ClubUML

CSYE7945 Spring 2013

[Server Side Architecture]

[Richard Do]

[Prashant Shukla]

[Zhe Zhang]

[2/8/2013]

Revision: [1.2]

Table of Contents

Table of Contents 2

Revision History 3

Introduction 4

Detailed Design 5

Upload Factory 6

Parsing an ECORE File 7

Parsing a Papyrus Model 8

Comparing Framework 10

Comparing two Papyrus Model 11

Comparing two Ecore Model 12

Appendix 13

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision** | **Date** | **Author(s)** | **Notes** |
| 1.0 | 2/8/13 | Prashant Shukla | Initial version |
| 1.1 | 02/10/13 | Richard Do | Added Ecore parsing details |
| 1.2 | 2/24/13 | Richard Do | Added Upload Factory Method , Ecore upload code migration, Xmi Compare mechanism |
| 1.2 | 2/24/13 | Prashant Shukla | XMI Parse |
| 1.2 | 2/25/13 | Zhe Zhang | Compare Framework |
|  |  |  |  |

# Introduction

The purpose of this document is to describe implementation & architecture of ClubUML. We have divided the project into the following sections.

1. Upload Mechanism
2. Compare Mechanism
3. Merge Mechanism
4. Download Mechanism

The Assumption is that ClubUML only supports model generated either by ECORE or Papyrus. Details about these UML tools can be found on the web.

Also we can’t compare an ECORE model and a Papyrus model.

Also it can be assumed that the ECORE and Papyrus model were created by different users on different computers.

# Detailed Design

When we upload a file, we need to do the following

1. Determine the type of the file ECORE or Papyrus. This can be as simple as determining the extension of the files.
2. If the file type is Papyrus, make sure we have uploaded all the necessary files to make the comparison (file\_name.uml , file\_name.notation and file\_name.di)
3. If the file type is Papyrus, we need to determine the models that are currently present in the files.
4. If the file type is ECORE, we can start processing the file as ECORE is only used for class diagrams and does not require any other files**.**

## Upload Factory

Figure 1: Class diagram describing the Upload mechanism. This image describes the Factory Design pattern we are using to handle different file types.

Parametrized factory method is used to determine which upload processor object to instantiate and return depending on the extension type of the file(s) passed into the argument. It currently handles Ecore and Xmi file extensions.

## Parsing an ECORE File

The implementation for uploading an Ecore file is basically the same (parse ecore file, create java file, create dot file and generate PNG file), but it is now migrated to the EcoreUploadProcess class in controller.upload package. This new class is used by the new Upload Factory Method previous presented.

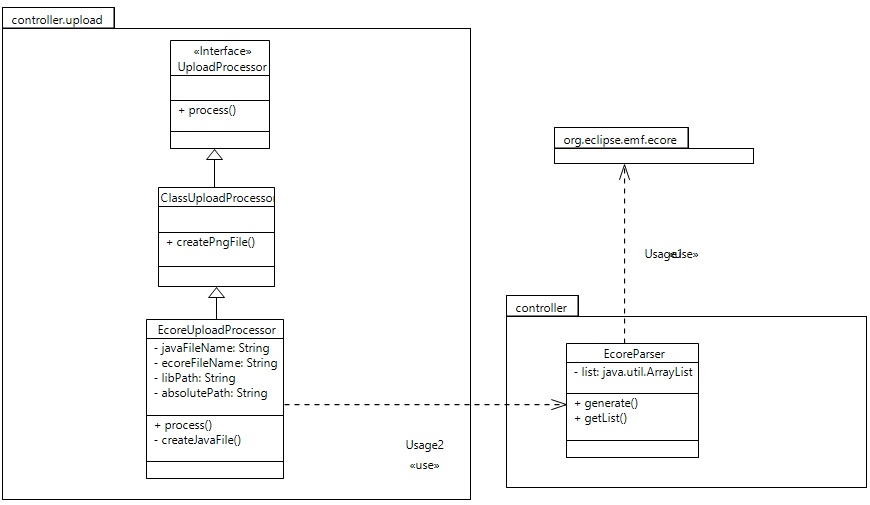


Figure 2

## Parsing a Papyrus Model

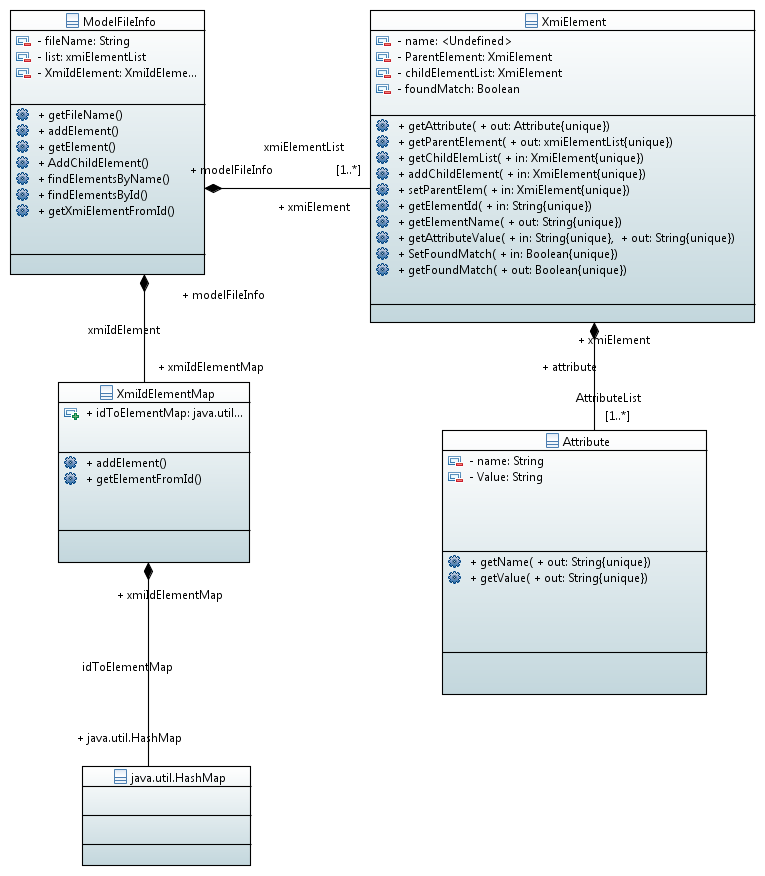


Figure : Class diagram for the XML PARSER

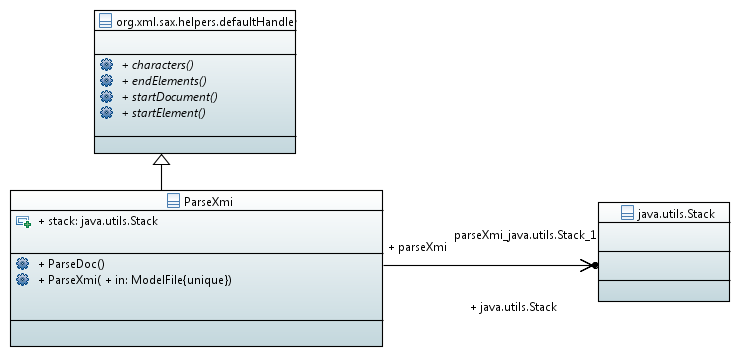


Figure : Class diagram of the XMI Parser.

When a user uploads a Papyrus project (A papyrus project includes 3 files \*.uml , \*.notation and \*.di file).We create an instance of the UmlUploadProcessor (shown in figure 1). We first have to parse the files. The parse mechanism can be found under the uml2parser package of the project. We have divided the package into 5 classes.

1. ParseXmi : This class implements a SAX parser. Please see the documentation on SAXParser for more details. This class opens a XMI files breaks the file into elements and attributes.
2. ModelFileInfo : This class contains a list of all the Elements present in the XMI file. The name of the file the elements are associated with.
3. XmiElement : This class contains a list of all the child elements present in the Element. A list of all the attributes. And parent Element.
4. XmiIdToElement :  So most Elements are referenced using Id. So we created a Hashmap of all the ID to elements.
5. Attribute : An Attribute has a name and value.

Above is a class diagram indicating how we can parse a XMI file that is generated by the Papyrus. Once we parsed the XMI file, based on the information from it we can perform any operation e.g., Comparison, Merging, Models supported etc.

## Comparing Framework

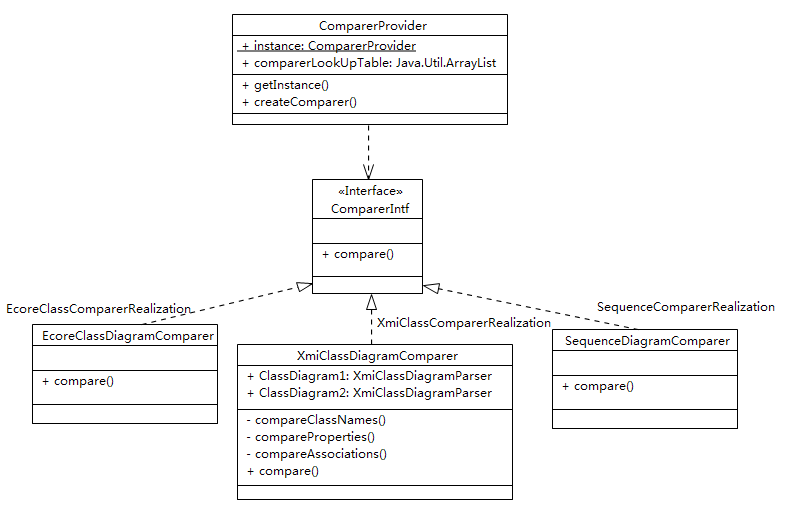


Figure 1: Class diagram describing the Compare mechanism. This image describes the Factory Design pattern we are using to handle different file types.

CreateComparer method is used to determine which comparer object to instantiate and return depending on diagram model type passed into the argument. It currently handles Ecore Class Diagram and Xmi Class Diagram and Xmi Sequence Diagram.

## Comparing two Papyrus Model

XmiClassDiagramComparer class contains parsers to locate the XmiElements for the compare algorithms, such as Class, Attribute and Association compares. The merge UI will invoke the compare() method with an argument for which layer to compare (layer: Class, Attribute and Association), and the compare() method will invoke the specific compare method. For example, invoking compare(“Class”) will invoke the compareClassNames() method and return a JSON string that encapsulates information about differences and similarities between the class diagrams' names.

(Note: The API may change as we work with the Merge UI team. There are a number of re-factoring opportunities as most of the code for parsing was pulled from the Xmi Upload class.)

## 

Figure 5: Implementation of the Compare function

## Comparing two Ecore Model

[TBD]

# Appendix

[If needed, include appendices here, such as details of previous work]