19 May 2006

Epsilon - Project plan for a Eclipse GMT subproject

Scope of the Epsilon subproject

The Epsilon subproject aims at building a framework for supporting the construction of domain-specific languages and tools for model management tasks, i.e., model merging, model comparison, inter- and intra-model consistency checking, text generation, etc. It will provide a metamodel-agnostic approach that supports model management tasks for any kind of metamodel and model instances.

The Epsilon subproject will provide the basis for building domain-specific model management languages and tools:

- an executable language for model navigation and comparison, called the Epsilon Object Language
- a virtual machine for execution of models
- directions and documentation to explain how to use the language and virtual machine for building task-specific languages.

The subproject will particularly support the third bullet point by providing a number of task-specific model management languages and tools as guidance, including:

- Epsilon Merging Language (EML): for model merging, i.e., integrating two or more models into a consistent, coherent single model in arbitrary metamodels;
- Epsilon Transformation Language (ETL): for transforming models, used within the EML for handling concepts that do not appear in all languages;
- Epsilon Generation Language (EGL): for generating text from models.
- Epsilon MOF Action Semantics (EAS): an implementation of an action semantics for MOF 2.0.

The Epsilon project will focus on supporting those aspects needed in the context of general model management, including:

- A metamodel-independent mechanism for navigating models and identifying model elements of interest. This mechanism will be executable and declarative, and where possible will be closely aligned with the Object Constraint Language (OCL) standard.
- A metamodel-independent means for comparing models and model elements, in order to help support update and deletion tasks for model management.
- A metamodel-independent mechanism for creating and reading models.
- Fundamental operations for loading and serialising models.
- Ease of use: An easy-to-use interface

Epsilon code availability / initial contribution

At this time, code has been developed that supports parsing, checking, editing, and execution of the Epsilon Object Language, which supports model navigation and comparison. Currently, these are deployed as separate Eclipse plug-in components, which run within Eclipse. The parser and run-time may also be used independently of Eclipse.

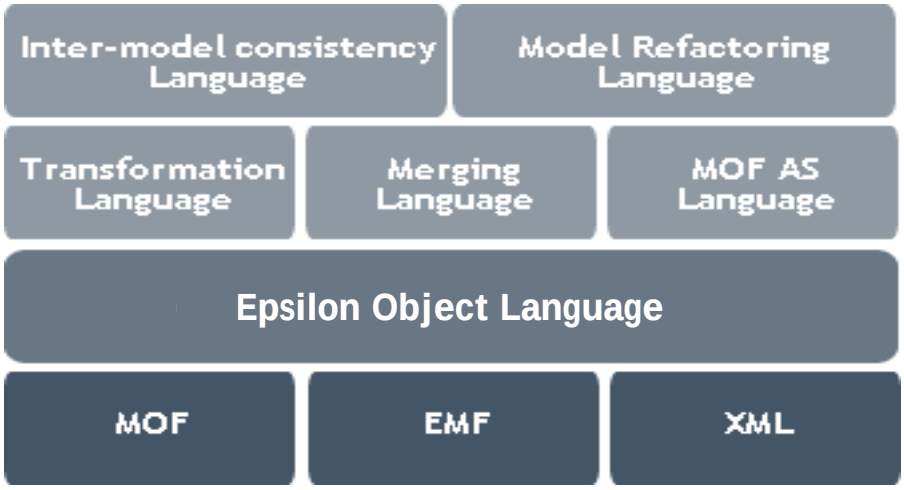
The following code is ready for submission to an Eclipse Epsilon subproject:

- A parser and virtual machine for the model navigation and comparison language (the Epsilon Object Language).
- The Epsilon Merging Language, for specifying merging rules, along with a virtual machine for executing rules.
- The Epsilon Transformation Language, for specifying transformation rules, along with a virtual machine for executing rules.

An initial user guide and examples will also be made available. The intent is to supplement this with additional forms of tutorial information (e.g., a Flash tutorial) when appropriate.

Epsilon Architecture Overview

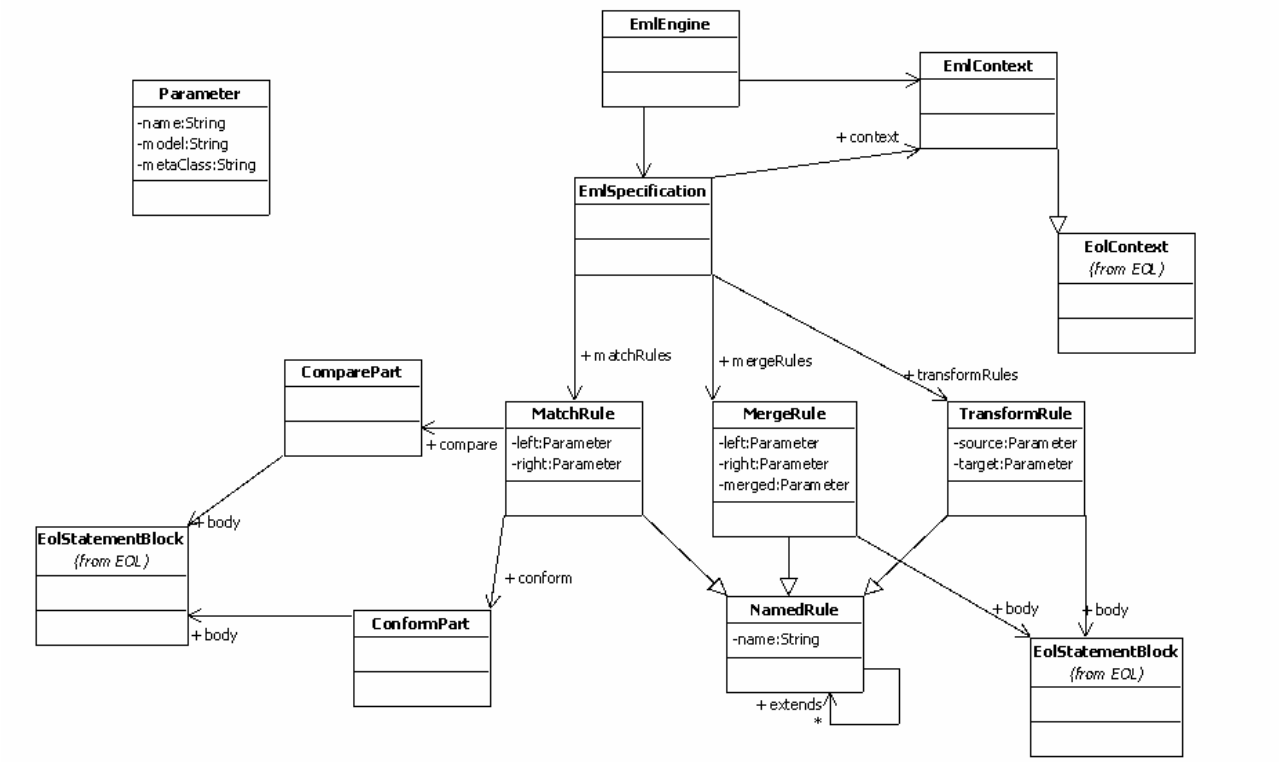
The informal architecture of the Epsilon platform is depicted below.



The core of the platform is the Epsilon Object Language (EOL) which provides a metamodel agnostic layer for navigating models, querying models, and modifying models. EOL has some similarities to OCL but does not completely conform to the OCL 2.0 standard. New, domain-specific languages (e.g., ETL, EML, a MOF 2.0 Action Semantics language, ...) are defined on top of EOL and reuse its facilities for navigating and querying models.

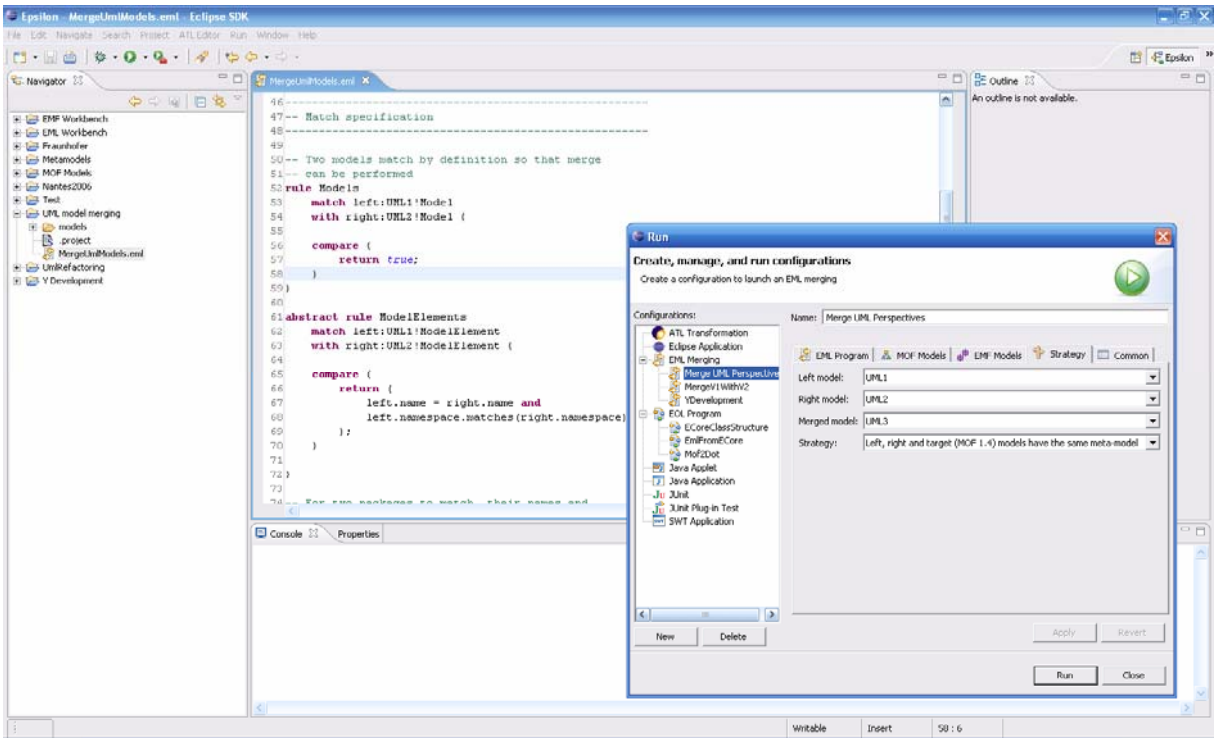
Initial support has been provided for MOF models, EMF models, and XML models, but additional models will be investigated.

The Epsilon Merging Language (EML) is a recent addition to the Epsilon platform. The overall design of the EML language, and its tool support, is illustrated below.



The EML engine applies to an EML specification, defined in terms of a number of rules. These rules make use of EOL for comparison and model navigation. The context in which rules are executed is also supplied via EOL.

A screenshot of the EML design tool is given below.



The Epsilon binaries are available for download; please see the descriptions at http://www.cs.york.ac.uk/~dkolovos/epsilon

Epsilon community

At this time, Epsilon is being developed in a community at the University of York, UK, supported and tested by the European Integrated Project MODELWARE (IST Project 511731). We will continue development of Epsilon in this context.

The MODELWARE project, and other successor projects will provide the necessary time and resources to ensure a continuity of development in the subproject.

We will encourage user communities to use the technology and contribute to improvement through feedback. We will also encourage user and external developers to contribute examples, reusable transformations, and code.

We aim to involve users and developers through publicity in publications, conferences, and presentations. We are also using Epsilon in lecturing at the University of York in courses in model-driven development and language design, and will make our teaching material available to lecturers and instructors worldwide.