6 October 2006

Epsilon - Project plan for an Eclipse GMT component

Scope of the Epsilon component

The Epsilon component aims at building a framework for supporting the construction of integrated 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 component will provide the basis for building integrated domain-specific model management languages and tools:

- an executable language for model navigation and modification, called the Epsilon Object Language (EOL)
- an extensible virtual machine for execution of EOL programs.
- directions and documentation to explain how to use the language and virtual machine for building integrated task-specific languages.
- a suite of integrated, task-specific languages, with tool support, for model management. These language and tools are all based on EOL and share common facilities and tools, and currently consist of: the Epsilon Merging Language (EML) for model merging, i.e., integrating two or more models of arbitrary metamodels into a consistent, coherent single model; Epsilon Transformation Language (ETL) for transforming models (re-used in EML); Epsilon Comparison Language (ECL), for comparing models of arbitrary metamodels (re-used in EML).

We emphasise that the Epsilon component consists of not only the core language, EOL, and its tools, but also the integrated task-specific languages, which support each other. For example, the transformation and comparison languages are both used within EML.

The Epsilon component 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 modifying 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 modification. Currently, these are deployed as separate plug-in components, which run within Eclipse. The parser and run-time may also be used independently of Eclipse.

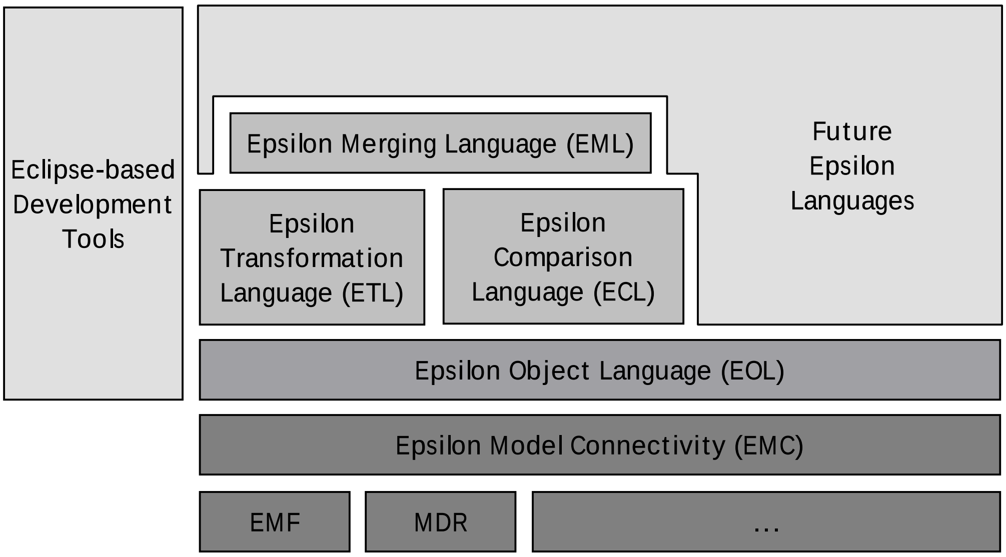
The following code is ready for submission to an Eclipse Epsilon component:

- A parser and virtual machine for the model navigation and modification 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.
- The Epsilon Comparison Language, for specifying comparison rules, along with a virtual machine for executing rules.

An initial user guide, installation guide, and examples will also be made available. An online tutorial will also be provided for getting started.

Epsilon Architecture Overview

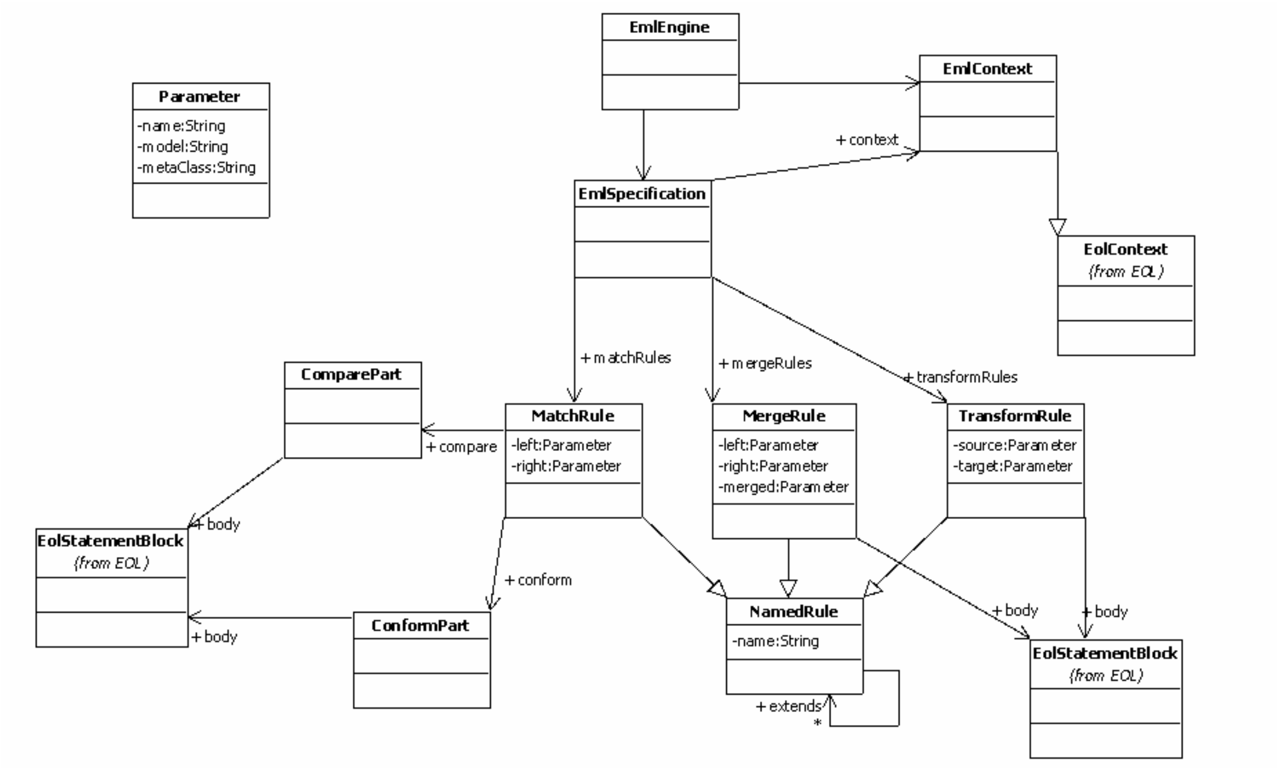
The informal architecture of the Epsilon platform is depicted below.


Eclipse-based Development Tools
Epsilon Merging Language (EML)
Future Epsilon Languages
Epsilon Transformation Language (ETL)
Epsilon Comparison Language (ECL)
Epsilon Object Language (EOL)
Epsilon Model Connectivity (EMC)
EMF
MDR
...

The core of the platform is the Epsilon Object Language (EOL), which provides a metamodel agnostic layer for navigating, querying and modifying models. EOL reuses part of the OCL language 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 modifying models.

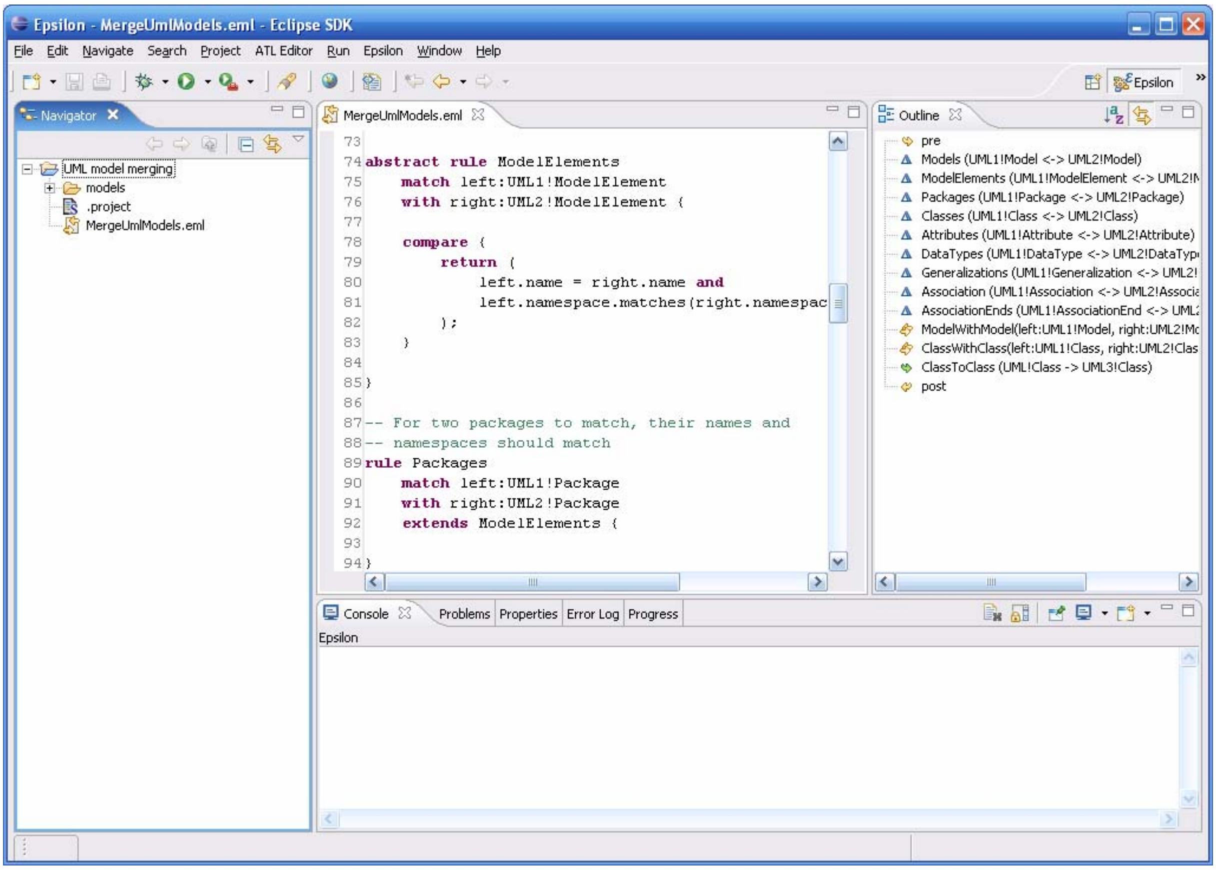
Initial support has been provided for MOF (MDR) models, EMF models, and XML models, but additional models will be investigated.

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



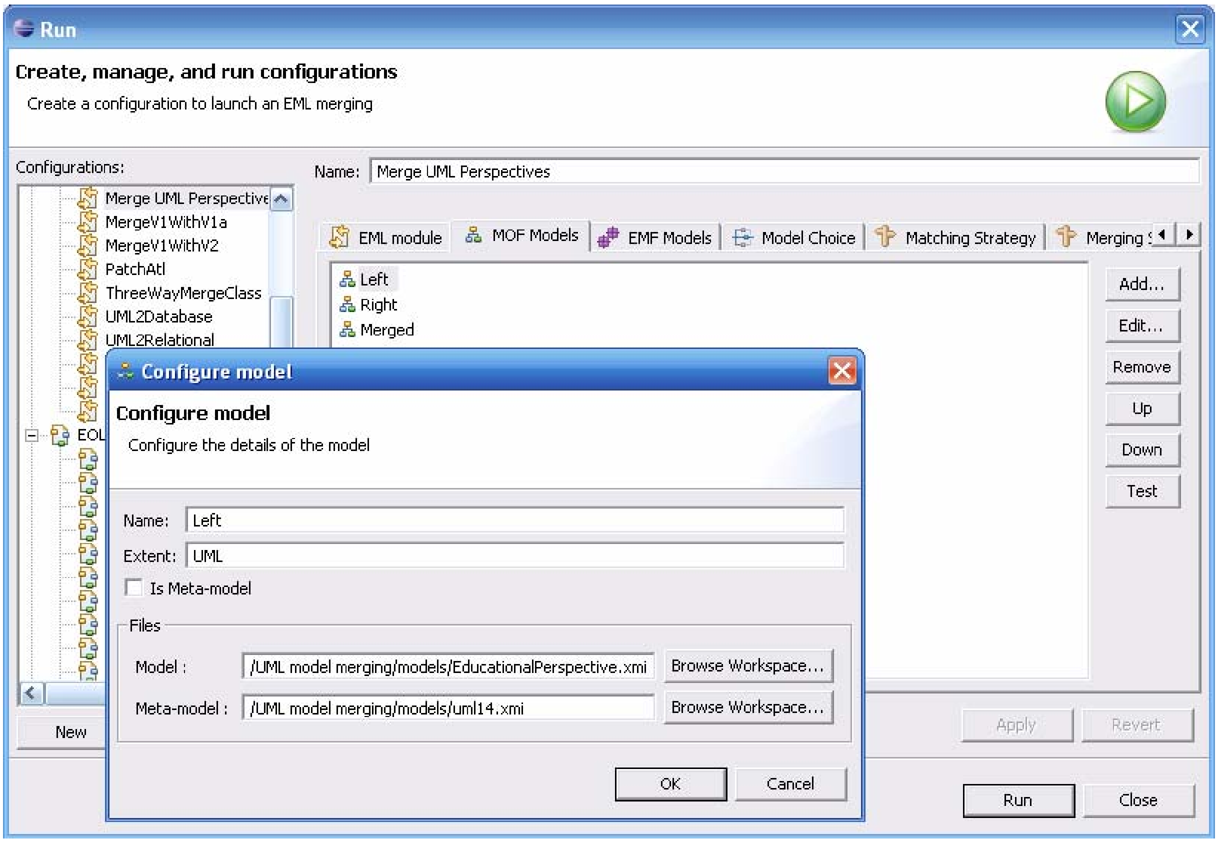
The EML engine applies to an EML specification, defined in terms of a number of rules. These rules make use of EOL for comparison and model navigation. The context in which rules are executed is also supplied via EOL.

Screenshots of the EML editor and the EML launch configuration are provided below.



```
abstract rule ModelElements
    match left:UML1!ModelElement
    with right:UML2!ModelElement {

    compare {
        return (
            left.name = right.name and
            left.namespace.matches(right.namespac
        );
    }

}

-- For two packages to match, their names and
-- namespaces should match
rule Packages
    match left:UML1!Package
    with right:UML2!Package
    extends ModelElements {

}
```

- pre
- Models (UML1!Model <-> UML2!Model)
- ModelElements (UML1!ModelElement <-> UML2!N
- Packages (UML1!Package <-> UML2!Package)
- Classes (UML1!Class <-> UML2!Class)
- Attributes (UML1!Attribute <-> UML2!Attribute)
- DataTypes (UML1!DataType <-> UML2!DataTyp
- Generalizations (UML1!Generalization <-> UML2!
- Association (UML1!Association <-> UML2!Associa
- AssociationEnds (UML1!AssociationEnd <-> UML2
- ModelWithModel(left:UML1!Model, right:UML2!Mo
- ClassWithClass(left:UML1!Class, right:UML2!Clas
- ClassToClass (UML!Class -> UML3!Class)
- post



Create, manage, and run configurations

Create a configuration to launch an EML merging

Name: Merge UML Perspectives

- Left
- Right
- Merged

Configure model

Configure the details of the model

Name: Left

Extent: UML

☐ Is Meta-model

Files

Model : /UML model merging/models/EducationalPerspective.xmi

Meta-model : /UML model merging/models/uml14.xmi

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

Epsilon community

Epsilon was 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 the context of MODELWARE's successor project, MODELPLEX, which started September 2006. MODELPLEX will provide the necessary time and resources to ensure a continuity of development in the component.

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.