows users) have reported problems in the past with using MySQL together with earlier versions of this project on Windows. In these cases, we recommend PostgreSQL.

12.1. PostgreSQL

You can download PostgreSQL from http://www.postgresql.org/download/ . It is the recommended database . It is not required, but it is recommended that you also install a graphical database browser such as pgAdminIII or PhpPgAdmin, both available from the url above. The JDBC driver should come with the download. Alternatively the PostgreSQL jdbc driver can be downloaded from http://jdbc.postgresql.org/

12.2. MySQL

You can download MySQL from http://dev.mysql.com/downloads/ . phpMyAdmin (http://www.phpmyadmin.net/home_page/downloads.php) is suggested as the GUI for MySQL. If using MySQL, the jdbc driver can be downloaded at: http://dev.mysql.com/downloads/connector/j/3.1.html

Instructions for Subversion checkout are available on the Source Repository information page.

3. Setting Up, Compiling, and Running the Hibernate STK

1. Maven Profiles

The Hibernate STK makes use of two of the profiles generated by AndroMDA within the top-level pom.xml : the default profile and the validation (val) profile. The default profile should contain all of the connection details for your main database, while the validation profile should contain all of the details for your unit testing and general testing database.

In this way, you don't have to constantly be changing the values in your pom.xml when you are switching between a testing environment and your normal environment. This page shows you how to set the values for each of these profiles.

The default profile is identified within the pom.xml with the term "local" and the default activation setting:

```
[...]
  <profiles>
    <profile>
      <id>local</id>
      <activation>
        <activeByDefault>true</activeByDefault>
      </activation>
[...]
```

The testing profile is identified within the pom.xml with the term "validation" (short name "val"):

```
[...]
    <profile>
      <id>validation</id>
      <activation>
        <property>
          <name>env</name>
          <value>val</value>
        </property>
      </activation>
[...]
```

By default, no unit tests will be performed when the default (local) profile is in use. If you plan to run the unit tests, then please make both a default and a test database using the steps below.

If you wish to use any of the other two available profiles, then please do. Currently, the STK makes no use of them.

2. Create a database to hold the metadata

2.1. Default (Local) Profile

Create a database inside your chosen DBMS entitled, for example, "fuge". You will also need to have access to a database user that is the owner of this database, e.g create a new user called "fuge" that you use specifically for accessing and modifying the metadata database. Modify the following elements in the top-level pom.xml located at your-checkout/trunk/pom.xml for the profile with the "local" id. This will change the jdbc connection details to a value suitable for your database connection, and also provide the STK with your database username and password:

```
                <jdbc.url>
                </jdbc.url>

                <jdbc.username></jdbc.username>
                <jdbc.password></jdbc.password>
```

2.2. Testing (Validation) Profile

Follow the same steps as above, but making the changes in the variables within the validation profile rather than the local profile.

3. Point Maven at the correct jdbc jar

3.1. Default (Local) Profile

Modify the following elements in the top-level pom.xml located at your-checkout/trunk/pom.xml for the profile with the "local" id. This will ensure that the STK is pointed at the appropriate location for your jdbc jar and the appropriate class name for your jdbc driver:

```
        <jdbc.driver>
        </jdbc.driver>

        <jdbc.driver.jar>
        </jdbc.driver.jar>
```

The values of these elements will depend on what database type you've chosen to use. For example, the value of jdbc.driver would be org.postgresql.Driver for PostgreSQL, and com.mysql.jdbc.Driver for MySQL.

3.2. Testing (Validation) Profile

Follow the same steps as above, but making the changes in the variables within the validation profile rather than the local profile.

4. Optional Modification of the UML

OPTIONAL: If you wish to make changes to the FuGE UML prior to generating the code, the UML may be found in trunk/fuge-hibernate-mda/src/main/uml/FuGE-v1-profile.mdzip. Any changes you make in the UML must be converted from this MagicDraw proprietary file type to a standard UML2 file type. Within MagicDraw, you will need to choose File -> Export ->"EMF UML2 (v1.x) XMI" and save the resulting files to the same directory that the mdzip resides in. Do not commit these exported files back to Subversion: the master version of the file is the mdzip.

5. Compilation

5.1. Default (Local) Profile

Change into the top-level trunk/ directory. Generate all of the automatically-generated AndroMDA sources:

```
        mvn install
```

You should see a "BUILD SUCCESSFUL" message at the end of it. You must be connected to the internet for this step to work, as there will be many jars that need to be downloaded.

You now have all auto-generated code.

5.2. Testing (Validation) Profile

This is the same as the Default profile, except you have to explicitly state that you want to compile using the Validation profile. Further, during the initial install, you don't want the unit tests to be run, as you want all jars to be made first.

```
mvn -Denv=val -DskipTests install
```

The Validation profile has now been used to make all FuGE Hibernate STK jars.

You can auto-generate the Java IDE project files using Maven 2. The auto-generation step should be performed for the first time after successfully installing and compiling the STK. If you need to update the auto-generated files at any time, you can just re-run the command.

For example, to autogenerate IntelliJ files ( ), change directory to within the top-level trunk directory and run the following command:

```
mvn idea:idea
```

There are other similar commands for IDEs such as Eclipse. Details can be found on the Maven website.

If you are using IntelliJ 7.x or higher, the first time you use the project files, IntelliJ will tell you it needs to upgrade the project files. Just agree to all dialog boxes.

6. Default (Local) Profile

There are many useful AndroMDA-specific maven commands in the trunk/readme.txt generated by AndroMDA, including the commands necessary to create the tables in the database, also shown below. You will need to create the database before using the STK.

```
mvn -f fuge-hibernate-core/pom.xml andromdapp:schema -Dtasks=create
```

If you need to re-create the database from scratch at any time, you can add the drop task to the task list as shown below:

```
            mvn -f fuge-hibernate-core/pom.xml andromdapp:schema -Dtasks=drop,create
```

7. Testing (Validation) Profile

If you wish to create the test database (used in the unit tests), then you will need to run this command after having run the Compiling step for the Validation profile:

```
            mvn -f fuge-hibernate-core/pom.xml -Denv=val andromdapp:schema -
Dtasks=create
```

4. Known Problems

1. The Chicken and the Egg

Within the FuGE-OM, the AccessRight association (visible within the AuditMain class diagram) links the SecurityAccess entity to the OntologyTerm entity with a 0...1 relationship. You will find within the XmlDbRoundtripTest that any reference to the "accessRight" association end is removed before comparing the original XML file and the XML file that was generated from the database information. This is because of a basic chicken-and-the-egg problem between AuditCollection and OntologyCollection.

This attribute of SecurityAccess links to an item in the OntologyCollection. However, the AuditCollection *must* be loaded in the database before we can proceed to parsing the OntologyCollection, as the OntologyCollection may have Audit Trails that are referenced by the AuditCollection. However, if the ontology term that is referenced by the accessRight association is not yet in the database, that association simply cannot be set within the SecurityAccess class.

In normal use, such ontology terms would already be present in the database, however for a unit test round trip which is generating a new random XML file for each unit test, the ontology term is NOT yet in the database. Hence the chicken-and-the-egg situation. Therefore, this association is purposely removed from the XML file prior to running the comparison of the before and after XML files, so that a known bug won't shield an unknown one.

Any suggestions on resolving this situation would be most appreciated.

2. Linking Table Between Software and Equipment is Not Getting Filled

Within the FuGE-OM, the Software entity is linked to the Equipment entity with a Many2Many relationship. For some reason, the spring-controlled Hibernate session is not properly flushing the session after setting up an association of this type. Because of this, the Software2Equipment table within the database, meant to hold such links, is never populated. This behaviour is shown in the GenericSoftwareMappingHelperTest class, which currently fails.

A full explaination of this error is present in the Andromda forums , where I have yet to receive a response. This part of the code will be fixed as soon as I can figure out how to add the appropriate session.flush() statement.

Any suggestions on resolving this situation would be most appreciated.

5. Testing the STK

This STK uses TestNG for its test suite. Maven has an integrated test suite plugin called Surefire, which has Testng support .

There are a small number of unit tests already available with this STK, and more are planned. These tests are currently limited to the fuge-hibernate-mapping sub-project, and can be found within that directory. Tests are disabled by default within the default (local) profile and within the production profile so that production databases are not accidentally modified with test data.

You should run the tests after re-running the Maven install command with the correct profile, thus ensuring that the database connection details are the right ones for the test setup.

To run the tests, simply run this command from the top-level trunk directory:

```
mvn -o -f fuge-hibernate-mapping/pom.xml test -Denv=val
```

All should pass except two (testSoftware2EquipmentLink and testSoftware2EquipmentLinkWithFlush), which are demonstrating a bug discovered within the auto-generated Hibernate and Spring code. Full details of this bug are available in the Andromda forums , and as soon as I have a fix, the unit tests should all pass.

6. Example Java Code Provided in the STK

The mapping between the JAXB2 classes and persistence classes generated by the cartridges must currently be done manually. This STK provides the handwritten mapping code between the JAXB2 classes and the Hibernate cartridge-generated POJOs.

The code that performs this mapping is present within net.sourceforge.symba.mapping.hibernatejaxb2 package inside the fuge-hibernate-mapping sub-project. It is tested with a round trip from the XML to the database and back again.

The DatabaseObjectHelper class contains some convienience methods to help access to the database services. These are described below:

- getOrCreate : Use when you're not sure if your object is already in the database. It will either retrieve your object or create a new one. Checks to see if the identifier provided is already in the database. If it is, return the associated object. If it isn't, then create the object with the provided identifier and return it. If no identifier was provided, then create a fresh identifier and return the new object. This does NOT create any object in the database.
- assignAndSave : Should only be used if you want to re-assign an identifier to an already existing object and then load that object into the database, as it does NOT create a new object but assumes a pre-existing one. You may pass a Person object as the person who should be marked as the audit trail owner. This person must already be loaded in the database, if used. However, the method will deal properly with null values in the person argument, therefore if you don't have person information, just pass a null value for that argument and the audit information will be created without it.

There are various helper classes available to help you load and unload XML into the database. These classes make use of the net.sourceforge.symba.mapping.hibernatejaxb2.helper mapping classes.

1. PeopleUnmarshaler (Example Main in UnmarshalPeople)

Allows you to load the contents of a FuGE-ML AuditCollection into the database. This is useful if you wish to pre-fill the database with the members of an organization or group.

2. XMLUnmarshaler (Example Main in UnmarshalXML and XMLRoundtrip)

Allows you to load the contents of a full FuGE-ML file into the database.

3. XMLMarshaler (Example Main in MarshalXML and XMLRoundtrip)

Allows you to extract a full FuGE experiment from the database and write it out in FugE-ML.

The GenerateOntologyIndividuals class within the net.sourceforge.symba.mapping.hibernatejaxb2 package allows the user to generate quickly a FuGE-ML OntologyCollection file that can then be loaded into the database as part of a FuGE-ML file.

The input file is of the type:

```
accession1::name1
accession2::name2
[...]
accessionN::nameN
```

And you'll get out some FuGE-ML to load into your database. The ontologySource URI and the URN namespace are hard-coded right now, so you will need to modify that to suit your needs.

7. Extending the Hibernate STK

SyMBA is an implementation of Milestone 3 FuGE, which will be upgraded to Version 1 within a couple of weeks of the production of this STK. It extends the FuGE-OM to allow for more complex versioning as well as a LSID implementation and a web interface for users.

8. Would You Like to Contribute?

1. Do you wish to contribute to the FuGE Hibernate STK?

If you have enjoyed using this STK, but also have some ideas for improvement, or perhaps wish to include some of your own code in this project, please begin by contacting fuge-devel@lists.sourceforge.net. Explain what things you would like to see included, and whether or not you would like to help add them.

If appropriate, the FuGE Administrators could add you as a developer, allowing you to commit code back to the FuGE subversion repository. There are some guidelines to follow, however, and please read on to see them.

2. Introduction

This section is for FuGE developers only: that is, those who are identified on the SourceForge project site as developers or administrators. It offers guidelines in modifying and documenting this STK. You will only be able to modify the website or the code in Subversion if you are a FuGE developer.

3. What to include in new Java Classes

If you're contributing new Java classes to the STK, please ensure that you have put in useful javadoc for all methods. If you've created a new package or sub-project, you'll also need to check the top-level pom.xml in the reporting element, and ensure that your source code will get picked up by the javadoc plugin when the mvn site:site command is run.

Also, please copy the license section from one of the existing Java classes and include it in your file.

4. Subversion Best-Practices

You can commit changes if one of the FuGE SourceForge administrators adds you as a FuGE developer. You will need to contact fuge-devel@lists.sourceforge.net if you wish to be added.

- Run "svn update"Every time you start a programming session with the STK, and also just prior to running "svn commit". This will ensure that you resolve any potential conflicts prior to committing changes.
- Please only commit when you have a working STK: do not commit code that will break the build. We need to ensure that anyone who checks out our STK from the trunk/ can at least build the project, even if the trunk is in a snapshot state (which it will normally be in).
- Each time we create a point or full release, it will be tagged and marked in the tags/ subdirectory of the project. This means that users can always choose to stay with a particular release. If we need to bug-fix a particular release separately from what we're already building in the trunk, then we can create a branch for that release in the branches/ subdirectory and work from there. The tags/ subdirectory is never to be committed to, as that would be against Subversion best-practices.

More information on Subversion can be found in its Manual .

5. Beyond Javadoc: Documenting Your Work

If any of the changes you make change how a user would install or use this STK, please ensure that you update the Hibernate documentation, which we make available via the website both in html and pdf format. The actual documentation is built using the APT (Almost Plain Text) Format and then generated by maven into other formats.

The APT files are made into a book using the doxia maven plugin - this is a plugin that, in a more generic usage, also builds the maven site documentation when you run the command mvn site:site . The benefit of using the Doxia book plugin is that it can create the book in multiple versions: currently we build the book in xdoc format, which is used by the mvn site:site command, and in pdf, which is manually copied to the FuGE website for download and offline use. Please see the Doxia Plugin Manual and information on writing in APT format .

You build the new version of the book by going into the books/ sub-directory and running the book-generation command:

```
cd books/
mvn doxia:render-books
```

This builds the book in the books/target/generated-site directory.

6. Re-Building the FuGE Hibernate STK Website

When you are ready to re-build the entire website, go into the top-level directory and generate the site docs in each of the sub-projects:

```
mvn site:site
```

This actually builds all of the html files. However, they are still in their individual sub-directory's target/site directory. To put them all together and copy them to the FuGE website, you need to use the mvn site:deploy command. However, to prevent accidental copying, the shell.sf.net site element is commented out. Please ensure this remains the case when you are committing changes to SVN.

Also edit your $M2_HOME/settings.xml file by adding the following:

```
<server>
<id>shell.sf.net</id>
<username>your sf username</username>
<password>your sf password</password>
</server>
```

mvn site:deploy JUST copies the already-generated site docs. If you make a change to any of the APT files, you must run mvn site:site FIRST, to re-generate the site docs, THEN mvn site:deploy to copy them to the final location described in the distributionManagement element of the pom.xml

It is a good idea to test the maven-generated site before publishing to the fuge website. Change the url element of the local-test site element below to local directory that is right for you, then deploy locally:

```
mvn site:deploy
```

and check the resulting site. If all looks OK, comment-out the local-test site element and un-comment the shell.sf.net site element. Only developers with write-access to the shell.sf.net server and the fuge group area will be able to perform this goal. Run the deploy command again to copy to the real SourceForge webserver:

```
mvn site:deploy
```

The only problem with the site deployment at the moment is that it doesn't copy the non-html books properly. This means that the pdf version of the installation instructions doesn't get put on the server. So, as a final step, you need to scp the pdf manually:

```
    scp books/target/generated-site/pdf/fuge-hibernate/fuge-xsd.pdf  your-
username@shell.sf.net:/home/groups/f/fu/fuge/htdocs/stks/hibernate-stk/
```

You can only run site:deploy successfully if you are down as a FuGE developer on the SF site.

9. Modifications to the FuGE-OM

1. Modifications to the FuGE-OM for the Hibernate and the EJB3 STK

This document contains the changes to the reference FuGE v1 UML Object model required to properly generate the Hibernate code using the AndroMDA Hibernate+Spring cartridge. It is based extensively on Leandro Hermida's similar document for the FuGE EJB3 STK, so that the two UML documents will be similar. Deviations from his modifications are noted.

- Rename the copied package to net.sourceforge.fuge
- Within the net.sourceforge.fuge package change ALL the packages in the entire FuGE hierarchy to lowercase so that it will later produce JavaBeans standard package names (e.g. ConceptualMolecule --> conceptualmolecule)
- (This step was not needed in the Hibernate STK UML Diagram, as it is already fixed there: Fix typos in UML model, where +referenceAbleCollection between FuGE and ReferenceableCollection should read +referenceableCollection)
- (This step was also not needed within the Hibernate FuGE-OM) Search and remove all AbstractAssociation stereotyped associations (20 of 26 instances) EXCEPT the following (which don't seem to have a concrete partner - am still checking with developers about this) * +dimensionElements : net.sourceforge.fuge::bio::data::DimensionElement
    - +dimensions : net.sourceforge.fuge::bio::data::Dimension
    - +inputPartitions : net.sourceforge.fuge::bio::data::DataPartition
    - +outputPartitions : net.sourceforge.fuge::bio::data::DataPartition
    - +partitionPairs : net.sourceforge.fuge::bio::data::PartitionPair
    - +supportingData : net.sourceforge.fuge::bio::data::Data
- Look through all of the associations in the FuGE model and remove any association name numbers that got added during copying (e.g. Provider1 --> Provider)
    - net.sourceforge.fuge.Provider1
    - net.sourceforge.fuge.bio.data.DimensionElementSet1
    - net.sourceforge.fuge.bio.data.PartitionedData1
    - net.sourceforge.fuge.bio.investigation.Conclusion1
    - net.sourceforge.fuge.bio.investigation.Hypothesis1
    - net.sourceforge.fuge.bio.material.Components1
    - net.sourceforge.fuge.bio.material.MeasuredMaterial1
    - net.sourceforge.fuge.common.protocol.Actions1
    - net.sourceforge.fuge.common.protocol.ChildProtocol1
    - net.sourceforge.fuge.common.protocol.Equipment1
    - net.sourceforge.fuge.common.protocol.EquipmentParts1

    - net.sourceforge.fuge.common.protocol.InputCompleteMaterials1
    - net.sourceforge.fuge.common.protocol.ParameterPairs1
    - net.sourceforge.fuge.common.protocol.Parameters1
    - net.sourceforge.fuge.common.protocol.Parameters2
    - net.sourceforge.fuge.common.protocol.Parameters3
    - net.sourceforge.fuge.common.protocol.Parameters4
    - net.sourceforge.fuge.common.protocol.Parameters5
    - net.sourceforge.fuge.common.protocol.Protocol1
    - net.sourceforge.fuge.common.protocol.Software1
    - net.sourceforge.fuge.common.protocol.Software2Equipment1
    - net.sourceforge.fuge.common.protocol.Value1
- All many-to-many associations with two-way navigability MUST have an aggregation (or composition) end
    - (added to end labelled "equipment", though it then appears on the end labelled "software" in the diagram!) Software2Equipment (between GenericSoftware and GenericEquipment) should be "shared" on +equipment end
- Fix database table name clashes between certain associations and classes using @andromda.persistence.table tagged value

Association	Tagged Value	Location	Reason
Equipment	PROTOCOL_EQUIPMENT	between GenericEquipment and GenericProtocol	clashes with Equipment class
Software	PROTOCOL_SOFTWARE	between GenericSoftware and GenericProtocol	clashes with Software class
Provider	PARAMETERIZABLE_PROVIDER	between Parameterizable and ContactRole	clashes with Provider class

- Create custom table names for classes and associations using @andromda.persistence.table tagged value

Class	Tagged Value	Reason

FuGE	F_U_G_E	instead of FU_G_E
Software2Equipment (between GenericSoftware and GenericEquipment)	SOFTWARE2EQUIPMENT	instead of SOFTWARE2_EQUIPMEN T
Database	EXTERNAL_DATABASE	DATABASE is SQL reserved word

- Certain associations need better/corrected table names using @andromda.persistence.table tagged value

Association	Tagged Value	Location
Actions	PROTOCOL_ACTION	between GenericProtocol and Action
ActionApplications	PROTO_APPL_ACTION_APPL	between ProtocolApplication and ActionApplication
Affiliations	AFFILIATION	between Person and Organization
AllBibliographicReferences	REF_COLLECT_BIBLIO_REFERENCE	ReferenceableCollection and BibliographicReference
AllContacts	AUDIT_COLLECT_CONTACT	between AuditCollection and Contact
AllData	DATA_COLLECT_DATA	between DataCollection and Data
AllDatabases	REF_COLLECT_EXT_DATABASE	between ReferenceableCollection and Database
AllDataPartitions	DATA_COLLECT_DATA_PARTITION	between DataCollection and DataPartition
AllDimensions	DATA_COLLECT_DIMENSION	between DataCollection and Dimension
AllEquipment	PROTO_COLLECT_EQUIPMENT	between ProtocolCollection and Equipment

AllProtocolApplications	INVEST_COMPONENT_PROTO_APPL	between InvestigationComponent and ProtocolApplication
AllSequenceAnnotations	CONCEP_MOL_COLLECT_SEQ_ANNOT	between ConceptualMoleculeCollection and SequenceAnnotationSet
AllSoftware	PROTO_COLLECT_SOFTWARE	between ProtocolCollection and Software
Annotations	DESCRIBABLE_ANNOTATION	between Describable and OntologyTerm
BibliographicReferences	IDENT_BIBLIO_REFERENCE	Identifiable and BibliographicReference
Characteristics	MATERIAL_CHARACTERISTIC	between Material and OntologyTerm
Components	MATERIAL_COMPONENT	between GenericMaterial and GenericMaterial
ComponentDesignTypes	COMPONENT_DESIGN_TYPE	between InvestigationComponent and OntologyTerm
ConceptualMolecules	CONCEP_MOL_COLLECT_CONCEP_MOL	betweeen ConceptualMoleculeCollection and ConceptualMolecule
Contacts	MATERIAL_CONTACT	between Material and ContactRole
DatabaseReferences	IDENT_DB_REFERENCE	between Identifiable and DatabaseReference

DataPartitions	FACTOR_VALUE_DATA_PARTITION	between FactorValue and DataPartition
Descriptions	DESCRIBABLE_DESCRIPTION	between Describable and Description
Dimensions	DATA_DIMENSION	between Data and Dimension
DimensionElements	DIMENSION_DIMENSION_ELEMENT	between Dimension and DimensionElement
EquipmentApplications	PROTO_APPL_EQUIP_APPL	between ProtocolApplication and EquipmentApplication
EquipmentParts	EQUIPMENT_PART	between GenericEquipment and GenericEquipment
Factors	INVEST_COMPONENT_FACTOR	between InvestigationComponent and Factor
FactorCollection	INVEST_COLLECT_FACTOR	between InvestigationCollection and Factor
FactorValues	FACTOR_FACTOR_VALUE	between Factor and FactorValue
HigherLevelAnalyses	DATA_COLLECT_HIGHER_LEV_ANAL	between DataCollection and HigherLevelAnalysis
InputCompleteMaterials	INPUT_COMPLETE_MATERIAL	between GenericProtocolApplication and Material

InputMaterials	INPUT_MATERIAL	between GenericProtocolApplication and GenericMaterialMeasurement
InputPartitions	INPUT_PARTITION	between PartitionPair and DataPartition
InputTypes	INPUT_TYPE	between Protocol and OntologyTerm
Investigations	INVEST_COLLECT_INVESTIGATION	between InvestigationCollection and Investigation
InvestigationComponents	INVEST_INVEST_COMPONENT	between Investigation and InvestigationComponent
InvestigationTypes	INVESTIGATION_TYPE	between Investigation and OntologyTerm
Materials	MATERIAL_COLLECT_MATERIAL	between MaterialCollection and Material
Members	SECURITY_GROUP_MEMBER	between SecurityGroup and Contact
OntologySources	ONTOLOGY_COLLECT_SOURCE	between OntologyCollection and OntologySource
OntologyTerms	ONTOLOGY_COLLECT_TERM	between OntologyCollection and OntologyTerm

OutputMaterials	OUTPUT_MATERIAL	between GenericProtocolApplication and Material
OutputPartitions	OUTPUT_PARTITION	between PartitionPair and DataPartition
OutputTypes	OUTPUT_TYPE	between Protocol and OntologyTerm
Owners	SECURITY_OWNER	between Security and Contact
Parameters	EQUIPMENT_PARAMETER	between GenericEquipment and GenericParameter
Parameters	SOFTWARE_PARAMETER	between GenericSoftware and GenericParameter
Parameters	PROTOCOL_PARAMETER	between GenericProtocol and GenericParameter
Parameters	ACTION_PARAMETER	between GenericAction and GenericParameter
ParameterPairs	ACTION_PARAMETER_PAIR	between GenericAction and ParameterPair
ParameterValues	PARAM_APPL_PARAM_VALUE	between ParameterizableApplication and ParameterValue
PartitionPairs	PROTO_APPL_PARTITION_PAIR	between ProtocolApplication and PartitionPair

Performers	PROTO_APPL_PERFORMER	between ProtocolApplication and ContactRole
Properties	ONTOLOGY_INDIVIDUAL_PROPERTY	between OntologyIndividual and OntologyProperty
PropertySets	DESCRIBABLE_PROPERTY_SET	between Describable and NameValueType
Protocols	PROTO_COLLECT_PROTOCOL	between ProtocolCollection and Protocol
ProtocolApplications	PROTO_COLLECT_PROTO_APPL	between ProtocolCollection and ProtocolApplication
Providers	INVESTIGATION_PROVIDER	between Investigation and ContactRole
QualityControlStatistics	QUALITY_CONTROL_STATISTIC	between Material and OntologyTerm
RangeDescriptors	RANGE_DESCRIPTOR	between Range and OntologyTerm
SecurityGroups	AUDIT_COLLECT_SECURITY_GROUP	between AuditCollection and SecurityGroup
SecurityRights	SECURITY_RIGHT	between Security and SecurityAccess
SoftwareApplications	PROTO_APPL_SOFTWARE_APPL	between ProtocolApplication and SoftwareApplication

SourceMaterials	SOURCE_MATERIAL	between Investigation and Material
SummaryResults	SUMMARY_RESULT	between Investigation and HigherLevelAnalysis
Types	PARAMETERIZABLE_TYPE	between Parameterizable and OntologyTerm
Types	SEQUENCE_ANNOTATION_SET_TYPE	between SequenceAnnotationSet and OntologyTerm

- Certain associations ends need better/corrected column names using @andromda.persistence.column tagged value

Association End	Tagged Value	Location	Reason
+parameters	PARAMETER_ID	Parameters association between GenericAction and GenericParameter	override plural association end name
+parameterPairs	PARAMETER_PAIR_ID	ParameterPairs association between GenericAction and ParameterPair	override plural association end name
+affiliations	AFFILIATION_ID	Affliliations association between Person and Organization	override plural association end name
+allContacts	CONTACT_ID	AllContacts association between AuditCollection and Contact	override plural association end name
+securityGroups	SECURITY_GROUP_ID	SecurityGroups association between AuditCollection and SecurityGroup	override plural association end name
+componentDesignTypes	COMPONENT_DESIGN_TYPE_ID	ComponentDesignTypes association between InvestigationComponent and OntologyTerm	override plural association end name

+conceptualMolecules	CONCEPTUAL_MOLECULE_ID	ConceptualMolecules association between ConceptualMoleculeCollection and ConceptualMolecule	override plural association end name
+allSequenceAnnotations	SEQUENCE_ANNOTATION_ID	AllSequenceAnnotations association between ConceptualMoleculeCollection and SequenceAnnotationSet	override plural association end name
+allData	DATA_ID	AllData association between DataCollection and Data	override plural association end name
+allDataPartitions	DATA_PARTITION_ID	AllDataPartitions association between DataCollection and DataPartition	override plural association end name
+allDimensions	DIMENSION_ID	AllDimensions association between DataCollection and Dimension	override plural association end name
+higherLevelAnalyses	HIGHER_LEVEL_ANALYSIS_ID	HigherLevelAnalyses association between DataCollection and HigherLevelAnalysis	override plural association end name

+dimensions	DIMENSION_ID	Dimensions association between Data and Dimension	override plural association end name
+annotations	ANNOTATION_ID	Annotations association between Describable and OntologyTerm	override plural association end name
+descriptions	DESCRIPTION_ID	Descriptions association between Describable and Description	override plural association end name
+propertySets	PROPERTY_SET_ID	PropertySets assocation between Describable and NameValueType	override plural association end name
+dimensionElements	DIMENSION_ELEMENT_ID	DimensionElements association between Dimension and DimensionElement	override plural association end name
+parameters	PARAMETER_ID	Parameters association between GenericEquipment and GenericParameter	override plural association end name
+equipmentParts	EQUIPMENT_PART_ID	EquipmentParts association between GenericEquipment and GenericEquipment	override plural association end name

+factorCollection	FACTOR_ID	FactorCollection association between InvestigationCollection and Factor	override plural association end name
+factorValues	FACTOR_VALUE_ID	FactorValues association between Factor and FactorValue	override plural association end name
+dataPartitions	DATA_PARTITION_ID	DataPartitions association between FactorValue and DataPartition	override plural association end name
+bibliographicReferences	BIBLIOGRAPHIC_REFERENCE_ID	BibliographicReferences association between Identifiable and BibliographicReference	override plural association end name
+databaseReferences	DATABASE_REFERENCE_ID	DatabaseReferences association between Identifiable and DatabaseReference	override plural association end name
+inputCompleteMaterials	INPUT_COMPLETE_MATERIAL_ID	InputCompleteMaterials association between GenericProtocolApplication and Material	override plural association end name

+inputMaterials	INPUT_MATERIAL_ID	InputMaterials association between GenericProtocolApplication and GenericMaterialMeasurement	override plural association end name
+inputPartitions	INPUT_PARTITION_ID	InputPartitions association between ParitionPair and DataPartition	override plural association end name
+inputTypes	INPUT_TYPE_ID	InputTypes association between Protocol and OntologyTerm	override plural association end name
+providers	PROVIDER_ID	Providers assoication between Investigation and ContactRole	override plural association end name
+investigationTypes	INVESTIGATION_TYPE_ID	InvestigationTypes association between Investigation and OntologyTerm	override plural association end name
+investigations	INVESTIGATION_ID	Investigations association between InvestigationCollection and Investigations	override plural association end name

+factors	FACTOR_ID	Factors association between InvestigationComponent and Factor	override plural association end name
+allProtocolApplications	PROTOCOL_APPLICATION_ID	AllProtocolApplications association between InvestigationComponent and ProtocolApplication	override plural association end name
+investigationComponents	INVESTIGATION_COMPONENT_ID	InvestigationComponents association between Investigation and InvestigationComponent	override plural association end name
+characteristics	CHARACTERISTIC_ID	Characteristics association between Material and OntologyTerm	override plural association end name
+materials	MATERIAL_ID	Materials association between MaterialCollection and Material	override plural association end name
+components	COMPONENT_ID	Components association between GenericMaterial and GenericMaterial	override plural association end name
+contacts	CONTACT_ID	Contacts association between Material and ContactRole	override plural association end name

+sources	SOURCE_ID	Sources association between OntologyCollection and OntologySource	override plural association end name
+ontologyTerms	ONTOLOGY_TERM_ID	OntologyTerms association between OntologyCollection and OntologyTerm	override plural association end name
+properties	PROPERTY_ID	Properties association between OntologyIndividual and OntologyProperty	override plural association end name
+outputMaterials	OUTPUT_MATERIAL_ID	OutputMaterials association between GenericProtocolApplication and Material	override plural association end name
+outputPartitions	OUTPUT_PARTITION_ID	OutputPartitions association between PartitionPair and DataPartition	override plural association end name
+outputTypes	OUTPUT_TYPE_ID	OutputTypes association between Protocol and OntologyTerm	override plural association end name

+types	TYPE_ID	Types association between Parameterizable and OntologyTerm	override plural association end name
+parameterValues	PARAMETER_VALUE_ID	ParameterValues association between ParameterizableApplication and ParameterValue	override plural association end name
+actions	ACTION_ID	Actions association between GenericProtocol and Action	override plural association end name
+parameters	PARAMETER_ID	Parameters association between GenericProtocol and GenericParameter	override plural association end name
+actionApplications	ACTION_APPLICATION_ID	ActionApplications between ProtocolApplication and ActionApplication	override plural association end name
+equipmentApplications	EQUIPMENT_APPLICATION_ID	EquipmentApplications between ProtocolApplication and EquipmentApplication	override plural association end name

+partitionPairs	PARTITION_PAIR_ID	PartitionPairs association between ProtocolApplication and PartitionPair	override plural association end name
+performers	PERFORMER_ID	Performers association between ProtocolApplication and ContactRole	override plural association end name
+softwareApplications	SOFTWARE_APPLICATION_ID	SoftwareApplications association between ProtocolApplication and SoftwareApplications	override plural association end name
+allEquipment	EQUIPMENT_ID	AllEquipment association between ProtocolCollection and Equipment	override plural association end name
+protocols	PROTOCOL_ID	Protocols association between ProtocolCollection and Protocol	override plural association end name
+protocolApplications	PROTOCOL_APPLICATION_ID	ProtocolApplications association between ProtocolCollection and ProtocolApplications	override plural association end name

+allSoftwares	SOFTWARE_ID	AllSoftwares association between ProtcolCollection and Software	override plural association end name
+qualityControlStatistics	QUALITY_CONTROL_STATISTIC_ID	QualityControlStatistics association between Material and OntologyTerm	override plural association end name
+rangeDescriptors	RANGE_DESCRIPTOR_ID	RangeDescriptor association between Range and OntologyTerm	override plural association end name
+allBibliographicReferences	BIBLIOGRAPHIC_REFERENCE_ID	AllBibliographicReferences association between ReferenceableCollection and BibliographicReference	override plural association end name
+allDatabases	DATABASE_ID	AllDatabases association between ReferenceableCollection and Database	override plural association end name
+members	MEMBER_ID	Members association between SecurityGroup and Contact	override plural association end name
+owners	OWNER_ID	Owners association between Security and Contact	override plural association end name

+securityRights	SECURITY_RIGHT_ID	SecurityRights association between Security and SecurityAccess	override plural association end name
+types	TYPE_ID	Types association between SequenceAnnotationSet and OntologyTerm	override plural association end name
+parameters	PARAMETER_ID	Parameters association between GenericSoftware and GenericParameter	override plural association end name
+sourceMaterials	SOURCE_MATERIAL_ID	SourceMaterials association between Investigation and Material	override plural association end name
+summaryResults	SUMMARY_RESULT_ID	SummaryResults association between Investigation and HigherLevelAnalysis	override plural association end name

- datatype::Object[] within GenericInternalData.storage has been changed to datatype::Blob. Some mapping of this type is required to temporarily fix the inability of PostgreSQL to deal with Object arrays, as there is no default mapping.

2. Hibernate-only FuGE-OM Modifications

These modifications were only performed on the Hibernate STK's version of the FuGE-OM.

2.1. Changes to the standard PostgreSQL Mapping

There is a file, PostgreSQLExtension.xml, within fuge-hibernate/src/main/uml/config/mappings, which specifies any additional mappings between UML data types and Postgres types. Below are a summary of the contents of the file:

- datatype::Date is mapped to TIMESTAMP WITH TIME ZONE\ I originally tried to map datatype::URI to CHARACTER VARYING(1024) (this datatype::URI mapping has been created to force URIs to be stored as a string, as there is no default mapping). However, this didn't work as it was still being passed as a java.net.URI and ultimately caused exceptions that looked a bit like: "org.springframework.orm.hibernate3.HibernateSystemException: could not deserialize; nested exception is org.hibernate.type.SerializationException: could not deserialize at net.sourceforge.fuge.service.EntityServiceBase.getIdentifiable(EntityServiceBase.java:1113)". Therefore I changed both the PostgreSQL and Java mapping so that both in the database and in Java, the datatype::URI is mapped to CHARACTER VARYING(1024) (for PostgreSQL) and java.lang.String (for Java). This isn't ideal, but until I can figure out a way to have the URI dealt with correctly, at least it doesn't cause problems. This meant that the following URIs are actually strings within the API, though the UML has not changed:
    - net.sourceforge.fuge.common.references.Database.URI
    - net.sourceforge.fuge.common.description.Uri.location
    - net.sourceforge.fuge.bio.data.ExternalData.location
    - net.sourceforge.fuge.common.ontology.OntologySource.ontologyURI

2.2. Changes to column attributes

- Had to set the Unique stereotype for Identifiable.identifier, as this hadn't been done, and without it we cannot retrieve objects based on their identifier.

2.3. Changes to entity names

- The "URI" entity is problematic for Spring, as any class name whose second letter is upper case can cause problems. The best way to solve it is to change the name of this entity to "Uri".

2.4. Changes to entity column/property names

- Certain property names of entities needed to be changed because they were PostgreSQL reserved words. This can be done with the @andromda.persistence.column tagged value.

Property/Column Name	Tagged Value	Comments
+end (hibernate implementation only)	END_DATE	It is the +end property of the Investigation entity
+end (hibernate implementation only)	END_POSITION	It is the +end property of the Sequence entity

2.5. Changes from lazy to eager loading

- The association between AuditCollection and SecurityGroup is composition, but it is still lazily fetched. This is perhaps because it is a 0...* association, and can therefore can have more than one security group? In any case, have set UML to be eager fetched. This means setting @andromda.hibernate.lazy to false within the tag for the association end "securityGroups".
- In the same way, the following association ends were set to @andromda.hibernate.lazy to false:
    - association end (AE): "securityGroups" (the first instance above)
    - AE: "securityCollection"
    - AE: "securityRights"
    - AE: "annotations" for Describable
    - AE: bibliographicReferences for Identifiable
    - AE: parent for Organization
    - AE: affiliations for Person
    - AE: members for SecurityGroups
    - AE: outputMaterials, inputData, inputCompleteMaterials, outputData for GenericProtocolApplication
    - AE: performer for Audit
    - AE: equipment for GenericSoftware
    - AE: equipment for GenericProtocol
    - AE: software for GenericProtocol

- AE: inputTypes, outputTypes for Protocol
- AE: contact for ContactRole
- AE: ontologySource for OntologyTerm

2.6. Change to Cardinalities

The back cardinality (the side with the diamond in it) of GenericAction to Protocol (named childProtocol) should be 0...* rather than 0...1 . It doesn't affect the XSD because these aren't checked but will affect the database. This is something that probably should be fixed in the UML.

10. Further Reading

1. Contributors to Documentation

Special thanks goes to Frank Gibson Rainer Scoepf, and Joerg Servos for helping find bugs in and suggesting additions to the installation docs.

2. Resources used in the creation of this document

1

. A large part of this documentation was originally developed for the SyMBA project. The developers of this project have kindly allowed the transfer of much of the documentation across from SyMBA to the FuGE Version 1 STK.

2

http://wiki.ficcs.org/ficcs/FuGE-to-XSD , accessed 6 August 2007.

3

http://galaxy.andromda.org/index.php?option=com_content&amp;task=view&amp;id=105&amp;Itemid=89 , accessed 6 August 2007.