Towards Interoperability of UML Tools for Exchanging High-Fidelity Diagrams

Shihong Huang
Dept. of Computer Science
Florida Atlantic University
shihong@cse.fau.edu

Vaishali Gohel
Dept. of Computer Sciences
Florida Atlantic University
vgohel@fau.edu

Sam Hsu
Dept. of Computer Sciences
Florida Atlantic University
sam@cse.fau.edu

ABSTRACT

In today's global software engineering projects, where development activities are distributed geographically and temporally, it is increasingly important for CASE tools to maintain the information (both syntactic and semantic) captured in the design models. The Unified Modeling Language (UML) is the *de facto* standard for modeling software applications and UML diagrams serve as graphical documentations of the software system. The interoperability of UML modeling tools is important in supporting the model exchange. Tool interoperability is often implemented using XML Metadata Interchange (XMI). Unfortunately, there is a loss of fidelity of the design documentation when transforming between UML and XMI due to the compatibility of different versions of UML, XMI and add-on proprietary information, which hinder reuse. This paper reports on an ongoing study evaluating the interoperability of UML modeling tools by assessing the quality of XMI documents representing the design. Case studies in the paper demonstrate a framework of preserving the fidelity of UML models data when importing and exporting different UML models in a distributed heterogeneous environment.

Categories and Subject Descriptors

D.2.7 [**Software Engineering**]: Distribution, Maintenance, and Enhancement – Documentation, Extensibility, Portability,

General Terms

Documentation, Human Factors

Keywords: UML, XMI, graphical documentation, interoperability, model exchange, design reuse

1 INTRODUCTION

Software designers extensively use the graphical documentation of their design in software development. In addition to the use of textual documentation, graphical documentation can provide more visual information (i.e., layout and relationships) of existing software system. The complexity of diagrams has increased over time, and so have the tools used to produce them [8]. The Unified Modeling Language (UML) is the *de facto* standard for graphically documenting modern software applications and UML diagrams serve as efficient graphical documentations of the system. Tilley et al [12] have conducted a series of studies on assessing the qualitative efficacy of UML diagrams as a form of graphical documentation. The results from their experiments suggest that the UML's efficacy in support of program understanding is limited by factors such as ill-defined syntax and semantics, spatial layout, and domain knowledge.

In addition to focusing on the efficacy of UML diagrams as a form of graphical documentation, the interoperability of UML modeling tools plays an important role in the globalization software development environments, where the development activities are distributed geographically and temporally. Increasing interoperability of UML modeling tools can increase the reusability of UML diagrams; furthermore, it increases the portability and design reuse that will strongly influence the amount of implementation and maintenance effort needed later.

Preserving the fidelity is an important consideration when it comes to tools that produce and/or consume models represented as UML diagrams, or codified in some textual format (e.g., XMI). The ability of modeling tools to preserve the fidelity of design documentation can save considerable cost and drastically reduce the software development cycle. Most software design tools provide support for editing, viewing, storing, and transforming designs, but lack of interoperability support among different modeling tools, thus reduces the chances of reusability. The interoperability of modeling tools is important for reasons such as:

(1) Organizations often use more than one tool in their development environments, and there are various companies that require the design model interchange.

(2) There are different vendors available in market for UML diagramming tools with their own specifications. Therefore, there is need for the common interface between them that understands the generic specification.

(3) In the globalization development environments, coordinating software artifacts, which developed in a distributed manner both geographically and temporally, require interoperability among the design models produced by different modeling tools.

To address the above problems and allow the model interoperability among various tools, this paper adopts XML Metadata Interchange (XMI) as an appropriate interoperability standard for UML models interchange among various UML modeling tools. XMI is an open standard that is used to achieve interoperability of UML diagrams among various tools in distributed heterogeneous environments.

Studies have shown that major UML tools currently provide model interchange by using XMI [2]. However, the transformation between UML and XMI is not complete or standardized. In other words, there is a loss of fidelity of the design information, such as overlapping order, color, and other information, when importing or exporting the models. The loss of data is in part due to the use of different specifications and different versions of UML and XMI in the modeling tools. For example, some loss of data can be identified when importing and exporting the XMI 1.0 or 1.1 files from or to Rational ROSE Enterprise Edition version 0.7. To generalize the data loss problem, the following diagrams (Figure 1 and Figure 2) identify some problem areas that need attention while importing and exporting UML diagrams among tools.

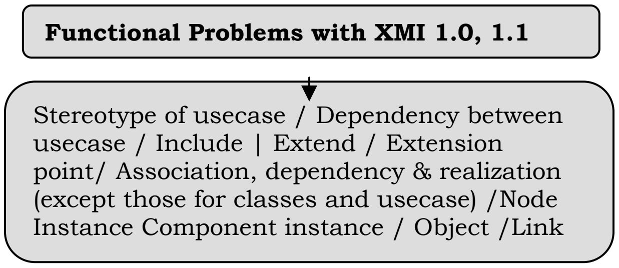

Functional Problems with XMI 1.0, 1.1
Stereotype of usecase / Dependency between usecase / Include | Extend / Extension point/ Association, dependency & realization (except those for classes and usecase) /Node Instance Component instance / Object /Link

Figure 1: Functional Problem Identification

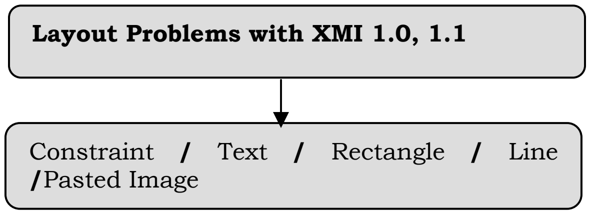

Layout Problems with XMI 1.0, 1.1
Constraint / Text / Rectangle / Line /Pasted Image

Figure 2: Layout Problem Identification

This paper reports on an ongoing study of evaluating the interoperability of UML modeling tools by assessing the quality of XMI documents representing the design. Case studies have been conducted on the proposed framework of assessing UML data interoperability of UML modeling tools among open standard software (e.g., ArgoUML) and dominant industrial tools (e.g., Rational ROSE).

The next section discusses some of the fundamental issues related to metadata and models interchange standards, and related work on model interoperability among CASE tools. Section 3 presents the research goal and approach to identify and maintain data operability among UML modeling tools. Section 4 presents the case study to illustrate the approach for preserving the data fidelity of models interchange. Finally, Section 5 summarizes the paper and outlines avenues for future research.

2 BACKGROUND AND RELATED WORK

There is a set of standards and technologies that can be used in implementing models interchange among different tools. The Object Management Group (OMG) [11] supports most of the standards, some of them, which are related to UML modeling interchange, are discussed in the following section. Similar efforts to implement the interoperability of the UML modeling tools are discussed in the subsequent section.

2.1 Background

Some of the general terms related to metadata and model interchange are as follows:

UML:

Unified Modeling Language (UML) [6] is a standard language for designing, specifying, constructing, developing and documenting the artifacts of a software system. The industry accepted standard versions are regulated by OMG and the latest published version is UML 2.1.1 [5]. UML diagrams serve as graphical documentations of software system.

XMI:

XML Metadata Interchange (XMI) is an eXtensible Markup Language (XML®) based document that represents model elements and their information in an interchangeable format. It is the standard specified by Object Management Group (OMG®) [11] for exchanging information via XML. XMI can be used for representing any metadata whose metamodel can be represented in MOF. The most common use of XMI is as an interchange format for UML models. The current version of XMI captures only the semantic information of UML model information (i.e., the relationships), while discarding most of the visual details of a particular UML diagram. In other words, it does not preserve diagrammatic layout. Different tools can use their own methods to represent the layout information in XMI. This is typically done using the extension mechanisms to XMI [7].

MOF:

Meta-Object Facility (MOF) is a four-layer architecture that is used for generation of meta-models and can be exported from one platform to another, rendered in different formats and addresses the issues of both a designer and a programmer through the architecture.

Figure 3 describes the generation of XMI by combining XML and MOF.

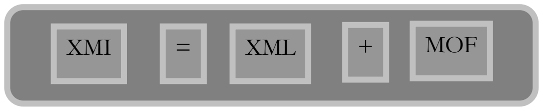


Figure 3: XMI Formula

The comprehensive support for XMI based model exchange is supported in UML [6]. XMI analysis for UML designing tools allows organizations to leverage existing UML models from proprietary tools to other modeling tools and vice versa with minimum data loss. This portability provides benefits of design reusability and flexible UML design interchange using standard notation.

Metamodel directly implies that XMI serves as an interchange format for UML. Specifically, XMI intends to help UML CASE tools to exchange their data models among each other to increase the reusability [6][3][4].

The transformation of the models into XMI is indeed a major breakthrough in facilitating interoperability across various tools and platforms [3]. The XMI format 1.0 was chosen for export and it provides support up to UML version 1.3.

The next step is to understand the technical details on how to make the model exchange interoperable among various tools. Figure 4 displays the relationship among three extension mechanisms that could be used to export the additional features required in a metamodel, according to OMG UML 2.1.1 specification document [5].

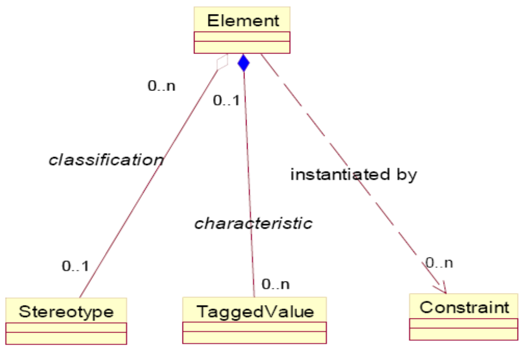


Figure 4: Relation between Extensibiilty Mechanisms for an Element [1]

The most important point to note here is that the UNISYS plug-in for Rational ROSE exports diagram information through XMI (DI) (XMI Diagram Interchange) [5]. The geometric information is appended at the end of the file when the diagram information generation is selected in the exporter. UML semantics defines the metaclass information while UML notation defines the diagram information [5].

2.2 Broader View of Model Interoperability

To illustrate the role of XMI in model transformation, Figure 5 depicts the relationships between various metalayers, XMI production, and XMI reader and writer modules of XMI parser. The XMI parser falls in the M2 layer within the XMI metamodel. In order to make the UML model interoperable, the XMI parser is divided into two modules: XMI reader and XMI writer.

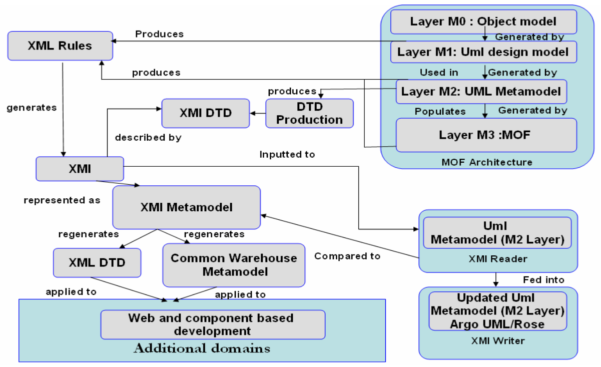


Figure 5: Role of XMI in Model Transformation

2.3 Related Work

There is a significant amount of ongoing efforts into providing the interoperability of UML modeling tools. However, there has been a lamentable lack of systematically assessing the fidelity of both semantic data and visual layout of UML models when importing and exporting the models between different tools.

A few researchers have conducted experiments in assessing preservation of the fidelity of UML diagrams. Persson et al [7] compared XMI supporting features in nine UML modeling tools, which included both open source and proprietary tools. Their results suggested that there is both successful and unsuccessful transfer of the models and the XMI-based model interchange between UML modeling tools is weakly supported in practice. At the time of their study, none of the tools supported XMI 2.0, but now we have ArgoUML 0.24 that supports XMI 2.0. None of the tool supports UML 2.0; this feature is very important for layout preservation. Their final result stated that the layout information could not be preserved in open source software [7]. Thus the work carried out by them is very useful to understand the XMI and UML combinations in the past and their research work is the initial point where anyone can start research on UML diagram interoperability.

For interoperability of modeling tools, Stevens pointed out that XMI is the need to have a standard way for UML tools to interchange UML models [10]. This paper discussed how XMI could be exploited for performing model analysis and housekeeping tasks and for the integration of third party or in-house tools. This paper also explained the use of scripts to integrate the UML tools that supports XMI with other open source tools. This approach is similar to the functionality performed by the XMI parser generated in our methodology.

UML model exchange using XMI by Jiang et al [2] provided the information about how XMI can be used to exchange data and metadata between different tools in distributed heterogeneous environments. The first step in their research was to parse the XML file and store the result in a repository, and then used the repository API to access the information instantly [2]. Their paper concluded with the result that one of the practical problems of XMI is that it does not support the model exchange among various tools. The target of our research is to resolve these practical problems in XMI and to make it possible to support model interoperability by creating XMI parser that can identify and provide the lost information during model interchange.

3 RESEARCH GOAL AND APPROACH

3.1 Research Goal

The overall goal of this research is to address the interoperability problems of UML modeling tools for the purpose to increase reusability of design documentation. The sample UML modeling tools we selected include dominant industry tool (e.g., Rational ROSE) and open source software (e.g., ArgoUML).

Figure 6 shows the workflow of the system required for making the UML diagrams reusable by converting them to XMI, parse the XMI file, and make them compatible for interpretation by other tools.

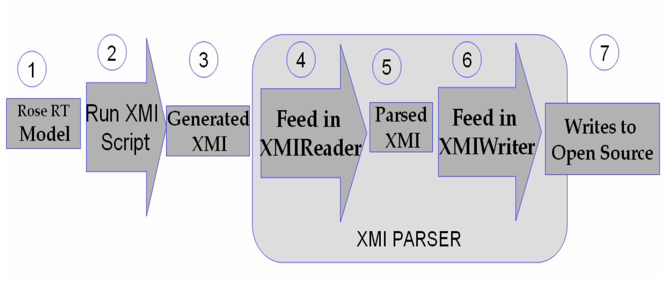


Figure 6: Workflow of Implementation

Currently, a major limitation of the XMI standard is that it does not include a standard way to save layout information. For example, though we have standard ways to record certain attributes in user's UML model, we do not have standard ways to record the size and their layout details. Although layout information is not relevant for code generation, however, to use a language effectively requires mastery of not only its syntax and semantics, but also of its diagrammatically [5].

The extension mechanism of XMI, such as "diagramming.diagram" tag, can be used to preserve the layout information during interoperability of UML diagrams amongst other tools.

3.2 The Approach

The approach proposed by Jiang et al [2] for the UML model interchange among Rational ROSE and ArgoUML is to parse the XML file and store the result in a repository first, and then use the repository API to access the information instantly. This is a good approach but it is not complete yet.

In our approach, XMI version that is compliant with UML 1.4 is fed into the XMI Reader, which is presented as a module in the XMI parser. The parser then generates the MOF (Meta-Object Facility) back that is used for the metamodeling and mapping new elements. XMI Writer that is also part of the module of the XMI parser is then used to output the generated XMI to open source, e.g., ArgoUML.

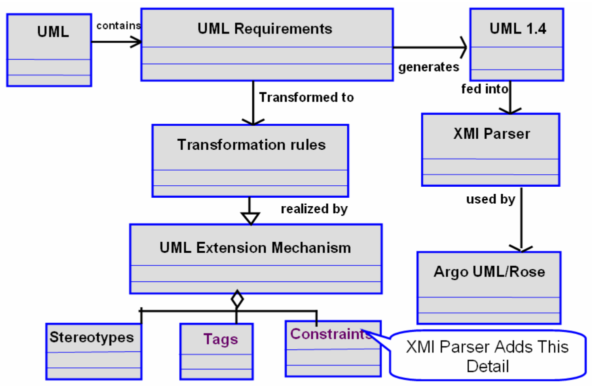


Figure 7: XMI Parser in UML Modeling Diagram

The tags in Figure 7 are the key components used in the export of additional information from the UML metamodel. The tags can be used to export any information in the model that is not supported in the syntax of the XMI.

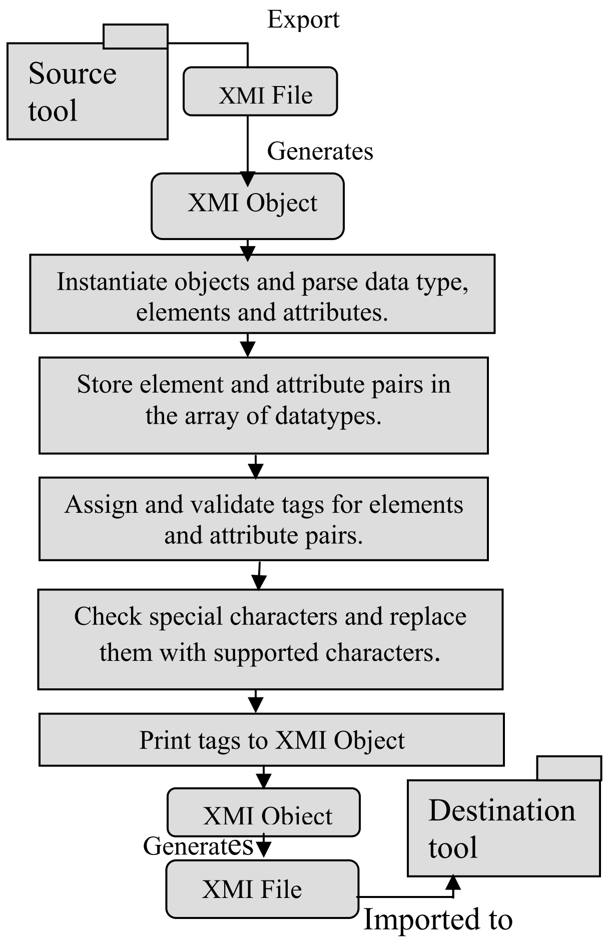

Export
Source tool
XMI File
Generates
XMI Object
Instantiate objects and parse data type, elements and attributes.
Store element and attribute pairs in the array of datatypes.
Assign and validate tags for elements and attribute pairs.
Check special characters and replace them with supported characters.
Print tags to XMI Object
XMI Object
Generates
XMI File
Destination tool
Imported to

Figure 8: Flowchart of Processing Tags in XMI

The flowchart in Figure 8 states the methods that tags are exported among various models by assigning tags, replacing special characters with the special characters supported by the target tool, and writing them before export.

4 CASE STUDY

This section demonstrates how to preserve data information of UML diagrams between different tools. The two sample tools used in the case study are dominant industry modeling tool (e.g, Rational ROSE) and open source software (e.g., ArgoUML).

There are some known facts about the compatibility of different versions of XMI and UML. The combination of XMI versions 1.0 and UML versions 1.3 is old and are rarely used for model interoperability. The combination of XMI version 1.1 and UML version 1.3/1.4 are currently widely used by various CASE tools. The combination of XMI version 2.1 and UML version 1.4/2.0/2.1 are the challenge faced by industry for data and layout preservation.

In this case study, we first generate the XMI file from UML diagram that is imported or exported from the Rational ROSE 7.0 or ArgoUML 0.24. The second step is to pass the XMI file as the input to the parser. In the third step, the parser processes the XMI file by validating and making the file compatible to the tool to which file is to be exported. In the last step, the modified XMI file will then be passed as an output to the open source, for instance AgroUML. The following architectural diagram in Figure 9 shows this transformation process.

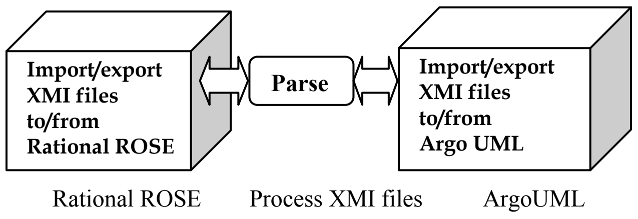

Import/export XMI files to/from Rational ROSE
Parse
Import/export XMI files to/from Argo UML
Rational ROSE
Process XMI files
ArgoUML

Figure 9: Architectural Diagram

UML model could be completely interoperable among various CASE tools if it can preserve all the functional requirements. The following sections describe the details of the case study.

4.1 The Need of XMI Parser

The need of an XMI parser is unavoidable due to the following two reasons:

(1) Subsequently released XMI versions have syntax changes that make it harder to make XMI forward and backward compatible. Thus we identify the need of a parser that makes XMI interoperable.

(2) Import and export of the same XMI elements can be done in varied ways. As an example, link can be imported and exported as tagged value, path or supported in syntax. In addition, vendors sometimes use their proprietary forms of export that cause the information loss while importing to other CASE tools. Thus we need a parser that understands all the generic ways and serves to specific requirement of target CASE tool.

4.2 XMI/UML Support in CASE Tools

The first goal of our experiment is to identify the availability of various XMI and UML version combinations in different CASE tools, such as Rational ROSE and ArgoUML. Results of the analysis of tools are shown in Table 1, which illustrates the support provided by the dominant industry standard Rational ROSE enterprise edition 7.0 for UML versions 1.3, 1.4, 2.0 and XMI versions 1.0, 1.1, 1.2.

In the following two tables, "Y" means that the tool supports the combination of respective versions of UML/XMI and "N" means that the tool specified does not support the combination of respective versions of UML/ XMI.

Table 1: Rational ROSE Enterprise Edition 7.0 with the XMI Patch

UML/XMI	1.0	1.1	1.2
1.3	Y	Y	N
1.4	Y	Y	N
2.0	N	N	N

Table 2 illustrates the support provided by the open source software Argo UML 0.24 for UML versions 1.3, 1.4, 2.0 and XMI versions 1.0, 1.1, 1.2.

Table 2: ArgoUML 0.22 and 0.24

UML/XMI	1.0	1.1	1.2	Version
1.3	Y	Y	N	0.22
1.4	Y	Y	Y	0.24
2.0	N	N	N	0.24

4.3 Testing UML Interoperability

To test the interoperability of the UML diagrams, the next step in our case study is to apply the architectural diagram in Figure 10 with the specific example. Below experiment shows the loss of layout information and some data elements, when the class diagram is exported and imported using UML 1.3 and 1.4 and XMI 1.0 and 1.1 formats between dominant industry standard Rational ROSE and open source ArgoUML. Figure 10 is the sample class diagram used for this case study.

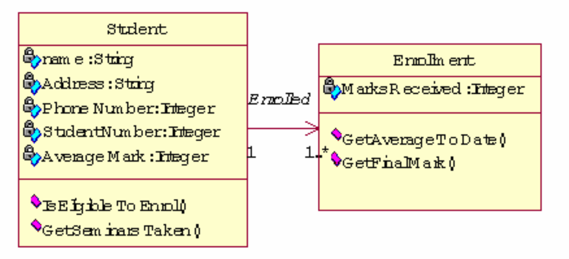


Figure 10: Sample Class Diagram

The experiment is divided into four scenarios based on the tools that are used to import and export particular combination of different versions of UML and XMI.

Scenario I: Import/Export XMI from Rational ROSE to Rational ROSE:

UML model interchange from ROSE Enterprise Edition 7.0 to ROSE Enterprise Edition 7.0 provided successful interchange of both data and layout of class diagram for UML 1.3 and XMI 1.0 and 1.1. Figure 11 shows the result of the import and export.

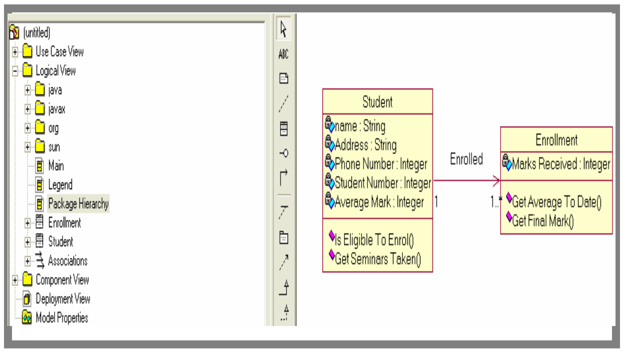


Figure 11: Diagram and Data Preservation of Rational ROSE to Rational ROSE Export/Import

Whereas for the combination of UML 1.4 and XMI 1.0 and 1.1, only the data can be successfully imported and exported; there is no layout preservation.

Scenario II: Import/Export XMI from ArgoUML 0.24 to Argo UML 0.24:

Model interchange among the latest version of ArgoUML 0.24 to ArgoUML 0.24 using XMI import/export is successful in preserving the data but no diagramming layout is preserved for UML 1.3/UML1.4 and XMI 1.0/XMI 1.1. Figure 12 shows the result of the import and export.

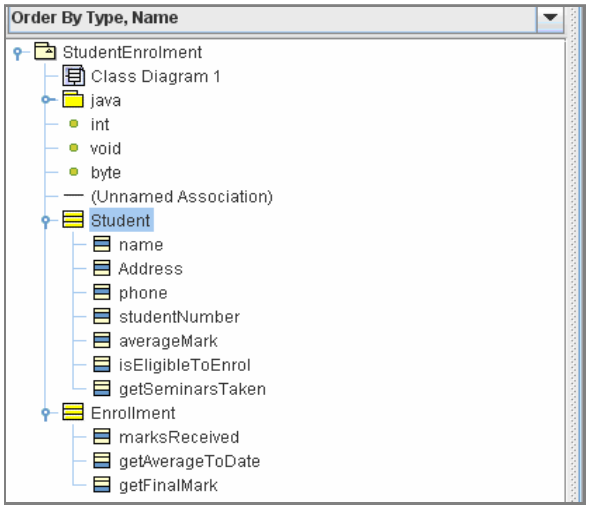


Figure 12: Data Preservation of ArgoUML 0.24 to ArgoUML 0.24 Export/Import

Scenario III: Export XMI from Rational ROSE and Import to ArgoUML 0.24

In this scenario we tried importing and exporting the same class diagram between two different tools. At first, we exported the above class diagram from ROSE using UML 1.3 and XMI 1.0/1.1 and respectively imported that to ArgoUML. Data is preserved in this case but there is no diagramming information preserved. Figure 13 shows the result of the import and export.

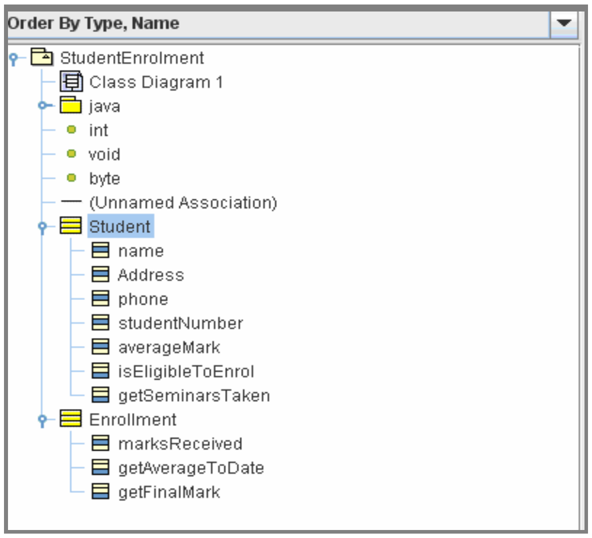


Figure 13: Data Preservation of Rational ROSE to ArgoUML 0.24 Export/ Import Respectively

In our next step we exported the same class diagram using UML 1.4 and XMI 1.0/1.1 and respectively imported that to ArgoUML. This transformation produced the following XMIformat error shown in Figure 14.

```
XMI format error :
org.argouml.model.XmiException:
javax.jmi.xmi.MalformedXMIException:
org.netbeans.lib.jmi.util.DebugException:
 The same value of xmi.idref used second time:
X.NOTFOUND,file:/C:/DOCUME~1/Vishali/LO
CALS~1/Temp/zargo_model_36418.xmi
```

Figure 14: Error Message of ROSE to ArgoUML 0.24 Export/ Import Respectively

To fix this problem we will need an XMI parser, which enables the file exported from ROSE compatible with ArgoUML during UML model importing.

The parser parses the XMI file and makes it compatible for interoperability among Rational ROSE and ArgoUML by identifying and modifying the tags that are not followed by ArgoUML, and the generated error described above. Figure 15 shows the successful data interoperability of UML 1.4 and XMI

1.0/1.1 between Rational ROSE and ArgoUML after applying our XMI parser that fixed the problem.

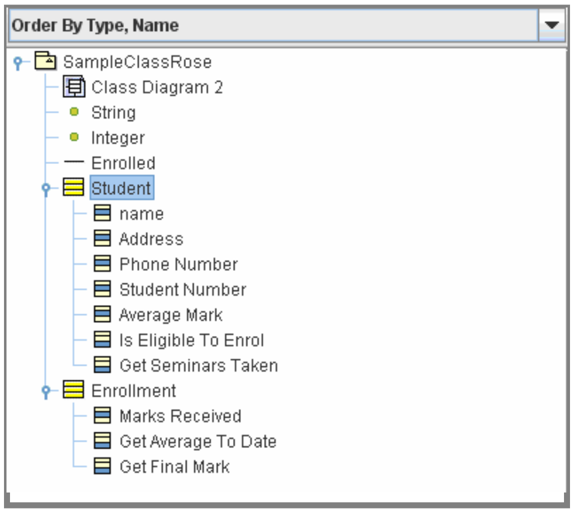
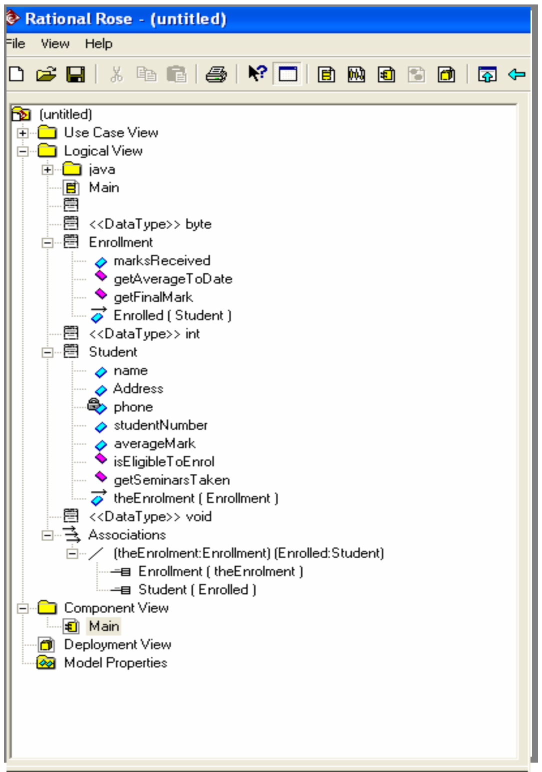

Order By Type, Name
SampleClassRose
Class Diagram 2
String
Integer
Enrolled
Student
name
Address
Phone Number
Student Number
Average Mark
Is Eligible To Enrol
Get Seminars Taken
Enrollment
Marks Received
Get Average To Date
Get Final Mark

Figure 15: Parsed ROSE XMI Imported to ArgoUML

Scenario IV: Export XMI from ArgoUML 0.24 and Import to Rational ROSE

In this scenario we tried exporting the UML diagrams from ArgoUML 0.24 using *UML 1.3/1.4 and XMI 1.0/1.1* and respectively import them to Rational ROSE. Here we got 100% success for data transmission but no diagramming layout was preserved. Figure 16 shows the details of the data preservation.



(untitled)
Use Case View
Logical View
java
Main
<<DataType>> byte
Enrollment
marksReceived
getAverageToDate
getFinalMark
Enrolled (Student)
<<DataType>> int
Student
name
Address
phone
studentNumber
averageMark
isEligibleToEnrol
getSeminarsTaken
theEnrolment (Enrollment)
<<DataType>> void
Associations
(theEnrolment:Enrollment) (Enrolled:Student)
Enrollment (theEnrolment)
Student (Enrolled)
Component View
Main
Deployment View
Model Properties

Figure 16: Data Preservation of XMI Imported to Rational ROSE

Figure 17 summarizes the case study. Importing and exporting of UML file in XMI format for the class diagram in the same tool gave success in data transmission; but when we tried exporting and importing XMI file in different tools, such as ArgoUML and Rational ROSE, ArgoUML produced data format error as described Figure 14. This error can be taken care of by our parser as the result shown in Figure 15.

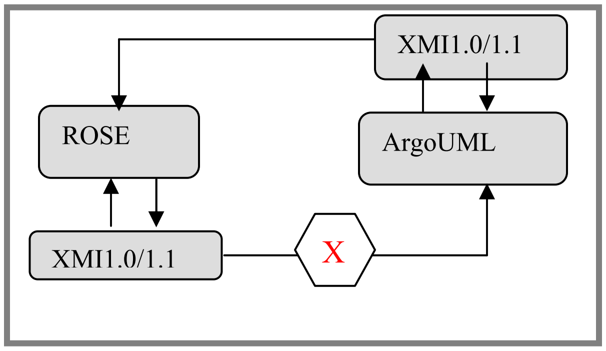


Figure 17: Results of UML1.4 Data Import and Export in Rational ROSE and ArgoUML.

5 SUMMARY

The major focus of our work is to provide the reusable UML diagrams without any data loss among different UML modeling tools. The representative tools used in the case study are the dominant industry standard (e.g., Rational ROSE) and open source software (e.g.,ArgoUML). By applying a XMI parser that includes both a XMI Reader and a XMI Writer, we can identify and modify the data loss during the model interchange. Preliminary analysis of the results suggest that there is considerable room for improvement in popular UML tools in terms of their support of XMI, and in terms of the ability of XMI itself to properly represent UML diagrams.

Future work will include the development of XMI file reader and writer under the name of XMI parser using java. The parser will automatically detect and modify all the XMI tags that are responsible for inadequate data interchange between ROSE and ArgoUML. We also plan to implement the parser using the XMI extension mechanism. The parser will take the XMI file as the input and will stabilize and preserve all the required attributes and generate the output XMI file when imported to the other tool will produce the same design layout without loosing the detailed design attributes like cardinality for use case. The properties of the design attributes can be prioritized by using Analytical Hierarchy Process (AHP) [9].

REFERENCES

[1] Gopinath, Shankar. "Real-Time UML to XMI Conversion." Master Thesis, Stockholm, Sweden 2006.

[2] Jiang, J. and Systa, T. “Exploring Differences in Exchange Formats – Tool Support and Case Studies.” In Proceedings of Seventh European Conference on Software Maintenance and Reengineering (CSMR’03: March 26-28, 2003; Benevento, Italy) pp. 388-398. Los Alamitos, CA: IEEE Computer Society Press, 2003.(DOI Bookmark:

http://doi.ieeecomputersociety.org/10.1109/CSMR.2003.1192448)

[3] Object Management Group- XML Metadata Interchange (XMI) specification version 1.0-2.0 http://www.omg.org/technology/documents/modeling_spec_catalog.htm#XMI

[4] Object Management Group. The Meta Object Facility (MOF) Core, version 2.0. Online at http://www.omg.org/docs/formal/06-01-01.pdf

[5] Object Management Group. UML 2.1.1 Formal Specification. http://www.omg.org/docs/formal/07-02-03.pdf.

[6] Object Management Group: OMG, Introduction to OMG's Unified Modeling Language (UML) Object management. http://www.omg.org/gettingstarted/what_is_uml.htm.

[7] Persson, A., Gustavsson, H., Lings, B., Lundell, B., Mattsson, A., Arlig, U. "OSS tools in a heterogeneous environment for embedded systems modeling: an analysis of adoptions of XMI." *In Proceedings of the Fifth Workshop on Open Source Software Engineering* (5-WOSSE: May 17, 2005; St. Louis Missouri, USA).

[8] Robbins, Jason Elliot. "Cognitive Support Features for Software Development Tools." PhD Dissertation University of California, Irvine, 1999.

[9] Saaty,T. L. *Fundamentals of the Analytic Hierarchy Process*. RWS Publications, 4922 Ellsworth Avenue, Pittsburgh, PA 15413, 2000.

[10] Stevens P. "Small-Scale XMI Programming: A Revolution in UML Tool Use?" *In Automated Software Engineering*, 10, 1(2003). pp. 5-21.

[11] The Object Management Group (OMG) online at http://www.omg.org/

[12] Tilley, S. and Huang, S. "A Qualitative Assessment of the Efficacy of UML Diagrams as a Form of Graphical Documentation in Aiding Program Understanding." Proceedings of the 21st Annual International Conference on Design of Communication (SIGDOC 2003: October 12-15, 2003; San Francisco, CA), pp. 184-191. ACM Press: New York, NY, 2003.