Transformation of Protégé Ontologies into the Eclipse Modeling Framework
A Practical Use Case based on the Foundational Model of Anatomy

Deepak K. Sharma, Thomas M. Johnson, Harold R. Solbrig, Dr. Christopher G. Chute MD, DrPH
Mayo Clinic College of Medicine, Division of Biomedical Informatics, Rochester, MN

Introduction

The Eclipse Modeling Framework (EMF) "is a modeling framework and code generation facility for building tools and other applications based on a structured data model. From a model specification described in XMI, EMF provides tools and runtime support to produce a set of Java classes for the model, a set of adapter classes that enable viewing and command-based editing of the model, and a basic editor."[1] In addition to providing a jump-start to a rich collection of Eclipse UI widgets, EMF provides a model-centric bridge between Java, UML, XML Schema and a variety of secondary storage mechanisms.

This presentation describes how we combined EMF with the native Protégé API and used this combination transform the information contained a sophisticated native Protégé Ontology into a form that could be used on the Mayo Lexical Grid.

The Problem

The Foundational Model of Anatomy (FMA)[2] is a computer based knowledge source of anatomical information developed and maintained by University of Washington (UW). It currently contains nearly 70,000 concepts (more than 110,000 terms) that represent anatomical entities from cells, tissues, organs, and body parts including the entire body. The FMA contains anatomical concepts and relationships necessary to model the structure of the entire human body.

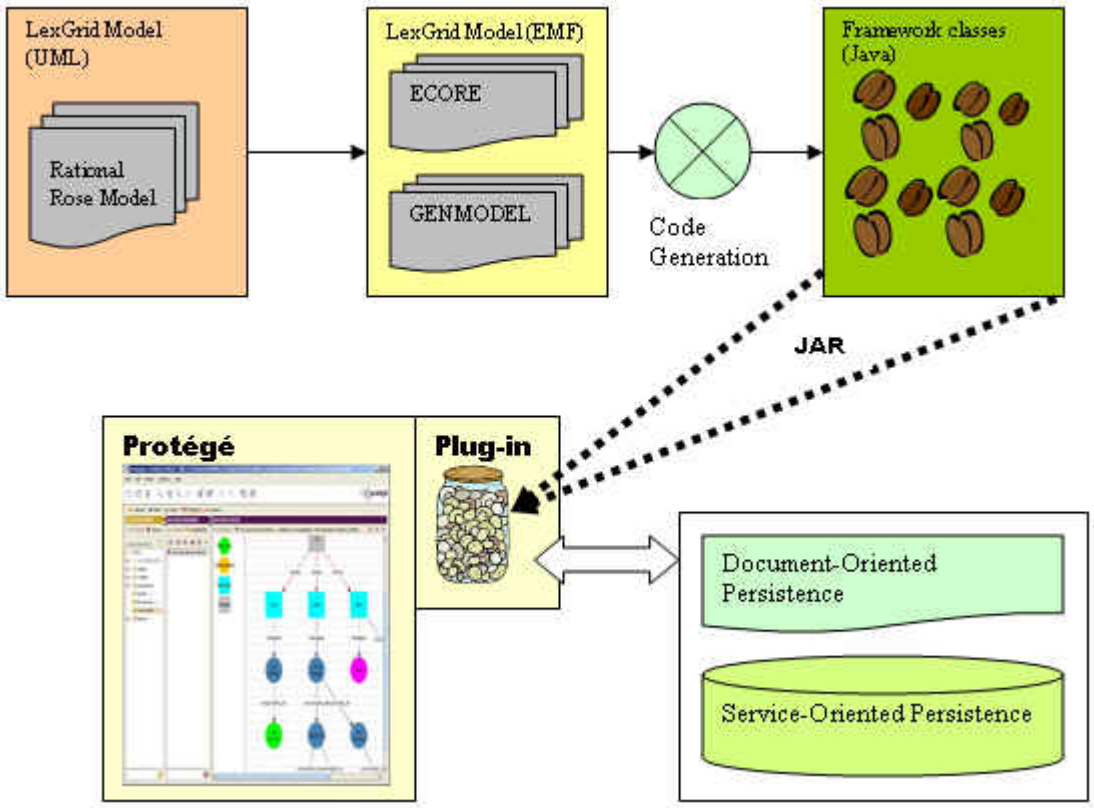
The FMA is currently represented in native (vs. OWL) Protégé. While Protégé provides an ideal development environment, it is difficult to utilize the resultant ontology in its native form. The user is presented with a choice of writing software that uses the Protégé API to access the content in its native form or developing an external model that can contain at least a subset of the rich FMA content in a form that renders it useful to existing software.

The Lexical Grid (or LexGrid)[3] project at the Mayo Clinic's Division of Biomedical Informatics was created to address exactly this sort of integration problem. LexGrid is built using an open model that is capable of representing a broad spectrum of terminological content, from simple code/phrase lists to complex OWL-based ontologies in a consistent, interoperable way. A suite of open-source tools has evolved around the LexGrid model that allows terminologies to be imported and exported, queried and edited in a variety of useful fashions. The Lexical Grid tool suite includes an open source implementation of both the ANSI/HL7 Common Terminology Services (CTS) and the OMG Terminology Query Services (TQS) interfaces.

The architecture of the Lexical Grid was well-suited to externalizing the FMA content. The core of the Lexical Grid[3] is built on a Model Driven Architecture (MDA) core and implemented using EMF. We decided to approach the export task by combining the EMF software and the native Protégé interface. This was accomplished by creating a semantic mapping between the constructs in the FMA model and corresponding constructs in the LexGrid EMF representation.

Combining EMF and the native Protégé API

LexGrid includes a common model for terminology representation which draws from communities engaged in description logic ontologies, widely-used clinical terminologies, and the heritage of terminology services to provide an abstraction capable of supporting a wide variety of disparate terminologies. This model in turn simplifies and clarifies the design and functionality of related tools and facilitates data inter-operability. The LexGrid model is mastered as XML Schema (XSD), with equivalent conversions for meta-models such as the Unified Modeling Language (UML) facilitated by the hyperModel[4] tool. Model representations are also available for select back-end repositories (e.g. relational database schema).



Eclipse Modeling Framework with Protégé

Generation of Java classes was performed using automated tooling provided by the Eclipse Modeling Framework Software Development Kit (SDK). This process consisted of two major phases. First, the LexGrid UML representation was imported to the EMF canonical representation (.ecore and .genmodel files, based on the OMG MOF 1.4 meta-model). The Java class files (interface, implementation, and utility) were then generated from the intermediate Ecore representation using menu actions also provided by the EMF SDK. Minor changes were made to the Ecore and genmodel classes prior to generation in order to fine tune code organization (e.g. package naming) and function (e.g. adjust attribute range or cardinality not carried forward from the original model).

The LexGrid EMF implementation supports two mechanisms of persisting information, referred to here as *Service-Oriented Persistence* and *Document-Oriented Persistence*. Service-Oriented Persistence describes the process by which Java objects are stored and retrieved from a repository through an architected API, typically without intermediate conversion to a flat file format. Support for service-oriented persistence is provided via an open-source persistence framework developed in conjunction with the LexGrid project; supported targets include relational database and Lightweight Directory Access Protocol (LDAP) repositories. Note that the generated EMF implementation was customized to support interfaces defined by this persistence framework. However, customizations are annotated according to EMF guidelines so as not to interfere with any subsequent re-generation from the Ecore models.

In contrast, Document-Oriented Persistence implies conversion of objects to one or more flat files (documents). Resulting files are typically stored in a standard PC file system. The application is typically aware of both the format and location of the document. This support is provided by the EMF framework itself; target representation is standard XML that is compliant with the published LexGrid XML schema.

The LexGrid Model EMF framework is integrated with Protégé as part of the Protégé plug-in that we wrote. This FMA transformation works as a great example to show the power and simplicity of using EMF framework and how easily it can be integrated with Protégé to work with any terminology already exist in Protégé. Once the FMA contents are transformed into this model, then it works as a hub and we can go in any direction we want – getting it exported in XML format, or storing it in SQL DB.

Mapping the FMA Semantics into the LexGrid model

Once the LexGrid EMF classes were incorporated into the Protégé workspace, we still faced the not-insignificant task of mapping the FMA model as it was represented in Protégé into their standard equivalents in the LexGrid model. FMA defines a large number of instances to represent relationships and terms. So we had to carefully look at them in order to perform categorization.

A Concept in the LexGrid Model supports attributes like definitions, comments, presentations and properties. A relation in the LexGrid model establishes an association between two local (or remote) concepts, or between a concept and a data value.

In FMA all concepts are instances of an entry called "Concept Name". Concepts are instances of Meta concepts for dimensional and anatomical categories.

Some generalizations were employed, after carefully studying FMA contents in Protégé project:

1. Treat all concepts and instances in FMA as concepts in the LexGrid Model (except the root concept named "Concept name").
2. List all slots and categorize them as presentations, properties or associations. If the slot type is a Protégé class or instance then it is considered an association, otherwise it is treated as a property or presentation depending on its name.
3. Anonymous instances with names like "fm_live_xxxxx" represent composite relationships (based on other simple relationships, or establishing relationships among more than 2 concepts).
4. Create all the concepts first in the LexGrid Model and assign properties as appropriate based on the Protégé slot entries (depending on its type as explained in step 2).
5. Traverse all instances of the FMA Concept "Concept name" and establish relationships depending on what those instances contain.

Results:

We were able to faithfully represent the FMA content in the LexGrid format. With better understanding of the FMA model, we were able to resolve some minor issues like multilingual representation of a term and identifying forward-reverse association names.

Conclusion and discussion:

The approach appears to work. We have demonstrated that the Eclipse Modeling Framework can be successfully integrated with Protégé, which, we believe has the potential to open a lot of doors. In addition, we have demonstrated that it is possible to take a rich, custom built ontology model from Protégé and transform it into a form that can be used and integrated with the larger terminology environment. Unfortunately, while this second task may be streamlined significantly, it will still need to be repeated for each new ontology model that emerges. There will be a separate workshop at the conference that will focus on alternative solutions to his second issue.

---

[1] http://www.eclipse.org/emf. Last viewed 2005-03-07

[2] http://sig.biostr.washington.edu/projects/fm/. Last viewed 2005-03-07

[3] http://informatics.mayo.edu/ Last viewed 2005-03-08

[4] http://www.xmlmodeling.com/hyperModel/ Last viewed 2005-03-11