2011

Interoperability Between AWSOME and Other Tools Using Model Driven Architecture

Chitra Srinivasan
Wright State University

INTEROPERABILITY BETWEEN AWSOME AND OTHER TOOLS USING MODEL DRIVEN ARCHITECTURE

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science

By

Chitra Srinivasan
Bachelor of Engineering in Electronics and Communication Engineering,
University of Madras, INDIA, 2004

2011
Wright State University

WRIGHT STATE UNIVERSITY

SCHOOL OF GRADUATE STUDIES

January 10, 2011

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY Chitra Srinivasan ENTITLED Interoperability Between AWSOME and Other Tools Using Model Driven Architecture OF THE REQUIREMENTS FOR THE DEGREE OF Master of Science.

Thomas C. Hartrum, Ph.D
Thesis Co-Advisor

Mateen Rizki, Ph.D
Thesis Co-Advisor

Mateen Rizki, Ph.D
Interim Department Chair

Committee on

Final Examination

Thomas C. Hartrum, Ph.D

Mateen Rizki, Ph.D

Yong Pei, Ph.D

Andrew Hsu, Ph.D.
Dean, School of Graduate Studies

ABSTRACT

Srinivasan, Chitra. M.S, Department of Computer Science and Engineering, Wright State University, 2011. Interoperability between AWSOME and Other Tools Using Model Driven Architecture.

AFIT Wide Spectrum Object Modeling Environment (AWSOME) is built on an Abstract Syntax Tree (AST) which is the meta-model of AWSOME. Transformations happen internally in AWSOME to transform anything in abstract to concrete. Earlier efforts in AWSOME were focused on developing tools to work on the AST. The goal of this thesis is to make AWSOME interoperable with other available tools. To achieve this goal, the Model Driven Architecture (MDA) concept is used. MDA is a framework aimed at portability, interoperability and reusability among different tools. Among many tools that use MDA context, EclipseUML 2008 and Dresden OCL Toolkit were considered to make interoperable with AWSOME.

This thesis focuses on making a model generated by AWSOME to be loaded in to EclipseUML 2008 and a model generated by EclipseUML 2008 to be parsed in to AWSOME. Also the Object Constraints Language (OCL) file generated by AWSOME should be able to be loaded in Dresden OCL Toolkit. The model to model mapping between these tools and the level of interoperability achieved between these tools is discussed in detail. Understanding the different functionalities of these tools and deciding the level of interoperability between the tools played a vital role in the design decisions made. All together three translation or transformation tools were developed to interface EclipseUML 2008 and Dresden OCL Toolkit with AWSOME.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would begin by thanking my thesis advisor Dr Thomas C. Hartrum for providing me with the opportunity to work with him. I am grateful to Dr Hartrum for his constant support and guidance throughout the course of this thesis work. His way of explaining things and defining the research problem kept me motivated and focused. I would especially like to thank him for his patience and knowledge in reviewing and editing the contents of this document. I am also thankful to Dr. Mateen Rizki and Dr. Yong Pei for their invaluable feedback on this work.

Last, but not least, I am grateful to all my family members for their love and support.

1. INTRODUCTION

1.1 Context

The software lifecycle can be generally divided into the following phases:

- Requirements Specification,
- Analysis and Design,
- Implementation, Testing and;
- Maintenance

The information gathered and documented in the first two phases (*i.e.,* requirement specification, analysis and design) rapidly changes during implementation. When it comes to the maintenance phase, there is no documentation or diagram available close to the current working model. By performing maintenance on the specifications, one can modify the model close to the user's conceptual model. This makes the process of automation of tools easier, as we know that the specification is up to date and conforming to the user requirements. [1]

Model Driven Architecture (MDA) is a proposal for developing software using models. So far, software development only uses models for documentation or as a guide for implementing. The goal of MDA is to promote models as the kernel of the development process. [2] According to the MDA concept, the specification (close to user requirements) is referred to as the Platform Independent Model (PIM) and the design model (close to implementation) is referred to as the Platform Specific Model (PSM). The automation of tools is accomplished by having transformation tools for converting PIM to PSM and PSM to code. The same PIM can be transformed to different PSMs and to their respective code as depicted in the Figure 1.1.

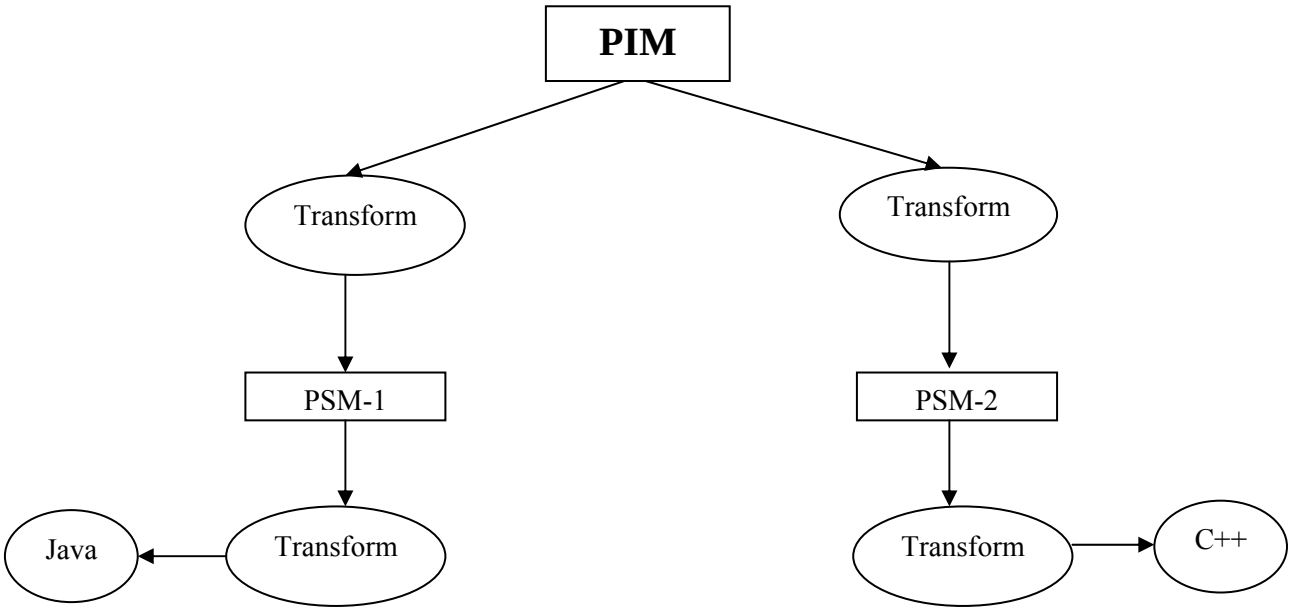


Figure: 1.1: Transformation of PIM to different PSMs and code.

Because automatic transformation is increasingly gaining in popularity, often software tools need to communicate with each other. A typical example would be the communication between an editor and analysis tool. From this arises the need for interoperability among different tools. The idea of MDA is to define interface standards so that different vendors can generate interoperable tools. The interoperability among cross platform applications can be achieved by generating bridges between them.

1.2 AFIT Wide Spectrum Object Modeling Environment (AWSOME)

Like the automatic PIM to PSM and PSM to code transformation tools explained in the MDA context, locally a similar system called AWSOME is being developed at Wright State University based on an earlier effort at the Air Force Institute of Technology (AFIT). Instead of having two specific models *i.e.*, PIM and PSM as in MDA, AWSOME has single model that is able to transform anything from abstract to concrete. Transformations happen internally in

AWSOME to refine the PIM and PSM. Java code generation is supported. Figure 1.2 provides an example flow chart of AWSOME transformations.

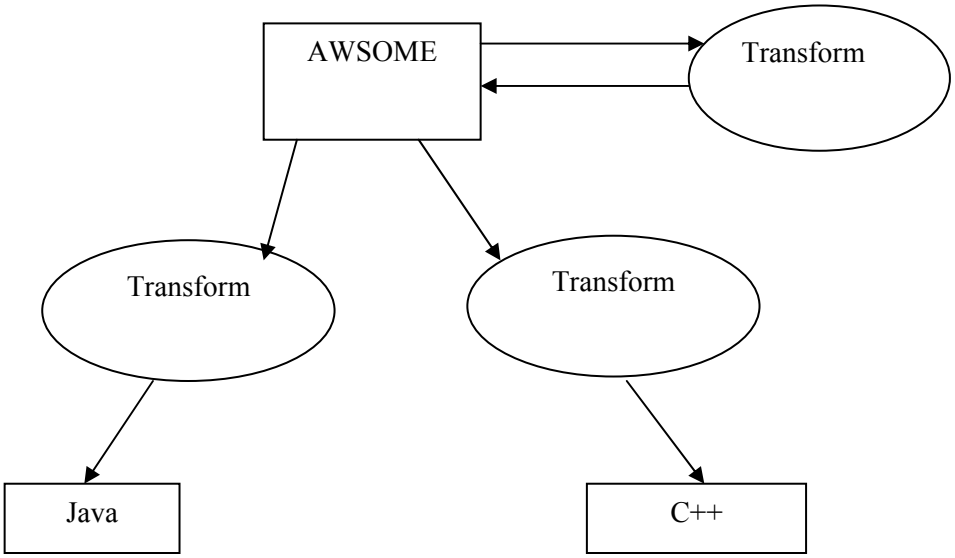


Figure 1.2: An example for AWSOME transformations

1.3 Objective

It should be possible for AWSOME tools to interface with other tools in the MDA context. Compatible files can be transferred between the AWSOME model and other tool through standard formats. In this thesis, we focus on demonstrating the interoperability between AWSOME tools and other available tools.

1.4 Approach

The following approach is used in this research to achieve the objective of making AWSOME interoperable with other tools

- Analyze the existing tools that use MDA context.
- Select the tools that are most suitable.

- Define mappings between AWSOME and the selected tools.
- Formulate the restrictions and assumptions to be made to resolve the discrepancies between the functionalities of the tools and AWSOME.
- Generate test cases to demonstrate the interoperability.
- Record the results.

1.5 Scope

Many tools are currently available that use the MDA context. The scope of this thesis will be limited to making AWSOME interoperable with two such tools. The interoperability of the tools will be demonstrated by being able to translate Unified Modeling Language (UML) and Object Constraint Language (OCL) between AWSOME and the selected tools.

1.6 Outline of thesis

This thesis is an effort to demonstrate the interoperability of tools under the MDA context. Chapter 2 describes the background needed to understand the rest of the thesis including a discussion of MDA, XMI, UML, OCL, AWSOME and the tools considered to be interfaced with AWSOME. Chapter 3 is an in-depth explanation of the requirements analysis and the tools selected to be interfaced with AWSOME. In Chapter 4 we discuss design issues. This chapter discusses in detail the design approach followed for mapping AWSOME with the selected tools. Chapter 5 demonstrates how the test cases are implemented and the test results after each test case is executed. It also focuses on test cases and the tools used to execute the test cases. Finally we conclude the thesis in Chapter 6 and suggest future work that can be applied to this thesis.

2. BACKGROUND

2.1 Introduction

This chapter presents a description of the Model Driven Architecture (MDA), XML Metadata Interchange (XMI), Unified Modeling Language (UML), Object Constraint Language (OCL) and the AFIT Wide Spectrum Object Modeling Environment (AWSOME). A detailed description of several tools that were considered to be interfaced with AWSOME is also presented.

2.2 Model Driven Architecture (MDA)

The Object Management Group (OMG) defines MDA as a framework to develop systems. It has been promoting MDA in an effort to gear towards satisfying customer needs. MDA has increased the level of abstraction and reuse to a great extent by introducing meta-models and meta-meta models. [3, 4, 7] By describing different models using meta-models, transformation among different models can be achieved and thus the interoperability and automation through tools can be obtained. [3, 4]

2.2.1 Principles of MDA

There are three levels of models in MDA that increase the level of abstraction:

- Computation Independent Model (CIM),
- Platform Independent Model (PIM) and;
- Platform Specific Model (PSM).

The MDA Life Cycle is illustrated and explained in Figure 2.1. [1] After the requirements are collected from the customer, they are modeled as CIM. Using the CIM models as input, in the analysis phase the PIM models are outputted. In the low level design, the PIM models are transferred to PSM models using a transformation tool. After generating both the PIM and PSM models in the previous phases, the model validation is done. Any changes made to the models are reflected both in PIM and PSM by iterating back to the analysis phase.

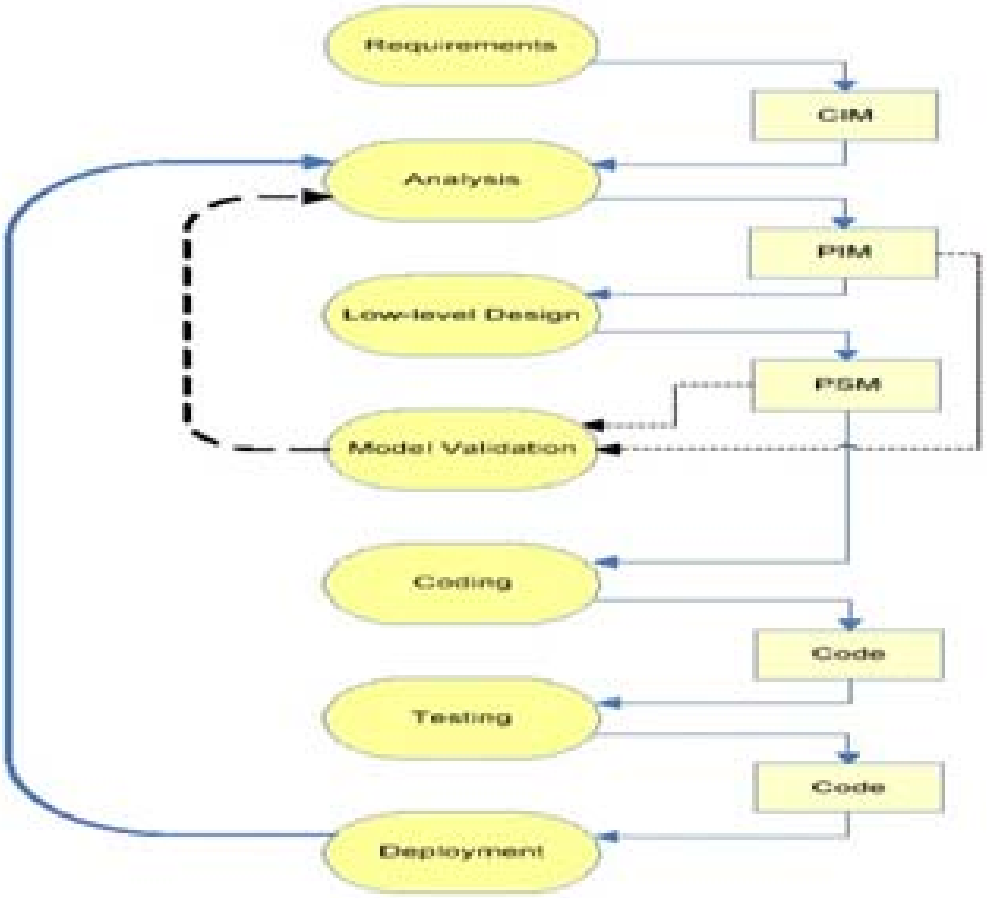


Figure 2.1: An example for MDA Life Cycle Model [1][4]

Thus, consistency in the models is maintained. The PSM models are transformed to code in the coding phase which is followed by testing and deployment. After the user acceptance testing and deployment, if any changes are required then an additional iteration is made back to the analysis phase.

2.2.2 Computational Independent Model (CIM)

Computation Independent model is the domain model of the system in which the requirements are modeled based on the environment in which the system will be used. As the name suggests, the CIM ignores of the technical details of the system. In an MDA perspective, all the requirements defined in the CIM should be able to be tracked by the PIM and PSM. It serves as a source to the PIM and PSM. [5]

2.2.3 Platform Independent Model

PIM not being technology specific gives the description of the system. The system is modeled on how to support the user requirements and to find the solutions for problems specified in the CIM. [5]

2.2.4 Platform Specific Model

PSM contains the details represented in PIM in addition to platform specific details. Any issues or exceptions due to the implementation platform are reflected in the PSM. Any changes done to PIM can be reflected in the PSM by automatic code regeneration. Hence PSM is kept up to date with PIM. [5]

2.2.5 Transformations

One of the key aspects of MDA is transformations. Transformations aid interoperability between different tools. The transformations can be from PIM to PSM and PSM to code. It can also be called mapping. MDA provides specifications for each of these types of mapping. Mappings can be of two types, as described below. [5]

- Model to model mappings
- Model elements mappings

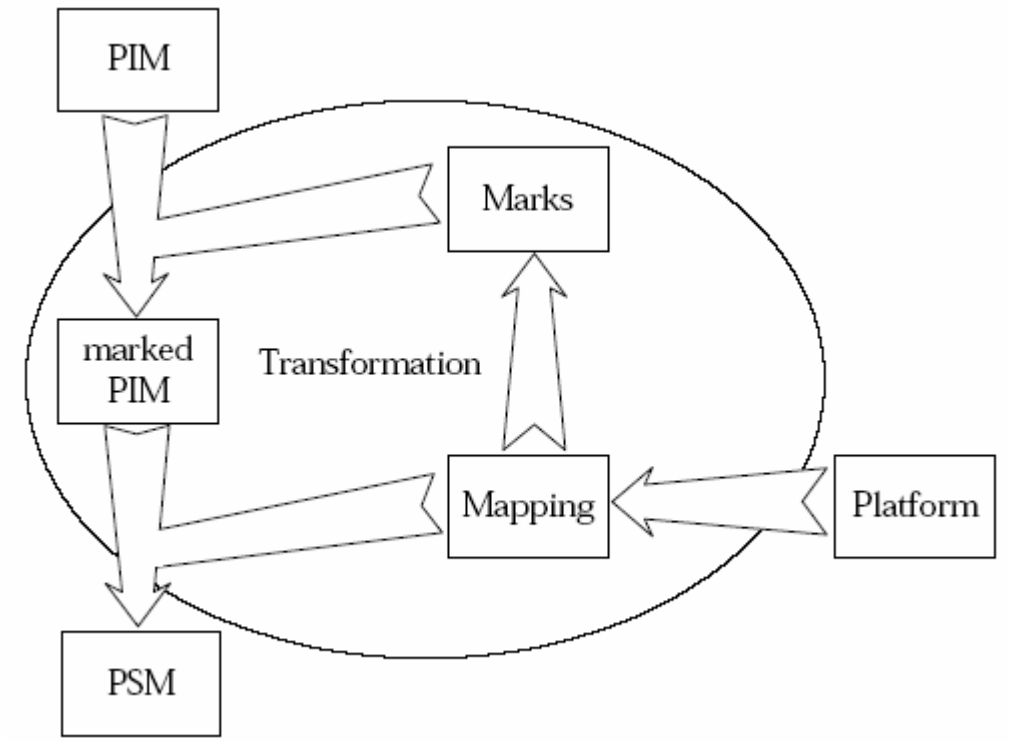


Figure 2.2: MDA general approach to transformation [6]

2.2.5.1 Model to model mappings

Model to model mappings involve transformation of any models built using PIM specifications to models built using PSM specifications. This idea can also be extended to

mapping of meta-models of PIM and PSM. These mappings also have guidelines on how to map different instances types of PIM to PSM. [5]

2.2.5.2 Model Elements Mappings

Model Elements mappings is a more specific case of model to model mappings, in which different model elements in PIM are marked to specify the particular way in which they have to be transformed to PSM. The marks placed on PIM belong to a particular transform hence they can be applied for transformations to a particular platform. [5]

Often both these types of mappings are combined. Because the model type mappings are solely based on the information available in the PIM, the marks cannot reflect the additional information that is to be used by the PSM. Due to the restrictions involved in which marks would suit a particular element in PIM, how the transformation has to be carried out is presented explicitly in model elements mappings.

2.2.6 Mode of Transformations

After mappings, transformations from PIM to PSM can be achieved both manually and automatically. Currently there are many automatic transformation tools available that can transform from PIM to PSM with the basic functionalities of PIM preserved. Additional functionalities of PIM can be added manually to the transformed model. Even when the PIM is not completely modeled, transformation can be made using automated tools. This allows the user to get an idea about the system to be developed beforehand and the flexibility to implement any changes that are required by the PIM. [5][1]

2.3 XML Metadata Interchange (XMI)

XML Metadata Interchange **(**XMI**)** is used in transferring metadata information via Extensible Markup Language (XML). [16] The models generated using XMI are in XML (Extensible Markup Language) representation. Thus XMI can work effectively with XML technologies. XMI being a part of MOF, can be used to link metadata at different levels of abstraction.[19] By using XMI**,** the UML model can be transformed from one tool to another tool or to another application for further development or improvisation.

2.4. Unified Modeling Language (UML) and Object Constraint Language (OCL)

UML is the language used to represent models involved in an application domain. It helps us to visualize and document models of software systems. In the MDA development process, the UML can be used to represent both PIM and PSM. One advantage of using UML is regardless of the methodology used to carry out analysis and design in the initial phases of the project, UML can be used to express the outcomes. UML 2.0 is the latest UML version. It also includes the Object Constraint language (OCL). [9]

OCL is used to describe expressions and constraints in Object Oriented models [1]. Thus it can add the necessary and additional information to the UML models. Since the expressions have mathematical foundations, they avoid any misinterpretations, allowing the model to convey precise details. Since OCL is strongly typed, the correctness of these expressions in the model can be checked during modeling before execution. AWSOME is based on UML and OCL. The AWL file serves to perform UML modeling. It also allows defining OCL like expressions, with which one can add precise details to the model.

2.5 AWSOME

2.5.1 AWSOME Meta-model

The meta-model of AWSOME is a subset of UML (Unified Modeling Language) [Section 2.4] that has OCL like expressions. It has a hierarchical structure. The AWSOME model consists of a set of classes which is composed of attributes, methods and a state based model [8]. “Underlying the Java implementation of AWSOME is a formal model that is the basis for correctness proofs and correctness-preserving transformations” [8].

2.5.2 AFIT Wide spectrum Language (AWL)

AWL is a formal specification language. It was used to support code generation from formal specifications. Like the AWSOME meta-model, the AWL is a wide spectrum language supporting imperative statement representation and declarative pre and post conditions.

The AWSOME Tool Box is a GUI present in the existing AWSOME system. The Tool Loader is present in the AWSOME Tool Box. Initially it is required to load the AWL file using the AWSOME Tool Loader. The AWL parser is developed using JavaCC, a Java based compiler-compiler. It is used to parse the AWL file. When the AWSOME tool is started the Abstract Syntax Tree (AST) is loaded. Generation of AWL file from the loaded AST is also supported. Thus any modifications made in the model can be saved in a form (AWL) that can be parsed in later for further changes.

2.5.3 Tool2009

The Tool2009 has all the independently developed tools that share the AWSOME AST.

The screenshot of Tool2009 is included in Figure 2.3

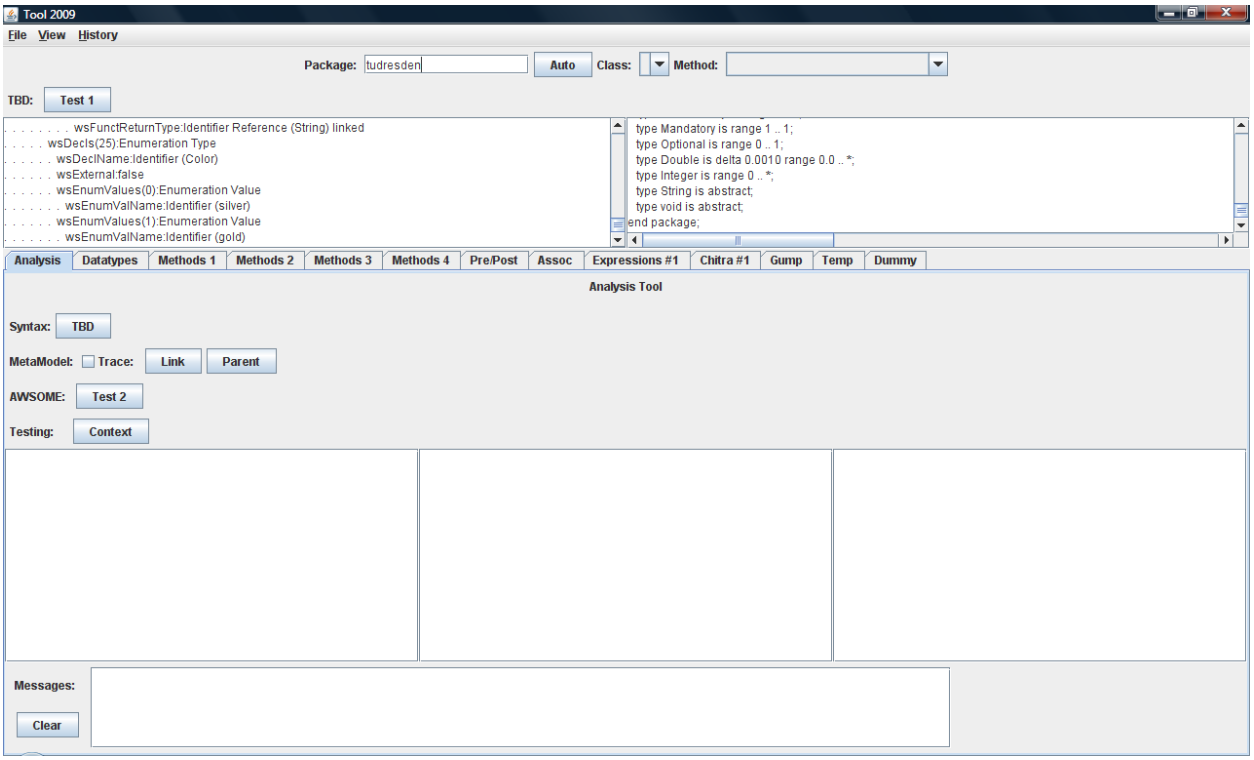


Figure 2.3: Tool2009 Screenshot

The top left panel has the outline of the loaded AWSOME AST and the top right panel has the parsed in AWL file. All the sub tools reside in separate tab panels. In each sub tool, several AWSOME transforms are performed. Tool2009 provides an easy way to combine and maintain the existing tools.

2.6 Tools with AWSOME

Some of the available tools that use the MDA context were considered to be made interoperable with AWSOME and automate the code generation. The tools are:

- ArgoUML
- EclipseUML 2008

- Delphia Object Modeler (DOM)
- Dresden OCL Toolkit

2.6.1. ArgoUML

ArgoUML [10] [11] is a powerful yet easy-to-use interactive, graphical software design environment that supports the design, development and documentation of object-oriented software applications. ArgoUML was conceived as a tool and environment for use in the analysis and design of object-oriented software systems. ArgoUML includes several novel features that address the identified cognitive needs of software designers. It is similar to many of the commercial CASE tools that are sold as tools for modeling software systems. ArgoUML has a number of very important distinctions from many of these tools, as described below:

- ArgoUML supports open standards extensively - UML, XMI, SVG, OCL and others. In this respect, ArgoUML is a few years ahead (almost five years after its initial introduction) of many commercial tools.
- ArgoUML is a 100% Java application. This allows ArgoUML to run on all platforms for which a reliable port of the Java2 platform is available.
- ArgoUML is an open source product. The availability of the source ensures that, a new generation of software designers and researchers now have a proven framework from which they can drive the development and evolution of CASE tools technologies.

ArgoUML is based directly on the UML 1.4 specification. The core model repository is an implementation of the Java Metadata Interface (JMI) which directly supports MOF and uses

the machine readable version of the UML 1.4 specification provided by the OMG. Furthermore, ArgoUML provides comprehensive support for OCL (the Object Constraint Language) and XMI (the XML Model Interchange format). ArgoUML supports XMI 1.0, 1.1, and 1.2 files which contain UML 1.3 and UML 1.4 models. To obtain the best compatibility with ArgoUML, it is recommended to export the models using UML 1.4 and XMI 1.1 or 1.2. The models may contain many objects (ModelElements) which form the complete UML description of the system. All ModelElements might be present on a diagram, but may not be required. Hence, the model stored in ArgoUML is independent of the contents of the diagrams. This may be explained by the possibility to generate programming code from the model. There is a way to only save the model information, which is by the menu "Tools" -> "Export as XMI..." This may be useful, *e.g.*, when generating programming code with an external tool that understands XMI. UML is itself an open standard and ArgoUML always tries to use open standards for all its interfaces. The following figure depicts a screenshot of ArgoUML

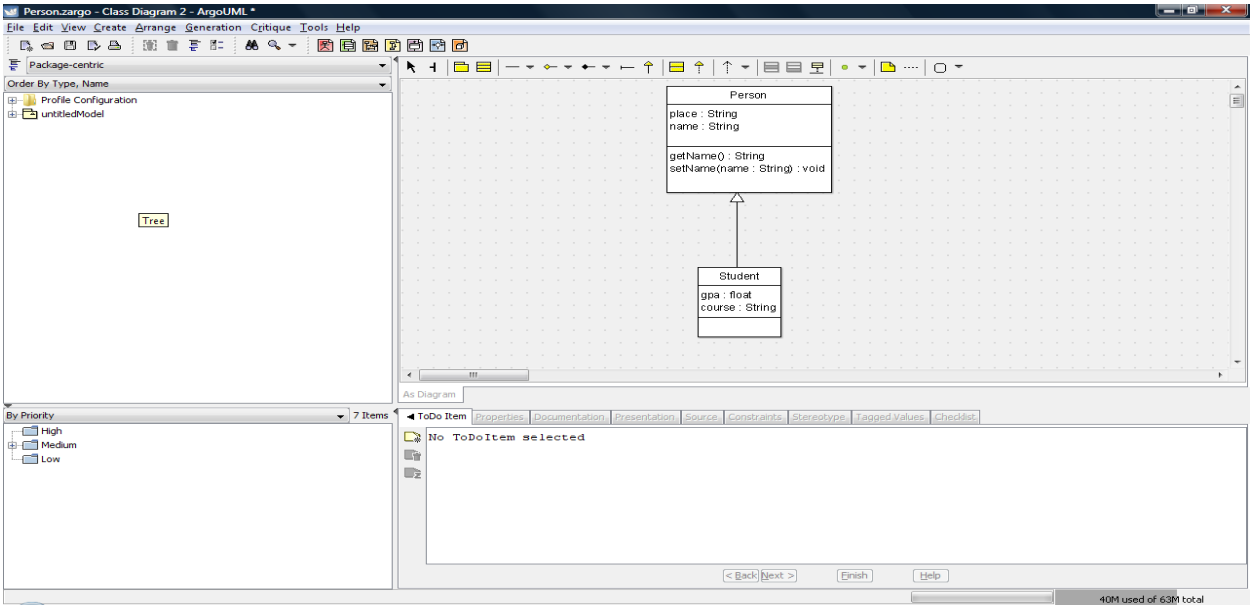


Figure 2.4: Screenshot of ArgoUML

XML Metadata Interchange (XMI) is the standard for saving the metadata that make up a particular UML model. The key advantage of adherence to open standards is that it permits easy inter-working between applications, and the ability to move from one application to another as necessary. In principle, this will allow you to take the model created in ArgoUML and import it into another tool. Generally, diagrams are drawn by selecting the model element desired and clicking in the diagram at the position required. Model elements that are already in the model, but not on a diagram, may be added to a diagram by selecting the model element in the explorer. The following file formats can be opened and loaded in ArgoUML:

- ArgoUML file (*.zargo, *.uml, *.xmi, *.xml, *.zip)
- ArgoUML compressed project file (*.zargo)
- ArgoUML project file (*.uml)
- XML Metadata Interchange (*.xmi)
- XML Metadata Interchange (*.xml)
- XMI compressed project file (*.zip)

2.6.2 EclipseUML 2008

The following section describes the architecture of EclipseUML [12]. EclipseUML architecture is composed of 4 stages which reduces complexity and opens new frontiers in traditional object oriented modeling. Developed on the revolutionary Omondo architecture, EclipseUML is built on the top of a Managed Object Format (MOF) and uses Ecore as opposed to traditional tools which use transformation layers.

- **Stage 1**: UML diagrams *(GEF)*

- **Stage 2**: UML Superstructure *(EclipseUML2 metamodel)*
- **Stage 3**: Model transformation *(EMF)*
- **Stage 4**: MOF

"EclipseUML 2008" uses a project meta-model composed by one or many diagrams. It is possible to create a full and complex model with EclipseUML 2008 and not create any Java element inside a Java project. This is important for the first modeling stage which certainly should be disassociated from the code. EclipseUML 2008 facilitates the drawing of UML 2.0 diagrams and saves them as a *.uml* file. The *.uml* file captures the data of the UML diagram in XMI format. No specific DTD or XML schema is given to the meta-model of the EclipseUML 2008. Figure 2.5 presents the meta-model of EclipseUML 2008, derived from the *.uml* files downloaded along with the EclipseUML 2008.

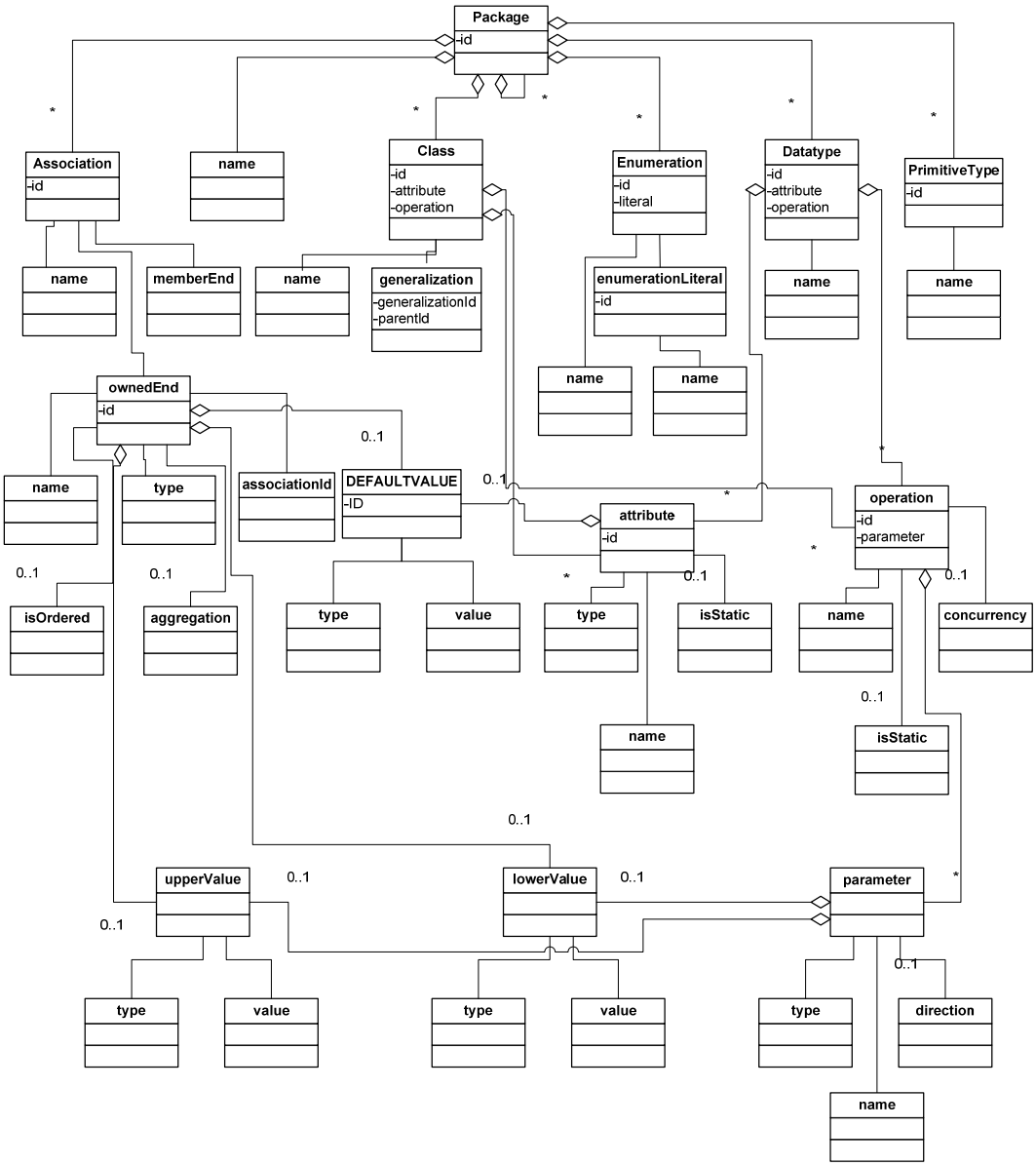


Figure 2.5: EclipseUML 2008 Meta-model

Eclipse UML is not an open source tool. After the trial period, the tool does not support the drawing of UML diagrams.

2.6.3. Delphia Object Modeler (DOM 3.2.6)

Delphia Object Modeler (DOM) is an object modeling tool. The diagrams are drawn in UML 1.4 version and it supports the import and export of XMI files. This tool does not have an XML schema or DTD file for its meta-model. Also there is no documentation or manual available for the tool. The following Figure 2.6 is a screenshot of DOM 3.2.6

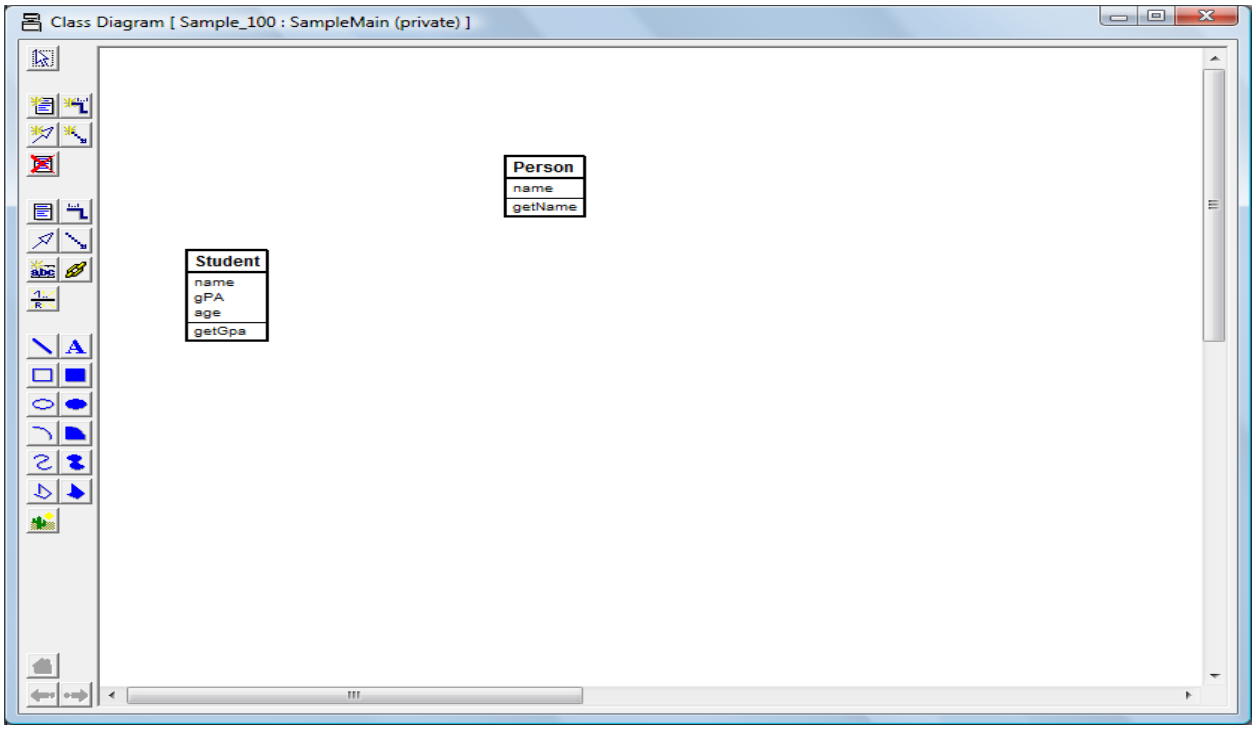


Figure 2.6: Screenshot of DOM 3.2.6

2.6.4 Dresden OCL Toolkit

The Dresden OCL Toolkit [14] is all about the Object Constraint Language (OCL). It extends the UML's graphical notation with the possibility of adding more formally defined textual constraints on method invocations and on class structures as a whole. Many aspects of a model that cannot be expressed adequately with the graphical notation alone find their representation in OCL constraints. Dresden OCL2 for Eclipse provides three tools:

- OCL2 Parser,
- OCL2 Interpreter and;
- OCL2 to Java code generator

OCL2 Parser can be used to open text files and parse them as OCL2 constraints. The constraints parsed by the parser are added to the model in the toolkit on which they have been defined. OCL2 Interpreter can be used to interpret already imported OCL2 constraints. Invariants, pre- and post-conditions can be interpreted. Initial and let expressions, definitions and body expressions can be used to enrich and prepare the OCL2 interpretation. OCL2Java is a code generator which can be used to generate Java code for imported OCL2 constraints. The code generator uses the aspect-oriented language AspectJ to instrument the generated code into the provided Java code for the imported model used for the code generation. The code generator supports the generation of code for invariants, pre- and post-conditions, definitions, initial and defined values, body and let expressions. Thus the Dresden OCL Toolkit facilitates the generation of AspectJ code by loading the UML 2.0 model and OCL 2.0 constraints file. The tool does not allow the generation of OCL but it can read the OCL file.

2.7 Other Research Efforts

One of the frameworks aimed at interoperability of tools under AMMA (ATLAS Model Management Architecture) context is DUALLY [13]. This framework creates interoperability between Architecture Description Languages (ADLs) and between an ADL and UML. DUALLY is based on the concept of transforming semantically equivalent concepts of one architectural model in to another architectural model. Each architectural model conforms to

a meta-model (architectural language). Therefore meta-transformation can also be made between two meta-models. Thus DUALLY framework can be implemented at both meta-modeling and modeling levels and permits model to model transformations.

DUALLY is implemented as an Eclipse plugin. Since Eclipse platform facilitates import/export using XMI, any tool which can import/export from/to Eclipse EMF using XMI can also use DUALLY. Tools that do not facilitate import/export from/to Eclipse EMF using XMI are subjected to initial step of translating their host format in to model-driven format. DUALLY also facilitates this initial step. DUALLY can be extended to integrate with MDA technologies used in Eclipse platform.

2.8 Summary

This chapter discussed MDA, AWSOME, UML, XMI and tools that were considered to interface with AWSOME. The next chapter will analyze each tool in detail and finalize upon which tools will be interfaced with AWSOME to demonstrate the interoperability between tools.

3. REQUIREMENT ANALYSIS

3.1 Introduction

This chapter analyzes in depth about the different tools considered for possible interfacing with AWSOME. The main focus of the research is on demonstrating the interoperability between AWSOME and other available tools. AWSOME can be interfaced with the available tools (discussed in Chapter 2) in the following two ways.

- To make AWSOME interoperable with available UML tools
    1. AWSOME should be able to parse a UML file and create the AWSOME AST from it.
    2. AWSOME should be able to generate a UML file from the loaded AWSOME AST.
- To make AWSOME interoperable with available OCL tools
    1. AWSOME tool should be able to generate OCL constraints from the loaded AWSOME AST.
    2. AWSOME tool should be able to parse an OCL file and add the constraints to the AWSOME AST.

The following sections describe the advantages and disadvantages of each tool to be interfaced with AWSOME.

3.2 Comparison of Candidate Tools

3.2.1 Dresden OCL Toolkit

The Dresden OCL Toolkit can parse a UML 2.0 file into the domain model and an OCL constraints file into the domain model constraints. From the loaded domain model and constraints, AspectJ code can be generated. One of the disadvantages with the Dresden OCL

Toolkit is it cannot generate the OCL file. The OCL file has to be written in a text editor and loaded into the tool.

3.2.2 ArgoUML

As discussed in Chapter 2, ArgoUML supports XMI 1.0, 1.1, and 1.2 files which contain UML 1.3 and UML 1.4 models. The disadvantage with using ArgoUML is it cannot support the UML 2.0 files. Dresden OCL Toolkit is the only available tool to parse OCL files that can support only UML 2.0 models. The UML 1.4 model generated from the ArgoUML file cannot be loaded into the Dresden OCL Toolkit. Since ArgoUML files cannot be supported by Dresden OCL Toolkit, it is not considered to be interfaced with AWSOME.

3.2.3 Delphia Object Modeler (DOM 3.2.6)

The DOM tool supports UML 1.4 diagrams and importing and exporting XMI files. But since the tool does not have an XML schema or DTD file for its meta-model and no documentation or manual describing the working of the tool, it cannot be interfaced with AWSOME.

3.2.4 EclipseUML 2008

EclipseUML 2008 facilitates the drawing of UML 2.0 diagrams. It can load a UML 2.0 model and allows changes to be made. The disadvantages of EclipseUML 2008 are there is no XML schema file or DTD file available for its meta-model and, as it is not an open source tool. After the trial period the UML 2.0 file can be viewed as an XMI source file, but not as a

diagram. The advantage of this tool is that it can be interfaced with any OCL tool that supports UML 2.0 models.

3.3 Tools selected to be interfaced with AWSOME

Consider the following comparison table for the different tools discussed so far.

Table 3.1: Tool Comparison

	Read			Write			Other
Tool	**UML 1.4**	**UML 2.0**	**OCL**	**UML 1.4**	**UML 2.0**	**OCL**	
Dresden OCL Toolkit	Yes	Yes	Yes	No	No	No	produce AspectJ code
ArgoUML	Yes	No	No	Yes	No	No	produce C++, C#, Java and PHP
Delphia Object Modeler	Yes	No	No	Yes	No	No	produce UML 1.4 diagrams
EclipseUML 2008	No	Yes	No	No	Yes	No	produce UML 2.0 diagrams

EclipseUML 2008 and the Dresden OCL Toolkit were chosen to be interfaced with AWSOME.

The following reasons derived from the Table 3.1 led to choosing these tools.

1. The UML 2.0 model is preferred over any UML 1.x because of the following additional features in UML 2.0:
    a. In UML every building block like classes, objects, etc. are classifiers. In UML 2.0 a set of classifiers can be nested or a behavior can be embedded in the classifier. This feature facilitates the developing of complex behaviors in simple models. [9]
    b. UML 2.0 provides an improved behavioral model as all the behavioral models are developed from the fundamental definition of a behavior.[9]

c. UML 2.0 has enhanced relationships between behavioral and structural models by using nested classifiers.[9]

2. Dresden OCL Toolkit is the only tool supporting OCL 2.0.
3. Dresden OCL Toolkit is the only tool that can do something unique with the UML 2.0 domain model and OCL 2.0 to generate AspectJ code.
4. EclipseUML 2008 is the only tool to read and write UML 2.0.
5. EclipseUML 2008 provides a graphic editor to draw and make changes to UML 2.0 file.
6. Both ArgoUML and Delphia Object Modeler cannot support UML 2.0 or OCL 2.0.
7. Dresden OCL Toolkit can load a UML 2.0 model and OCL 2.0 constraints file to generate the AspectJ code. Dresden OCL Toolkit can load the UML 2.0 model generated by EclipseUML 2008.
8. The above mentioned tools did not have XML Schema or DTD files and did not support OCL generation. So these two factors did not play a major role in tools selection.

3.4 AWSOME interfacing requirements

After tools selection, it is clearly evident that the next step would be to interface these tools with AWSOME. This involves making AWSOME compatible to support UML 2.0 and OCL files.

An example of a UML 2.0 file in XMI format is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<uml:Package xmi:version="2.1" xmlns:xmi="http://schema.omg.org/spec/XMI/2.1"
xmlns:uml="http://www.eclipse.org/uml2/2.1.0/UML"
xmi:id="_6pUsMMHhEd2UNuYhQ_ApoA" name="tudresden">
  <packagedElement xmi:type="uml:Package" xmi:id="_AsKYYMHiEd2UNuYhQ_ApoA"
name="ocl20">
    <packagedElement xmi:type="uml:Package" xmi:id="_BlraUMHiEd2UNuYhQ_ApoA"
name="pivot">
```

```
    <packagedElement xmi:type="uml:Package" xmi:id="_Cff-QMHiEd2UNuYhQ_ApoA"
name="examples">
      <packagedElement xmi:type="uml:Package" xmi:id="_DmIPIMHiEd2UNuYhQ_ApoA"
name="simple">
        <packagedElement xmi:type="uml:Class" xmi:id="_J0lVcMHiEd2UNuYhQ_ApoA"
name="Person">
          <ownedAttribute xmi:id="_MIZyMMHiEd2UNuYhQ_ApoA" name="name"
type="_IX5VAMHiEd2UNuYhQ_ApoA">
            <defaultValue xmi:type="uml:LiteralString" xmi:id="_Mjqx4MHiEd2UNuYhQ_ApoA"
value="y"/>
          </ownedAttribute>
          <ownedAttribute xmi:id="_Ppaa0MHiEd2UNuYhQ_ApoA" name="age"
type="_GxO5MMHiEd2UNuYhQ_ApoA"/>
        </packagedElement>
        <packagedElement xmi:type="uml:Class" xmi:id="_Tagt4MHiEd2UNuYhQ_ApoA"
name="Student">
          <generalization xmi:id="_U2mRYMHiEd2UNuYhQ_ApoA"
general="_J0lVcMHiEd2UNuYhQ_ApoA"/>
        </packagedElement>
        <packagedElement xmi:type="uml:Class" xmi:id="_XPL5YMHiEd2UNuYhQ_ApoA"
name="Professor">
          <generalization xmi:id="_XrWQ8MHiEd2UNuYhQ_ApoA"
general="_J0lVcMHiEd2UNuYhQ_ApoA"/>
        </packagedElement>
      </packagedElement>
    </packagedElement>
  </packagedElement>
  <packagedElement xmi:type="uml:PrimitiveType" xmi:id="_GxO5MMHiEd2UNuYhQ_ApoA"
name="int"/>
  <packagedElement xmi:type="uml:PrimitiveType" xmi:id="_IX5VAMHiEd2UNuYhQ_ApoA"
name="String"/>
</uml:Package>
```

A simple example of an OCL file is as follows:

```
package tudresden::ocl20::pivot::examples::simple

context Person::name
init: 'y'
endpackage
```

In order to interface EclipseUML 2008 with AWSOME, the following additional features should be added to the AWSOME tool

a. A UML Parser to parse the UML 2.0 file and create an AWSOME AST in Tool 2009.
b. A UML generator to generate the UML 2.0 file from the loaded AWSOME AST.

Using these two tools, AWSOME can be made interoperable with EclipseUML 2008.

In order to interface Dresden OCL Toolkit with AWSOME, the following additional features should be added to the AWSOME tool

a. An OCL Parser to parse the OCL 2.0 file and load the constraints into the AWSOME AST.
b. An OCL generator to generate OCL constraints from the loaded AWSOME AST.

Using these two tools, AWSOME can be made interoperable with Dresden OCL Toolkit. The additional task is to map the functionalities between AWSOME AST and UML 2.0 and between AWSOME AST and OCL 2.0 file. Since there are differences in the functionalities of these tools, certain assumptions and restrictions are imposed on these files. For an example, for every model element in the UML 2.0 file there is a unique object id called the XMI id, whereas AWSOME does not have such a unique id associated with its model elements. So this unique id has to be added to the AWSOME meta-model to facilitate the mapping between UML 2.0 and AWSOME AST. Such mapping details are discussed in detail in Chapter 4 under Sections 4.3, 4.4, 4.5, 4.6 and 4.7.

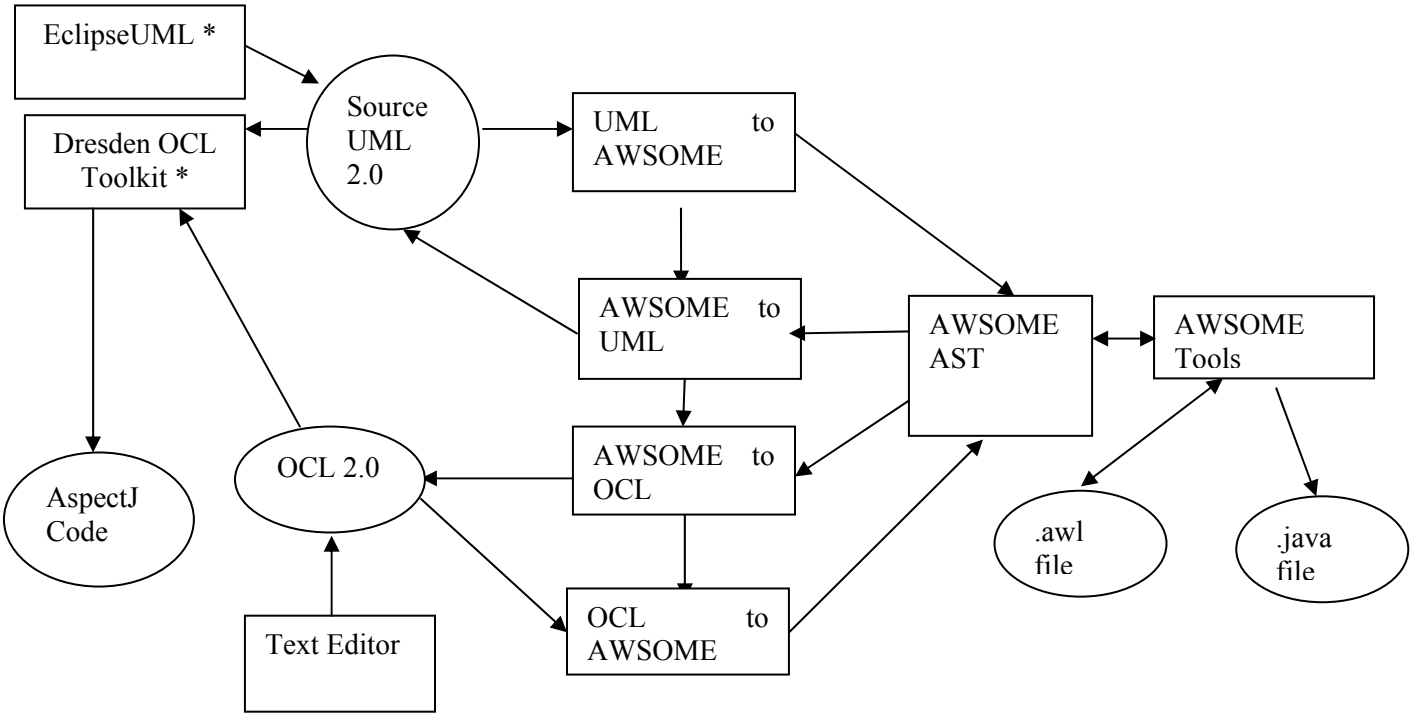
3.5 Summary

This chapter discusses about the advantages and disadvantages of the tools considered to be interfaced with AWSOME. It was determined to be feasible to interface AWSOME with EclipseUML 2008 and Dresden OCL Toolkit. Any UML 2.0 model generated by AWSOME should be able to load in EclipseUML 2008 and any UML 2.0 model should be able to be parsed in AWSOME. Similarly any OCL 2.0 file generated by AWSOME should be able to be loaded in Dresden OCL Toolkit. Since there is no DTD or XML Schema file available for EclipseUML 2008 and Dresden OCL Toolkit, examples should be used to demonstrate the interoperability. Also there are some design decisions yet to be made before interfacing these tools with AWSOME. The following Chapter 4 will discuss about the design in detail.

4. DESIGN

4.1 Introduction

The overall system design is presented in Figure 4.1. The design shows how AWSOME can be made interoperable with other tools. The EclipseUML can be used to generate UML 2.0 files. The Dresden OCL Toolkit can be made interoperable with any UML 2.0 file and OCL 2.0 file. The UML 2.0 file is loaded into the domain model and the OCL 2.0 file into the OCL Expressions of the Dresden OCL Toolkit. From the loaded domain model and OCL constraints, the AspectJ code is generated.



* Any tool that can read /write UML 2.0 and OCL 2.0

Figure 4.1: Overall System Design

The UML 2.0 and OCL 2.0 are linked with AWSOME through the following four transformation or translation tools.

1. UML to AWSOME: This tool is used to parse the UML file and load it into the AWSOME AST. The design detail of the tool is discussed in detail under Section 4.2
2. AWSOME to UML: This tool is used to generate the UML file from the AWSOME AST. The tool is discussed in detail under Section 4.3.
3. AWSOME to OCL: This tool is used to generate the OCL file from the AWSOME AST. The tool is discussed in detail under Section 4.7.
4. OCL to AWSOME: This tool is considered to be beyond the scope of this thesis due to the lack of time and sufficient background in parser theory. Also the OCL 2.0 syntax is not clearly defined.

Using the AWSOME tools such as Tool to AWL and Tool to Java, the .awl file and .java file can be generated from the AWSOME AST. The .awl file represents the AST in AFIT Wide spectrum Language (AWL).

4.2 UML to AWSOME

There are two design approaches in making AWSOME interoperable with EclipseUML. The first design approach is to parse the UML 2.0 file using a DOM Parser and build a Document Tree. The document tree can be stored as an EclipseUML meta-model of the UML Objects. Then the UML objects in the Eclipse meta-model can be converted to AWSOME objects in the AWSOME AST. The design is presented in Figure 4.2 (a).

The second approach would be to parse the UML 2.0 file using the UMLParser (discussed in Section 4.2.1) and build a Document Tree. The Document tree can be translated directly into the AWSOME AST using a translator. The existing AWL writer which is integrated with Tool to AWL can be used to generate the .awl file.

Thus in both the approaches the AWSOME AST can be built from the UML 2.0 file. Changes to the AST can be done with existing AWSOME tools.

The second approach was preferred over the first approach. Directly building the AWSOME AST from the parsed in Document tree is easier than to translate the Eclipse meta-model to the AWSOME AST.

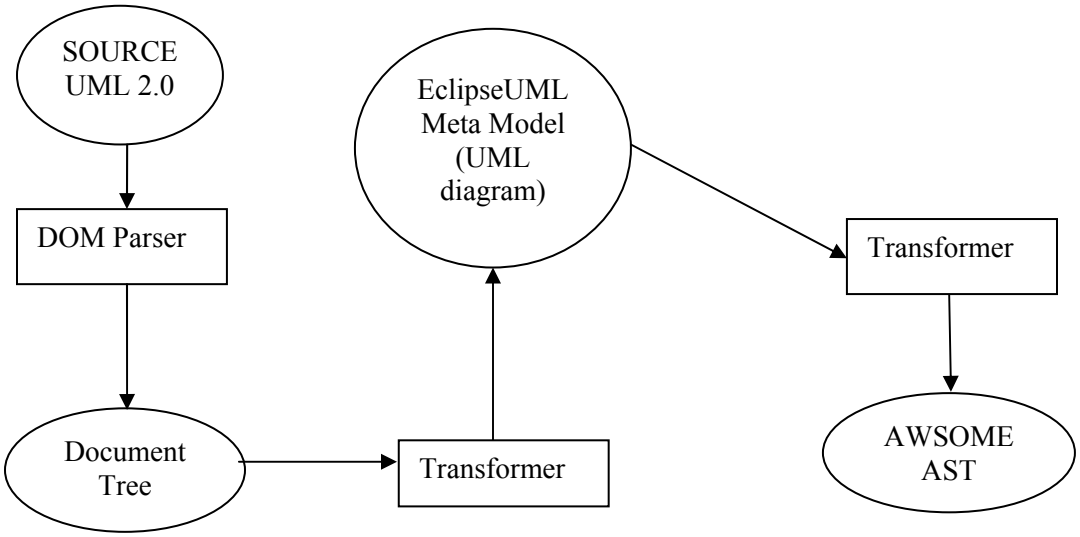


Figure 4.2 (a): UML to AWSOME – First Approach

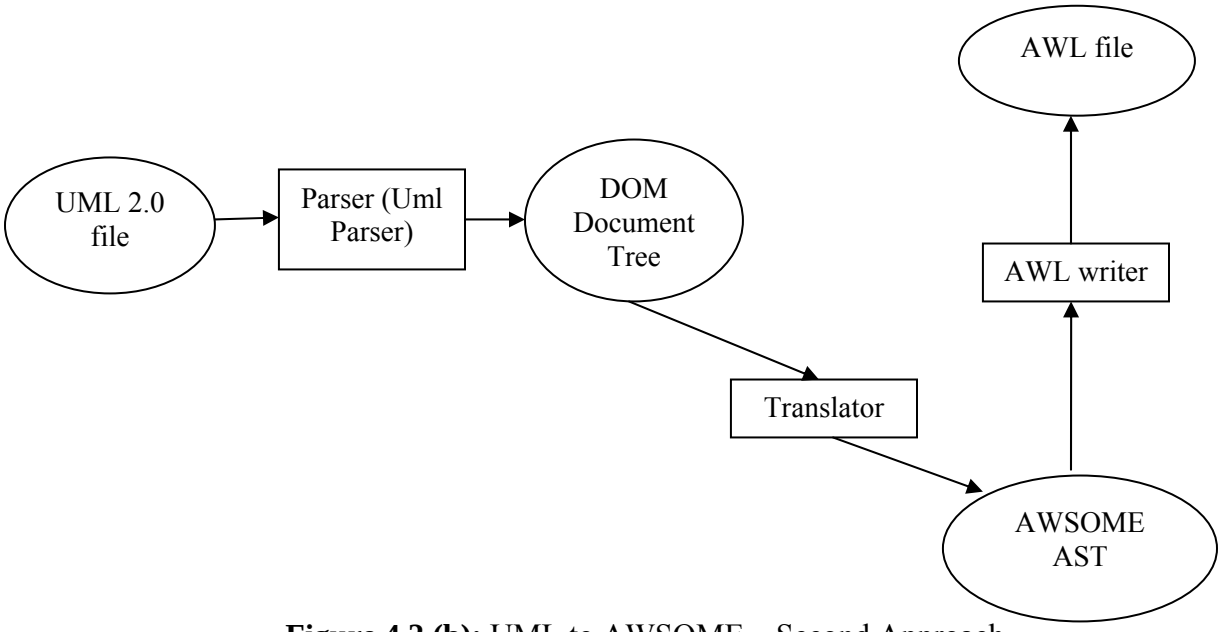


Figure 4.2 (b): UML to AWSOME – Second Approach

4.2.1 UmlParser Design

The **UmlParser** is a tool integrated with the ParseXMI button of Chitra #1 panel in Tool2009. The purpose of this tool is to parse the input UML 2.0 (XMI format) file and load it into the DOM tree. The DOM tree is translated into an AWSOME AST. The UmlParser parses the UML 2.0 file in three passes.

In the first pass the UML Primitive Types are parsed. Since the primitive types can be referred to by any attribute or parameter in the lower level, they are parsed at first pass and loaded as WsDecl in WsPackage of the AWSOME AST. The xmi id and primitive type is stored as a key-value pair in a Hash Map. Any attribute or parameter type can refer to the primitive type by storing the primitive type's xmi id. In order to obtain the primitive type corresponding to that xmi id and store it as a type in AWSOME, the hash map is populated in the first pass.

The second pass parses the following:

- classes,
- data types,
- enumeration types,
- associations and
- association classes.

The xmi id and name of each of the objects is populated in their respective hash maps.

The third pass parses the following:

- **In Class or Data Types:** attributes, default value of attributes, operations, function's return parameters, lower and upper bound for parameters and generalizations
- **In Associations:** association end and association end multiplicity i.e the lower and upper bounds for each association end.

- **In Association classes:** attributes, default value of attributes, operations, function's return parameters, lower and upper bound for parameters, association end and association end multiplicity.

Thus each pass parses the UML 2.0 file hierarchically from higher level objects to lower level objects.

4.2.2 Mapping UML 2.0 (.uml file in XMI format) to AWSOME

There are some functionalities in UML 2.0 that cannot be effectively represented in AWSOME. It was decided to support the common functionalities that can be represented in both AWSOME and UML 2.0. Some of the UML 2.0 functionalities are supported in AWSOME by doing some changes to the AWSOME AST.

1. All the xmi:ids in the .uml file are stored in attribute xmiId which is added to WsObject in AWSOME.
2. The attribute wsPrimitiveType1 was added to WsObject in AWSOME to store the UML Class and UML Datatype in AWSOME. The UML Class is stored in AWSOME as a WsClass with wsPrimitiveType1 flag set to false. The Uml Data type is stored as a WsClass with the wsPrimitiveType1 flag set to true.
3. In AWSOME the Boolean data type is predefined as an enumeration type, consisting of the values True or False. Hence a Boolean primitive type from a .uml file is not stored in the wsDecls of the AWSOME AST. It is added to a hash map containing all the primitive types from the .uml file.
4. If the default value of an attribute in the .uml file is of type literal string, in AWSOME it is stored by enclosing it in double quotes.

5. The static attributes in the .uml file can be stored by setting the attribute wsStatic to true in the WsAttribute of the AWSOME AST. But Tool To AWL and AWL Parser do not support static attributes specified in .awl file. Thus this data about static attribute is lost if the AWSOME meta-model is written to AWL.
6. If the ownedParameter tag in the .uml file has the direction specified as return then the parameter is stored as the attribute wsFunctReturnType in WsFunction, otherwise it is added to the vector wsSubProgFormals in WsFunction in AWSOME AST.
7. Since AWSOME cannot store lower and upper bounds for parameters, the upper / lower bounds are stored in a new WsIntegerType or WsRealType depending on whether the parameter type is integer or real. A new type name is generated on the fly and stored in wsDecls of the AWSOME AST. This new type name is also stored in the attribute wsParameterType in WsParameter. If the type of the parameter is anything other than integer or real then AWSOME does not store the lower and upper bounds for it. A better approach might be to use WsDerivedType in the future.
8. Along with the UML primitive types, the following possible association multiplicity types are also added to wsDecls in AWSOME.
    1. Mandatory 1..1
    2. OneToMany 1..*
    3. ZeroToMany 0..*
    4. Optional 0..1

    The associationEnd with lower and upper bound is checked for one of the pre defined association multiplicity types in wsDecls. If the multiplicity end is one of the pre defined multiplicity types, then the corresponding multiplicity name is stored in

wsAssocEndMultiplicity. If the multiplicity end is not one among the pre defined multiplicity types, then a new type name of type WsIntegerType is generated on the fly and stored in wsDecls of AWSOME AST.

9. The wsPrimitiveType1 attribute is set to true for any primitive type that is parsed from the .uml file. It is set to false for all user generated types that were generated to work around the inconsistencies between AWSOME and UML 2.0.
10. The UML upperValue tag with value as "*" is stored in AWSOME AST as null. Hence the xmi:id of upperValue tag cannot be stored in AWSOME AST.
11. The ownedOperation tag in the .uml file is checked to determine whether the return parameter is present or not. If the return parameter is present, the operation is stored in the AWSOME AST as WsFunction else it is stored as WsProcedure.
12. Since the function return value is stored as the attribute wsFunctReturnType in WsFunction, the xmi id of function return value cannot be stored in the AWSOME AST.
13. If an ownedOperation tag in the .uml file has concurrency set to "guarded", the concurrency value cannot be stored in the AWSOME AST.
14. If an ownedOperation tag in the .uml file has concurrency set to "concurrent", the concurrency value is stored by setting the attribute wsPrimitiveType1 to true in the WsFunction/ WsProcedure.
15. If an ownedOperation tag in the .uml file has concurrency set to "sequential", the concurrency value is stored by setting the attribute wsPrimitiveType1 to false in the WsFunction/ WsProcedure.
16. The association name in the .uml file can have space in it. Since AWSOME does not allow a name with spaces, the space is replaced by an underscore and stored in the AST.

For example: The association name “LoyaltyProgram ProgramPartners” in the .uml file is stored as “LoyaltyProgram_ProgramPartners” in the AWSOME AST.

17. The memberEnd value of a uml:Association or uml:AssociationClass tag is not stored in the AWSOME AST.
18. The ordered association end in the .uml file is stored by setting the wsOrder attribute to true in the AWSOME AST. The AWL generator can generate an .awl file for an ordered association end. But the AWL parser cannot parse that “ordered” string in the .awl file.
19. The default value of association ends in the .uml file is not stored in the AWSOME AST.
20. The xmi id of the generalization tag in the .uml file is not stored in the AWSOME AST.
21. The WsRealType in the AWSOME AST is always stored with a delta or digits value. Hence the UML primitive type equivalent WsRealType is stored in the AWSOME AST with a delta value as 0.001.
22. If the .uml file with multiple packages is parsed and loaded in Tool2009, only the top most package is viewed in the package JTextField of Tool2009. This is because AWSOME supports viewing a package at only one level in Tool2009. Hence the classes, data types and other objects in lower level packages cannot be viewed in Tool2009. As a part of future efforts, viewing multiple packages should be incorporated in Tool2009.
23. Association objects in the .uml file can be stored in the AWSOME AST. But Dresden OCL Toolkit does not parse association objects completely. The role specified in the ownedEnd in the Association Class is parsed as a property of a class to which the role refers. The ownedOperation and ownedAttribute of an association class is not parsed at all. So any test cases that are parsed into Dresden OCL Toolkit will not have association objects in it.

4.3 AWSOME to UML

The .uml file can be generated by clicking the GenerateXMI button in the Chitra #1 panel of Tool2009. The ToXMIVistor integrated with the GenerateXMI button visits each node in AWSOME to generate the corresponding .uml file in UML 2.0 format.

There are some functionalities in the AWSOME AST that cannot be effectively represented in UML 2.0. It was decided to support the common functionalities that can be represented in both AWSOME and UML 2.0. No change is done to UML 2.0 to support the difference in UML 2.0 and AWSOME.

4.3.1 Mapping from AWSOME to UML 2.0 (.uml file in XMI format)

1. If the wsPrimitiveType1 attribute in a WsClass of the AWSOME AST is set to true then the value of xmi:type of the packagedElement tag in the .uml file is set as uml:DataType, else it is set as uml:Class.
2. The Boolean data type stored in the AWSOME AST is represented in the .uml file with a new xmi:id.
3. The default value of a WsAttribute with type mentioned as WsLiteralString is stored in the .uml file without the double quotes enclosing the value.
4. The upperValue tag and lowerValue tag for association ends in the .uml file are generated by checking the multiplicity type associated with the association end in the AWSOME AST. From the lower and upper bound value obtained from the wsDecls of the multiplicity type in AWSOME AST, the value is stored in the upperValue tag and lowerValue tag in the .uml file.

5. If the value of wsPrimitiveTypeFlag1 attribute of WsDataType is true , the data type is stored in the .uml file in the uml:PrimitiveType tag. Otherwise the WsDataType is ignored as it is a user generated type.
6. AWSOME can store the upper and lower bounds of WsParameter as WsRealType. But EclipseUML does not support the upper and lower bounds to be of real type. So while generating the .uml file from AWSOME, the upper and lower values of type real in WsParameter will be truncated and generated as uml:LiteralInteger type.
7. If the upper bound value of WsAssocEnd or WsParameter is null in the AWSOME AST then the .uml file is generated with upperValue tag having a value of "*". A new xmi:id is generated for that tag.
8. The wsPrimitiveType1 attribute of WsFunction/ WsProcedure is checked to write the value of concurrency in the ownedOperation tag in the .uml file. If the wsPrimitiveType1 is set to true then concurrency value is set to be "concurrency". If the wsPrimitiveType1 is set to false then it need not be specified in the .uml file, as the concurrency value is assumed to be sequential by default in the .uml file.
9. The association name in the AWSOME AST is stored in the .uml file with all the underscores replaced by space.
10. The xmi ids of WsAssocEnds of a WsAssociation in the AWSOME AST are combined together to form the memberEnd value in the ownedEnd tag of the .uml file.
11. The xmi:id for generalization tag is generated in AWSOME and stored in the .uml file.
12. The delta value mentioned for WsRealType is ignored when the .uml file is generated.
13. Since EclipseUML has no container types specified, container types will not be generated in the .uml file from the AWSOME AST.

The following table represents the overall mapping between AWSOME and UML 2.0.

Table 4.1: Overall mapping between AWSOME and UML 2.0

UML 2.0		**AWSOME**	
UML Object	**Attribute**	**WsClasses**	**Field Name**
<every object>	xmi:id	WsObject	xmiId
Uml:Package	name	WsPackage	wsArtName
Uml:PrimitiveType	name	WsDeclaration	wsDeclName
Uml:Class	name	WsClass	wsDeclName
Uml:Class	isAbstract	WsClass	wsClassAbstract
ownedAttribute		WsClass	wsClassDataComponents[WsAttribute]
ownedAttribute	name	WsAttribute	WsDataObject.wsDeclName
ownedAttribute	type	WsAttribute	WsDataObject.wsDataObjectType
ownedAttribute	isStatic	WsAttribute	wsStatic
ownedDefaultValue	value	WsAttribute	WsDataObject.wsDataObjectValue
ownedOperation		WsClass	wsClassOperations[WsMethod]
ownedOperation	name	WsFunction/WsProcedure	wsDeclName
ownedOperation	isStatic	WsMethod	wsClassMethod
ownedOperation	concurrency	WsFunction/WsProcedure	wsPrimitiveType1
ownedParameter		WsFunction/WsProcedure	wsFunctReturnType/wsSubProgFormals
ownedParameter	name	WsParameter	wsParameterName
ownedParameter	type	WsParameter	wsParameterType
enumeration	name	WsEnumeration	wsDeclName
ownedLiteral		WsEnumeration	wsEnumValues[WsEnumerationValue]
ownedLiteral	name	WsEnumerationValue	wsEnumValName
association	name	WsAssociation	wsDeclName
ownedEnd		WsAssociation	wsAssociationEnds[WsWssociationEnd]

ownedEnd	name	WsAssociationEnd	wsAssocEndRole
ownedEnd	type	WsAssociationEnd	wsAssocEndClass
ownedEnd	isOrdered	wsAssociationEnd	wsOrder
ownedEnd	aggregation	wsAssociationEnd	wsAggregate
ownedEnd	lower/upper Value		
defaultValue			
associationClass	name	WsAssocObject	wsDeclName
ownedAttribute		WsAssocObject	wsAssocObjectComponents[WsAttribute]
ownedOperation		WsAssocObject	wsAssocObjectOperations[WsMethod]
ownedEnd		WsAssocObject	wsAssocObjectEnds[WsAssociationEnd]
ownedEnd	aggregation	WsAssociationEnd	isAggregate
generalization	general	WsClass	wsClassSuperClass

4.4 Restrictions and Assumptions in UML

There are some functionalities in UML 2.0 that are not supported by AWSOME. In order to support the generation of the AWSOME AST from a UML 2.0 file, some restrictions are applied to the UML 2.0 file. Depending on the data in the UML 2.0 file there are also some assumptions made before converting it to an AWSOME AST. The restrictions and assumptions made in UML are summarized below.

1. AWSOME does not support interfaces. But interfaces can be mapped to pure abstract classes with all abstract methods. The only drawback in mapping interfaces to abstract classes in AWSOME is that there could be objects that implement multiple interfaces or inherit an object and implement an interface. But in AWSOME multiple inheritance is

not supported. So the UML 2.0 input files are restricted not to include inheriting an interface or multiple interfaces.

2. The UML 2.0 input file has all the UML primitive types declared in the top level package. Any primitive type will be searched for only in the top level package while generating the .uml file from AWSOME.
3. All UML classes and data types should be declared in one package within the bottom or current package. Any attribute with type being a class is searched in its package only.
4. The UML:Attribute tag in the UML 2.0 file is assumed to be a variable and stored in the AWSOME AST as WSAttribute with typeDO set as "variable".
5. If the upperValue tag has no value attribute in the .uml file then the upper bound will not be set in AWSOME, which means upper bound is "*".
6. If the lowerValue tag has no value attribute in the .uml file then lower bound is set to 0 in AWSOME
7. If there is no upperValue tag present in the .uml file then upper bound is assumed to be "1" in AWSOME.
8. If there is no lowerValue tag present in the .uml file then lower bound is assumed to be "1" in AWSOME.
9. Since all UML multiplicity type names and parameter type names are stored with a suffix "_sub" or "_mult" in AWSOME, it is assumed that the UML primitive type names do not have the substring "_sub" or "_mult" in it.
10. The UML primitive types named as "int" or "Integer" (case insensitive) are stored as WsIntegerType in AWSOME.

11. The UML primitive types named as “double” or “float” or “real” or “money” (case insensitive) are stored as WsRealType in AWSOME.
12. Any other UML primitive type is stored as WsAbstractType in AWSOME.
13. In the UML 2.0 file, the ownedEnd of an association has aggregation set to shared or composite. If aggregation is shared, then the association end is assumed to be parent and wsAggregate is set to true in AWSOME. If aggregation is composite, then the ownedEnd is considered to be child and wsAggregate is set to false in AWSOME.

4.5 Restrictions and Assumptions in AWSOME

There are some functionalities in AWSOME that are not supported by UML 2.0. In order to support the generation of the UML 2.0 file from an AWSOME AST, some restrictions are applied to the AWSOME AST. Depending on the nodes in the AWSOME AST file there are also some assumptions made before converting it to UML 2.0. The restrictions and assumptions made in AWSOME are summarized below.

1. WsContainerFormer and Array types in AWSOME are not supported by UML 2.0 and OCL. So in an AWSOME AST, set formers and array types will be ignored and not converted to equivalent elements in the UML 2.0 file.
2. For any object that does not have an XMI id in the AWSOME AST, a new XMI id will be generated and stored in the UML 2.0 file.
3. The upper and lower bounds of integer and real data types in the AWSOME AST will not be stored in the UML 2.0 file. They will be converted to OCL constraints when the OCL 2.0 file is generated from the AWSOME AST.

4. There is no object in UML 2.0 equivalent to a WsAccess node in AWSOME. So the WsAccess Node in an AWSOME AST will not be stored in the UML 2.0 file.

4.6 AWSOME to OCL

The Dresden OCL Toolkit can generate AspectJ code by loading a UML 2.0 model (.uml file) and OCL 2.0 constraints file (.ocl file). In order to generate the OCL constraints file, AWSOME uses the ToOCLVisitor. The ToOCLVisitor is integrated with the evaluate button in the Chitra #1 panel of Tool2009. The ToOCLVisitor visits every node in the AWSOME AST to generate the corresponding .ocl file. The .ocl file contains all the OCL expressions and constraints.

4.6.1 Mapping from AWSOME to OCL 2.0 (.ocl) file

1. Any attribute with default value type as WsLiteralString in the AWSOME AST is represented with its default value in single quotes in the .ocl file.
2. In a post condition, all non-ticked variables are translated by suffixing @pre and all ticked variables have their ticks removed. For example

   in AWL: age = age' + 1

   in OCL: age@pre = age + 1
3. The WsLiteralSet node in the AWSOME AST is represented as Set{ <elements of set separated by comma> } in OCL.
4. The following table represents WsBinaryExpression nodes in OCL 2.0 file.

Table 4.2 (a): Representation of WsBinaryExpression nodes in OCL 2.0

AWSOME AST	OCL 2.0 file
WsDivision	/

WsGreaterThan	>
WsEqual	=
WsGreaterThanOrEqual	>=
WsLessThanorEqual	<=
WsAnd	and
WsAddition	+
WsImplication	implies
WsIn	includes
WsLessThan	<
WsNotEqual	<>
wSubtraction	-
WsUnion	union
WsOr	or
WsIntersection	intersection

5. The following table represents WsUnaryExpression nodes in OCL 2.0 file.

Table 4.2 (b): Representation of WsUnaryExpression nodes in OCL 2.0

AWSOME AST	**OCL 2.0 file**
WsNot	not
WsMinus	-

6. The following table represents WsQuantifiedExpression nodes in OCL 2.0 file.

Table 4.2 (c): Representation of WsQuantifiedExpression nodes in OCL 2.0

AWSOME AST	**OCL 2.0 file**
WsUniversal	forAll
WsExistential	exists

7. The WsThis node in an AWSOME AST is represented as *self* in OCL.
8. The WsEnumerationType and WsEnumerationValue node are represented as enumeration type and value in OCL.
9. The return value of a function is represented as a post condition with the syntax

<function name> = <return value> in AWSOME. The same is represented in OCL as result = <return value>

10. If the WsSelectedComponent accesses a built in function of OCL 2.0 like *size, sum, isEmpty, first, union*, then it is represented as a selected component followed by -> and the built in function name in OCL.

    For example:

    in AWL: deliveredServices.transaction.points.sum()

    in OCL: deliveredServices.transaction.points -> sum()

4.7 OCL Restrictions in AWSOME:

There are some features in OCL 2.0 that are not supported by AWSOME. The following summarizes the restrictions imposed while generating the OCL 2.0 file from the AWSOME AST.

1. AWSOME does not support the advanced OCL constructs such as oclIsTypeOf.
2. In general, attributes can be defined by declaring them in a class or by adding them in an OCL definition statement. But AWSOME has no way to add an attribute dynamically. So AWSOME does not support the generation of OCL definitions.
3. According to OCL grammar, initial value (init) can be supported in attributes and association ends but use of init in association ends is not clear. So AWSOME will not support initial values for association ends.
4. AWSOME does not support sequence operators other than *first*. Refer to Section 4.6.1 (10) to see how AWSOME supports *first.*

5. The post condition with result = <return value> is equivalent to a body expression in OCL. AWSOME will support only the former type.
6. Invariant names are not supported in AWSOME.
7. Because long invariants in AWSOME cannot be viewed in the Dresden OCL Toolkit, the invariants are split at conjunctions and stored as multiple invariants in the generated OCL 2.0 file.

4.8 Summary

This chapter explains in detail about the overall system design and the four transformation or transaction tools. The mapping between AWSOME and the UML 2.0 file with the restrictions and assumptions made are listed out clearly. Similarly the mapping between AWSOME and the OCL 2.0 file with restrictions and assumptions made are mentioned in detail. The idea of making AWSOME interoperable with other tools is clearly discussed in this chapter. The next chapter will discuss in detail the test cases and results obtained.

5. RESULTS

5.1 Introduction

In this chapter the tests conducted on the transformation or transaction tools discussed in Chapter 4 are described. In order to invoke the tests, the Chitra #1 panel was added to Tool2009. The buttons used in the Chitra#1 panel to invoke the tests are discussed in the following sections. The structure of Chitra1Panel is presented in Figure 5.1.

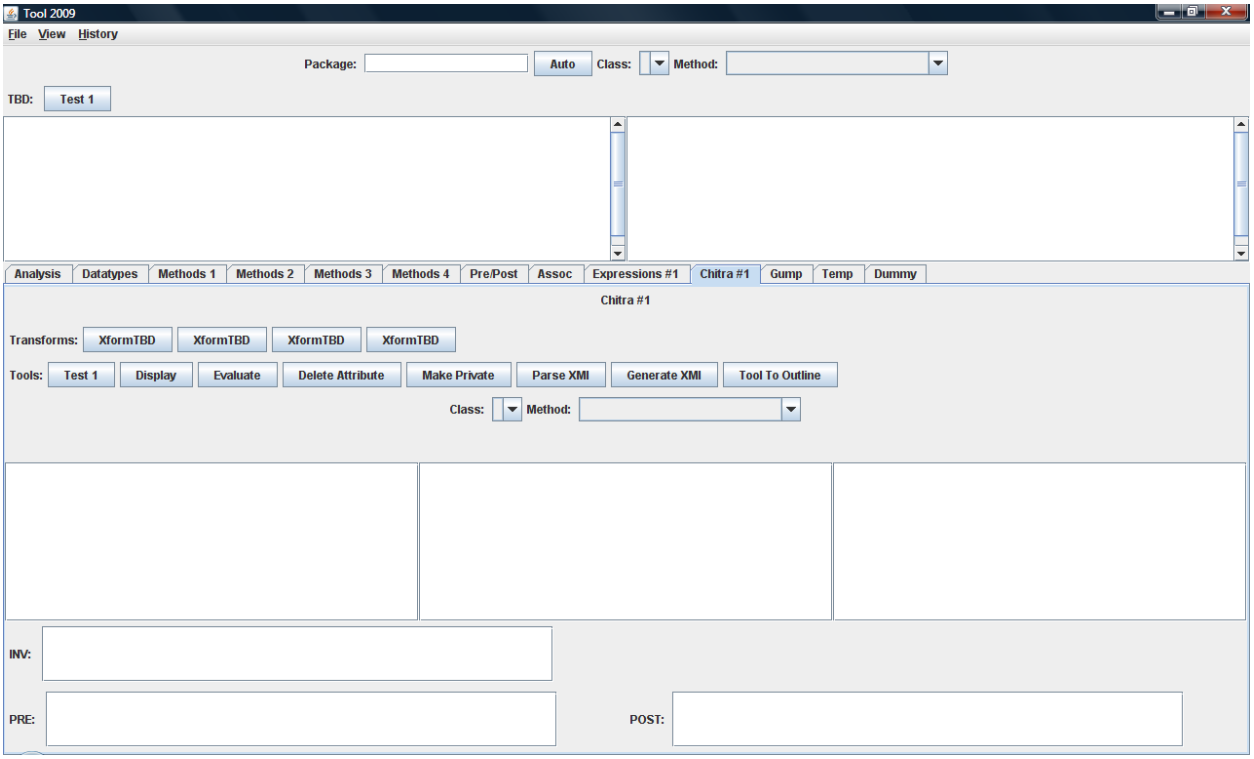


Figure 5.1: Chitra1Panel

5.2 Test Cases

The test cases are conducted on the following three primary transformation or translation tools.

- UML to AWSOME (discussed in Section 5.3)
- AWSOME to UML (discussed in Section 5.4)

- AWSOME to OCL (discussed in Section 5.5)

Each of the three transformation tools are tested and from the test results, the interoperability of AWSOME with other tools is demonstrated.

5.3 UML to AWSOME

The UML to AWSOME tool will parse the .uml file and load it as the AWSOME AST. The .uml files that were used as test cases are presented in the Appendix A. A simple example for the .uml file is given in Chapter 3 under Section 3.4. The test cases were generated so that they contain all UML objects that could be parsed into AWSOME. The different UML objects that can be present in a .uml file is given Table 4.3 of Chapter 4.

The .uml file was parsed in to AWSOME and the following tests were conducted

- the .uml file to the AWSOME AST
- the .uml file to the AWL file
- the .uml file to the outline file

The .uml file is parsed and loaded into AWSOME by clicking the ParseXMI button in the Chitra #1 panel. On clicking the ParseXMI button, the file choosing menu pops up and after choosing the .uml file to be parsed, the AWSOME AST is loaded in Tool2009. The corresponding AWL file is then generated.

The AWSOME AST was visually inspected with the parsed in .uml file to find any differences. There were no differences found in the data associated with the files. Similarly the generated AWL file and the parsed in .uml file were visually compared. No differences were found between the files. On clicking the Tool to Outline button in the Chitra #1 panel, the outline of the currently loaded AWSOME AST is generated. On visual inspection of the outline file and

the .uml file, no differences were found. Thus all the outputs of the UML to AWSOME tool passed the tests successfully. Figure 5.2 is an example of an AWL expression and the corresponding .uml file statement.

AWL Expression 1:

```
Class Person is
public name : String := "y";
end class
```

.uml file statement:

```
<packagedElement xmi:type="uml:Class" xmi:id="_J0lVcMHiEd2UNuYhQ_ApoA"
name="Person">
        <ownedAttribute xmi:id="_MIZyMMHiEd2UNuYhQ_ApoA" name="name"
type="_IX5VAMHiEd2UNuYhQ_ApoA">
          <defaultValue xmi:type="uml:LiteralString" xmi:id="_Mjqx4MHiEd2UNuYhQ_ApoA"
value="y"/>
        </ownedAttribute>
</packagedElement>
```

Figure 5.2: UML to AWSOME Expression

5.4 AWSOME to UML

The AWSOME to UML tool will convert the loaded AWSOME AST to a .uml file. The .awl file was parsed into AWSOME and the following tests were conducted:

- the .awl file to the.uml file
- the .uml file to the AWSOME AST to the .uml file

By selecting the Tool Loader (discussed in Section 2.5.2) in the AWSOME Tool Box, the AWSOME AST is created for the chosen .awl file. The GenerateXMI button of Chitra #1 panel is used to generate the .uml file from the loaded AWSOME AST. On clicking the GenerateXMI

button, the corresponding .uml file is generated and outputted. The AWL file and the generated .uml file were visually inspected and no differences were found.

Similarly a .uml file was parsed into AWSOME and the resulting AWSOME AST was converted back to a .uml file, using the ParseXMI and GenerateXMI buttons of Chitra #1 panel respectively. The parsed in .uml file and generated .uml file were compared by using the file comparison tool named Beyond Compare 3 [17]. No differences were found between the files. Thus the tests conducted for the AWSOME to UML tool passed for all outputs.

5.5 AWSOME to OCL

The AWSOME to OCL tool can generate an OCL file from the loaded AWSOME AST. The following tests were performed

- the .uml file to the extended .awl file to the .ocl file
- the .awl file to the .ocl file

The .awl file can be generated from the .uml file by clicking the ParseXMI button of Chitra #1 panel. The AWL expressions involving invaraints, pre and post conditions etc can be added manually to the .awl file. Because EclipseUML 2008 does not support constraints, the AWL expressions have to be added manually to the .awl file generated from the .uml file. Using the Tool Loader in the AWSOME Tool Box, parse the extended .awl file into AWSOME. By clicking the Evaluate button in the Chitra #1 panel, the corresponding .ocl file is generated.

In the case of an .awl file specifically written for AWSOME, the AWL expressions are already present in it. So the .awl file can be loaded directly into AWSOME and the .ocl file can be generated.

The generated OCL constraints were compared with the AWL expressions and it was found that all the expressions have been correctly translated from AWL to OCL.

Figure 5.3 has few examples of AWL expressions and corresponding OCL constraints.

AWL Expression 1:

```
class LoyaltyProgram is
invariant membership.currentLevel in levels
and forall(p : ProgramPartner) (p in partners => p.deliveredServices.size() >= 1)
and forall(s:Service) ((s in  partners.deliveredServices) => (s.pointsEarned = 0 and s.pointsBurned = 0))
=> (membership.accounts.isEmpty())
end class;
```

OCL constraint 1:

```
context LoyaltyProgram
inv:((((levels)->includes(membership.currentLevel))and(partners->forAll(p|((partners)
->includes(p))implies((p.deliveredServices->size())>=(1)))))and(partners.deliveredServices
->forAll(s|((partners.deliveredServices)-
>includes(s))implies(((s.pointsEarned)=(0))and((s.pointsBurned)=(0))))))implies(membership.accounts
->isEmpty())
```

AWL Expression 2:

```
class LoyaltyAccount is
invariant points > 0 => exists(t : Transaction)(t in transactions => t.points > 0)
end class;
```

OCL Constraint 2:

```
context LoyaltyAccount
inv: ((points)>(0))implies(transactions->exists(t|((transactions)
->includes(t))implies((t.points)>(0))))
```

AWL Expression 3:

```
class Membership
invariant currentLevel.name = "Silver" => (card.color = silver) and
 currentLevel.name = "Gold" => (card.color = gold)
end class;
```

OCL Constraint 3:

context Membership
inv: (((currentLevel.name)=('Silver'))implies(((card.color)=(Color::silver))and((currentLevel.name)=('Gold'))))implies((card.color)=(Color::gold))

Figure 5.3: AWSOME Expressions to OCL constraints

In order to test the validity of the generated OCL file, the Dresden OCL Toolkit can be used. The .uml file can be loaded as a domain model in the Dresden OCL Toolkit. The corresponding .ocl file can be loaded in the Dresden OCL Toolkit as constraints. For all the tests, the Dresden OCL Toolkit was able to load the .ocl file. This means the generated OCL constraints were in the correct syntax that can be parsed by Dresden OCL Toolkit. From the loaded model and the constraints, AspectJ code was generated by the Dresden Toolkit. This serves as a demonstration of the validity of the .ocl file generated from AWSOME.

5.6 Test Cases Used

In order to demonstrate the interoperability between AWSOME, UML 2.0 and OCL the following test cases were used.

- the .uml files (included in Appendix A)
- the .awl files generated by the UML to AWSOME tool (included in Appendix B)
- the .awl files and the corresponding .ocl files generated by the AWSOME to UML tool (included in Appendix C)

5.7 Summary

This chapter discussed in detail about the various tests conducted to ensure the proper working of the three primary transformation or transaction tools. The objective to make AWSOME interoperable with other tools is achieved by successfully passing all tests. The following chapter will present in detail about the conclusions arrived at based on the results obtained.

6. CONCLUSIONS AND FUTURE WORK

6.1 Introduction

This thesis began with the focus of making AWSOME interoperable with other available tools in the MDA context. Each stage of the software engineering life cycle was adopted to reach the goal. From the results obtained, the interoperability of AWSOME with Dresden OCL Toolkit and EclipseUML 2008 has been demonstrated. In the process of demonstrating the interoperability, several transformation or translation tools were constructed. The working of these tools and their use in making AWSOME interoperable with other tools are discussed in Chapter 4. Thus the three primary transformation or translation tools have been the pillars of making AWSOME interoperable with other tools.

6.2 Conclusions

The three primary transformation or translation tools that were constructed to make AWSOME interoperable with other tools are

- UML to AWSOME
- AWSOME to UML
- AWSOME to OCL

The correct working of these tools is demonstrated by conducting various test cases in Chapter 5. The UML to AWSOME and AWSOME to UML tools ensure the interoperability between AWSOME and EclipseUML 2008. The AWSOME to OCL tool ensures the interoperability of AWSOME with Dresden OCL Toolkit.

But due to the lack of availability of tools really using MDA, some functionalities could not be supported. For example, the EclipseUML 2008 cannot support WsContainerFormat and

ArrayTypes present in AWSOME. So such objects in the AWSOME AST will be ignored while generating the .uml files. Also there was no proper documentation, Meta-model, DTD and XML Schema available for the tools EclipseUML and Dresden OCL Toolkit. So certain assumptions had to be made from the generated test cases while mapping UML to AWSOME, AWSOME to UML and AWSOME to OCL. The assumptions 5, 6, 7 and 8 included under Section 4.4 in Chapter 4 serve as examples.

6.3 Recommendations for future research

This thesis has achieved the results that were desired. But there are some problems, identified in Chapter 4, that have to be tackled. The following are recommendations for some of the extensions that can be done to AWSOME in future.

- AWSOME can be extended to support multiple inheritance (discussed in restriction 1 under Section 4.4 in Chapter 4).
- The Tool2009 can be extended to support multiple package viewing (discussed in mapping 22 under Section 4.2.2 in Chapter 4).
- Tool To AWL and AWLParser can be extended to support static attributes (discussed in mapping 5 under Section 4.2.2 in Chapter 4).
- The AWL Parser can be extended to support ordered associations in the AWL file (discussed in mapping 18 under Section 4.2.2 in Chapter 4).
- AWSOME can be extended to support advanced OCL constructs such as oclIsTypeOf (included in restriction 1 under Section 4.7 in Chapter 4).

6.4 Summary

Overall this thesis has achieved the objective of making AWSOME interoperable with other tools under the MDA context. If there are tools for which meta-model, DTD or XML schema was available then the advantages of MDA could be realized to a great extent.

REFERENCES

1. **Anneke G. Kleppe, Jos Warmer, and Wim Bast**. *MDA Explained: The Model Driven Architecture: Practice and Promise*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, 2003.
2. **Balzer, R. and Cheatham, T.E., Jr.and Green, C.,** "Software Technology in the 1990"s: Using a New Paradigm," *IEEE Proceedings*, Vol. 16, November 1983,39-45.
3. **Brown A**., "An introduction to Model Driven Architecture Part I: MDA and today's systems". IBM Corporation. 2004 http://www-106.ibm.com/developerworks/rational/library/3100105.html.
4. **Renas Reda and Yusuf Tozmal**; *Model Driven Architecture –Test Methods and Tools*, School of Engineering at Blekinge Institute of Technology, Sweden, Master's Thesis, 2006
5. **OMG.**, OMG MDA Guide Version 1.0.1, OMG doc.omg/2003-06-01, available at http://www.omg.org/docs/omg/03-06-01.pdf, 2003.
6. **Cordier, T.**, *Lessons Learned From the Model-Driven Architecture Applied to Critical Systems Reliability: A Case Study*, Master's Thesis, 2006
7. **Mohsen Asadi, Mahdy Ravakhah, and Raman Ramsin**, "An MDA-Based System Development Lifecycle," *Asia International Conference on Modelling & Simulation*, pp. 836-842, 2008
8. **Hartrum, Thomas C. and Graham, Robert P., Jr.**, "The AFIT Wide Spectrum Object Modeling Environment: an AWSOME Beginning," *Proceedings of the IEEE 2000 National Aerospace & Electronics Conference (NAECON 2000)*, Oct 10-12, 2000, Dayton, OH, pp. 35-42.

9. **OMG.** Introduction to OMG's Unified Modeling Language (UML), available at http://www.omg.org/gettingstarted/what_is_uml.htm. 2005
10. **Kunle Odutola, Michiel van der Wulp**, *ArgoUML Quick Guide, Get started with ArgoUML 0.28,* available at http://argouml-stats.tigris.org/documentation/quick-guide-0.28/
11. **Alejandro Ramirez, Philippe Vanpeperstraete, Andreas Rueckert, Kunle Odutola, Jeremy Bennett, Linus Tolke and Michiel van der Wulp**, *ArgoUML User Manual - A tutorial and reference description,* available at http://argouml-stats.tigris.org/documentation/manual-0.28/
12. **UML**, Website description in http://www.uml2.org/
13. **Ivano Malavolta, Henry Muccini, Patrizio Pelliccione, Damien Andrew Tamburri**, "Providing Architectural Languages and Tools Interoperability through Model Transformation Technologies," *IEEE Transactions on Software Engineering*, vol. 36, no. 1, pp. 119-140, 2010.
14. **Octopus**, *OCL Tool for Precise Uml Specifications,* available at http://octopus.sourceforge.net/
15. **SourceForge**, Website description in http://dresden-ocl.sourceforge.net/aboutproject.html
16. **SourceForge**, Eclipse package description in http://dresden-ocl.sourceforge.net/4eclipse_packages.html
17. **OMG.** XML Metadata Interchange, available at http://www.omg.org/spec/XMI/ 2005
18. **Scooter Software, Inc**, Beyond Compare, Version 3, available at http://www.scootersoftware.com/

19. **SourceForge**, OCL for Eclipse, description in http://sourceforge.net/projects/dresden-ocl/files/dresden-ocl2-for-eclipse/2.0/ocl2-for-eclipse-2.0.tar.gz/download

20. **Timothy J. Grose, Gary C. Doney, Stephen A. Brodsky**, *Mastering XMI: Java Programming with XMI, XML, and UML*, OMG Press, 2002.

APPENDICES

Appendix A

The following are the input .uml files used to generate corresponding XMI files using UML to AWSOME tool.

simple.uml [18] – This example file comes along with the Dresden OCL Toolkit download.

```
<?xml version="1.0" encoding="UTF-8"?>
<uml:Package xmi:version="2.1" xmlns:xmi="http://schema.omg.org/spec/XMI/2.1"
xmlns:uml="http://www.eclipse.org/uml2/2.1.0/UML"
xmi:id="_6pUsMMHhEd2UNuYhQ_ApoA" name="tudresden">
  <packagedElement xmi:type="uml:Package" xmi:id="_AsKYYMHiEd2UNuYhQ_ApoA"
name="ocl20">
    <packagedElement xmi:type="uml:Package" xmi:id="_BlraUMHiEd2UNuYhQ_ApoA"
name="pivot">
      <packagedElement xmi:type="uml:Package" xmi:id="_Cff-QMHiEd2UNuYhQ_ApoA"
name="examples">
        <packagedElement xmi:type="uml:Package" xmi:id="_DmIPIMHiEd2UNuYhQ_ApoA"
name="simple">
          <packagedElement xmi:type="uml:Class" xmi:id="_J0lVcMHiEd2UNuYhQ_ApoA"
name="Person">
            <ownedAttribute xmi:id="_MIZyMMHiEd2UNuYhQ_ApoA" name="name"
type="_IX5VAMHiEd2UNuYhQ_ApoA">
              <defaultValue xmi:type="uml:LiteralString" xmi:id="_Mjqx4MHiEd2UNuYhQ_ApoA"
value="y"/>
            </ownedAttribute>
            <ownedAttribute xmi:id="_Ppaa0MHiEd2UNuYhQ_ApoA" name="age"
type="_GxO5MMHiEd2UNuYhQ_ApoA"/>
          </packagedElement>
          <packagedElement xmi:type="uml:Class" xmi:id="_Tagt4MHiEd2UNuYhQ_ApoA"
name="Student">
            <generalization xmi:id="_U2mRYMHiEd2UNuYhQ_ApoA"
general="_J0lVcMHiEd2UNuYhQ_ApoA"/>
          </packagedElement>
          <packagedElement xmi:type="uml:Class" xmi:id="_XPL5YMHiEd2UNuYhQ_ApoA"
name="Professor">
            <generalization xmi:id="_XrWQ8MHiEd2UNuYhQ_ApoA"
general="_J0lVcMHiEd2UNuYhQ_ApoA"/>
          </packagedElement>
        </packagedElement>
      </packagedElement>
    </packagedElement>
  </packagedElement>
```

```
 <packagedElement xmi:type="uml:PrimitiveType" xmi:id="_GxO5MMHiEd2UNuYhQ_ApoA"
name="int"/>
 <packagedElement xmi:type="uml:PrimitiveType" xmi:id="_IX5VAMHiEd2UNuYhQ_ApoA"
name="String"/>
</uml:Package>
```

royalsandloyals.uml [18] - This example file is provided along with the Dresden OCL Toolkit download.

```
<?xml version="1.0" encoding="UTF-8"?>
<uml:Package xmi:version="2.1" xmlns:xmi="http://schema.omg.org/spec/XMI/2.1"
xmlns:uml="http://www.eclipse.org/uml2/2.1.0/UML"
xmi:id="_6pUsMMHhEd2UNuYhQ_ApoA" name="tudresden">
 <packagedElement xmi:type="uml:Package" xmi:id="_AsKYYMHiEd2UNuYhQ_ApoA"
name="ocl20">
  <packagedElement xmi:type="uml:Package" xmi:id="_BlraUMHiEd2UNuYhQ_ApoA"
name="pivot">
   <packagedElement xmi:type="uml:Package" xmi:id="_Cff-QMHiEd2UNuYhQ_ApoA"
name="examples">
    <packagedElement xmi:type="uml:Package" xmi:id="_DmIPIMHiEd2UNuYhQ_ApoA"
name="royalsandloyals">
     <packagedElement xmi:type="uml:Class" xmi:id="__d9IUMnCEd23VbTmZ6CCEw"
name="Burning">
      <generalization xmi:id="_EArFkMnDEd23VbTmZ6CCEw"
general="_qdRHkMkkEd2S2cuzoMKaDg"/>
     </packagedElement>
     <packagedElement xmi:type="uml:Class" xmi:id="_VW-hEMVEEd2gxo2Yyu8isA"
name="Customer">
      <ownedAttribute xmi:id="_1y46wMnOEd23VbTmZ6CCEw" name="age"
type="_GxO5MMHiEd2UNuYhQ_ApoA"/>
      <ownedAttribute xmi:id="_MC0aoMoREd2R-LEScdZDXg" name="dateOfBirth"
type="_MVFXUMm5Ed23VbTmZ6CCEw"/>
      <ownedAttribute xmi:id="_FONTQMoREd2R-LEScdZDXg" name="isMale"
type="_73TBsMVDEd2gxo2Yyu8isA"/>
      <ownedAttribute xmi:id="_doldsMkwEd2S2cuzoMKaDg" name="name"
type="_IX5VAMHiEd2UNuYhQ_ApoA"/>
```

```
        <ownedAttribute xmi:id="_noF1gMm5Ed23VbTmZ6CCEw" name="title"
type="_IX5VAMHiEd2UNuYhQ_ApoA"/>
        <ownedOperation xmi:id="_LoRdkMoaEd2R-LEScdZDXg" name="birthdayHappens">
          <ownedParameter xmi:id="_75R_sMoaEd2R-LEScdZDXg" type="_ihHtUMoJEd2R-
LEScdZDXg" direction="return"/>
        </ownedOperation>
        <ownedOperation xmi:id="_KtaX0MoaEd2R-LEScdZDXg" name="getAge">
          <ownedParameter xmi:id="_5XTyMMoaEd2R-LEScdZDXg"
type="_GxO5MMHiEd2UNuYhQ_ApoA" direction="return"/>
        </ownedOperation>
      </packagedElement>
      <packagedElement xmi:type="uml:Class" xmi:id="_fBpSwMVIEd2gxo2Yyu8isA"
name="CustomerCard">
        <ownedAttribute xmi:id="_dJRj4MnSEd23VbTmZ6CCEw" name="color"
type="_2Cn4gMnREd23VbTmZ6CCEw"/>
        <ownedAttribute xmi:id="_f8pz0Mm5Ed23VbTmZ6CCEw" name="printedName"
type="_IX5VAMHiEd2UNuYhQ_ApoA"/>
        <ownedAttribute xmi:id="_gYE6MMVIEd2gxo2Yyu8isA" name="valid"
type="_73TBsMVDEd2gxo2Yyu8isA"/>
        <ownedAttribute xmi:id="__hwMwMnOEd23VbTmZ6CCEw" name="validFrom"
type="_MVFXUMm5Ed23VbTmZ6CCEw"/>
        <ownedAttribute xmi:id="_CZyJ0MnPEd23VbTmZ6CCEw" name="validThru"
type="_MVFXUMm5Ed23VbTmZ6CCEw"/>
      </packagedElement>
      <packagedElement xmi:type="uml:Class" xmi:id="_9ZyRIMnCEd23VbTmZ6CCEw"
name="Earning">
        <generalization xmi:id="_IDgo4MnDEd23VbTmZ6CCEw"
general="_qdRHkMkkEd2S2cuzoMKaDg"/>
      </packagedElement>
      <packagedElement xmi:type="uml:Class" xmi:id="_MG7VMMVIEd2gxo2Yyu8isA"
name="LoyaltyAccount">
        <ownedAttribute xmi:id="_W-Nn0MVIEd2gxo2Yyu8isA" name="points"
type="_GxO5MMHiEd2UNuYhQ_ApoA"/>
        <ownedAttribute xmi:id="_m30qoMnDEd23VbTmZ6CCEw" name="totalPointsEarned"
type="_GxO5MMHiEd2UNuYhQ_ApoA"/>
        <ownedAttribute xmi:id="_ufv9UMobEd2R-LEScdZDXg" name="number"
type="_GxO5MMHiEd2UNuYhQ_ApoA"/>
        <ownedOperation xmi:id="_07xlkMobEd2R-LEScdZDXg" name="burn">
          <ownedParameter xmi:id="_EZge4MocEd2R-LEScdZDXg" name="points"
type="_GxO5MMHiEd2UNuYhQ_ApoA"/>
          <ownedParameter xmi:id="__MBGUMobEd2R-LEScdZDXg"
type="_GxO5MMHiEd2UNuYhQ_ApoA" direction="return"/>
        </ownedOperation>
        <ownedOperation xmi:id="_zvn7gMobEd2R-LEScdZDXg" name="earn">
          <ownedParameter xmi:id="_Az4cAMocEd2R-LEScdZDXg" name="points"
type="_GxO5MMHiEd2UNuYhQ_ApoA"/>
```

```
          <ownedParameter xmi:id="__t2BAMobEd2R-LEScdZDXg"
type="_GxO5MMHiEd2UNuYhQ_ApoA" direction="return"/>
        </ownedOperation>
        <ownedOperation xmi:id="_ymYvEMm5Ed23VbTmZ6CCEw"
name="getCustomerName">
          <ownedParameter xmi:id="_0bqOoMm5Ed23VbTmZ6CCEw"
type="_IX5VAMHiEd2UNuYhQ_ApoA" direction="return"/>
        </ownedOperation>
        <ownedOperation xmi:id="_1eWHIMobEd2R-LEScdZDXg" name="isEmpty">
          <ownedParameter xmi:id="_JHZxYMocEd2R-LEScdZDXg"
type="_73TBsMVDEd2gxo2Yyu8isA" direction="return"/>
        </ownedOperation>
      </packagedElement>
      <packagedElement xmi:type="uml:Class" xmi:id="_AVjVsMVEEd2gxo2Yyu8isA"
name="LoyaltyProgram">
        <ownedAttribute xmi:id="_GSa8wMVEEd2gxo2Yyu8isA" name="name"
type="_IX5VAMHiEd2UNuYhQ_ApoA"/>
        <ownedOperation xmi:id="_v22JkMoPEd2R-LEScdZDXg" name="addService">
          <ownedParameter xmi:id="_zaFm0MoPEd2R-LEScdZDXg" name="aPartner"
type="_ODIWQMn1Ed2R-LEScdZDXg"/>
          <ownedParameter xmi:id="_0-q_AMoPEd2R-LEScdZDXg" name="aLevel"
type="_8fH4kMkmEd2S2cuzoMKaDg"/>
          <ownedParameter xmi:id="_z9QlUMoPEd2R-LEScdZDXg" name="aService"
type="_bFF6cMVEEd2gxo2Yyu8isA"/>
          <ownedParameter xmi:id="_2Yht4MoPEd2R-LEScdZDXg" type="_ihHtUMoJEd2R-
LEScdZDXg" direction="return"/>
        </ownedOperation>
        <ownedOperation xmi:id="_Ne3VUMVEEd2gxo2Yyu8isA" name="enroll"
concurrency="concurrent">
          <ownedParameter xmi:id="_T5ZIwMVEEd2gxo2Yyu8isA" name="c" type="_VW-
hEMVEEd2gxo2Yyu8isA"/>
          <ownedParameter xmi:id="_lRjNkMoJEd2R-LEScdZDXg"
type="_73TBsMVDEd2gxo2Yyu8isA" direction="return"/>
        </ownedOperation>
        <ownedOperation xmi:id="_BSpSEMoQEd2R-LEScdZDXg" name="getName">
          <ownedParameter xmi:id="_C0s6sMoQEd2R-LEScdZDXg"
type="_IX5VAMHiEd2UNuYhQ_ApoA" direction="return"/>
        </ownedOperation>
        <ownedOperation xmi:id="_ZUJhQMVEEd2gxo2Yyu8isA" name="getServices">
          <ownedParameter xmi:id="_NjeDsMqCEd2SKu80i6SbVA" name="return"
type="_bFF6cMVEEd2gxo2Yyu8isA" direction="return">
            <upperValue xmi:type="uml:LiteralUnlimitedNatural"
xmi:id="_Nj8vYMqDEd2SKu80i6SbVA" value="*"/>
            <lowerValue xmi:type="uml:LiteralInteger"
xmi:id="_NSXQIMqDEd2SKu80i6SbVA"/>
          </ownedParameter>
```

```
        </ownedOperation>
      </packagedElement>
      <packagedElement xmi:type="uml:Class" xmi:id="_ktNNcMkkEd2S2cuzoMKaDg"
name="Membership"/>
      <packagedElement xmi:type="uml:Class" xmi:id="_ODIWQMn1Ed2R-LEScdZDXg"
name="ProgramPartner">
        <ownedAttribute xmi:id="__tTjkModEd2R-LEScdZDXg" name="name"
type="_IX5VAMHiEd2UNuYhQ_ApoA"/>
        <ownedAttribute xmi:id="_6y_F0Mn1Ed2R-LEScdZDXg" name="numberOfCustomers"
type="_GxO5MMHiEd2UNuYhQ_ApoA"/>
      </packagedElement>
      <packagedElement xmi:type="uml:Class" xmi:id="_bFF6cMVEEd2gxo2Yyu8isA"
name="Service">
        <ownedAttribute xmi:id="_pOKSQMn3Ed2R-LEScdZDXg" name="condition"
type="_73TBsMVDEd2gxo2Yyu8isA"/>
        <ownedAttribute xmi:id="_rZL_YMn3Ed2R-LEScdZDXg" name="pointsBurned"
type="_GxO5MMHiEd2UNuYhQ_ApoA"/>
        <ownedAttribute xmi:id="_q2Bn8Mn3Ed2R-LEScdZDXg" name="pointsEarned"
type="_GxO5MMHiEd2UNuYhQ_ApoA"/>
        <ownedAttribute xmi:id="_r4VZgMn3Ed2R-LEScdZDXg" name="description"
type="_IX5VAMHiEd2UNuYhQ_ApoA"/>
        <ownedAttribute xmi:id="_scP-4Mn3Ed2R-LEScdZDXg" name="serviceNr"
type="_GxO5MMHiEd2UNuYhQ_ApoA"/>
        <ownedOperation xmi:id="_Zl_OQModEd2R-LEScdZDXg" name="calcPoints">
          <ownedParameter xmi:id="_gOqycModEd2R-LEScdZDXg"
type="_GxO5MMHiEd2UNuYhQ_ApoA" direction="return"/>
        </ownedOperation>
        <ownedOperation xmi:id="_aCmRwModEd2R-LEScdZDXg"
name="upgradePointsEarned">
          <ownedParameter xmi:id="_pR4fYModEd2R-LEScdZDXg" name="amount"
type="_GxO5MMHiEd2UNuYhQ_ApoA"/>
          <ownedParameter xmi:id="_jSnroModEd2R-LEScdZDXg" type="_ihHtUMoJEd2R-
LEScdZDXg" direction="return"/>
        </ownedOperation>
      </packagedElement>
      <packagedElement xmi:type="uml:Class" xmi:id="_8fH4kMkmEd2S2cuzoMKaDg"
name="ServiceLevel">
        <ownedAttribute xmi:id="_blCOYMksEd2S2cuzoMKaDg" name="name"
type="_IX5VAMHiEd2UNuYhQ_ApoA"/>
      </packagedElement>
      <packagedElement xmi:type="uml:Class" xmi:id="_qdRHkMkkEd2S2cuzoMKaDg"
name="Transaction">
        <ownedAttribute xmi:id="_fZ8HkMklEd2S2cuzoMKaDg" name="amount"
type="_kJxTwMklEd2S2cuzoMKaDg"/>
        <ownedAttribute xmi:id="__Y7BIMkvEd2S2cuzoMKaDg" name="points"
type="_GxO5MMHiEd2UNuYhQ_ApoA"/>
```

```
        <ownedAttribute xmi:id="_hr_PkMoeEd2R-LEScdZDXg" name="date"
type="_MVFXUMm5Ed23VbTmZ6CCEw"/>
        <ownedOperation xmi:id="_nI91oMoeEd2R-LEScdZDXg" name="getProgram">
          <ownedParameter xmi:id="_o1TboMoeEd2R-LEScdZDXg"
type="_AVjVsMVEEd2gxo2Yyu8isA" direction="return"/>
        </ownedOperation>
      </packagedElement>
      <packagedElement xmi:type="uml:Association"
xmi:id="_3XPEYMm4Ed23VbTmZ6CCEw" name="Customer CustomerCards"
memberEnd="_5liH8Mm4Ed23VbTmZ6CCEw _6tqNoMm4Ed23VbTmZ6CCEw">
        <ownedEnd xmi:id="_5liH8Mm4Ed23VbTmZ6CCEw" name="owner" type="_VW-
hEMVEEd2gxo2Yyu8isA" association="_3XPEYMm4Ed23VbTmZ6CCEw"/>
        <ownedEnd xmi:id="_6tqNoMm4Ed23VbTmZ6CCEw" name="cards"
type="_fBpSwMVIEd2gxo2Yyu8isA" association="_3XPEYMm4Ed23VbTmZ6CCEw">
          <upperValue xmi:type="uml:LiteralUnlimitedNatural"
xmi:id="_xFOxcMnREd23VbTmZ6CCEw" value="*"/>
          <lowerValue xmi:type="uml:LiteralInteger"
xmi:id="_w0PvIMnREd23VbTmZ6CCEw"/>
        </ownedEnd>
      </packagedElement>
      <packagedElement xmi:type="uml:Association" xmi:id="_1e-
1YMnQEd23VbTmZ6CCEw" name="Customers LoyaltyPrograms"
memberEnd="_3Q1NgMnQEd23VbTmZ6CCEw _4ns50MnQEd23VbTmZ6CCEw">
        <ownedEnd xmi:id="_3Q1NgMnQEd23VbTmZ6CCEw" name="participants"
type="_VW-hEMVEEd2gxo2Yyu8isA" association="_1e-1YMnQEd23VbTmZ6CCEw">
          <upperValue xmi:type="uml:LiteralUnlimitedNatural"
xmi:id="_8DMwUMnQEd23VbTmZ6CCEw" value="*"/>
          <lowerValue xmi:type="uml:LiteralInteger"
xmi:id="_7t4noMnQEd23VbTmZ6CCEw"/>
        </ownedEnd>
        <ownedEnd xmi:id="_4ns50MnQEd23VbTmZ6CCEw" name="programs"
type="_AVjVsMVEEd2gxo2Yyu8isA" association="_1e-1YMnQEd23VbTmZ6CCEw">
          <upperValue xmi:type="uml:LiteralUnlimitedNatural" xmi:id="_-
YNf4MnQEd23VbTmZ6CCEw" value="*"/>
          <lowerValue xmi:type="uml:LiteralInteger" xmi:id="_-
Fk3wMnQEd23VbTmZ6CCEw"/>
        </ownedEnd>
      </packagedElement>
      <packagedElement xmi:type="uml:Association"
xmi:id="_kNUNMMm4Ed23VbTmZ6CCEw" name="CustomerCard Membership"
memberEnd="_p5fs4Mm4Ed23VbTmZ6CCEw _rdoZIMm4Ed23VbTmZ6CCEw">
        <ownedEnd xmi:id="_p5fs4Mm4Ed23VbTmZ6CCEw" name="card"
type="_fBpSwMVIEd2gxo2Yyu8isA" association="_kNUNMMm4Ed23VbTmZ6CCEw"/>
        <ownedEnd xmi:id="_rdoZIMm4Ed23VbTmZ6CCEw" name="membership"
type="_ktNNcMkkEd2S2cuzoMKaDg" association="_kNUNMMm4Ed23VbTmZ6CCEw"/>
      </packagedElement>
```

```
        <packagedElement xmi:type="uml:Association" xmi:id="_cpOH4MkvEd2S2cuzoMKaDg"
name="CustomerCard Transactions" memberEnd="_e_SAUMkvEd2S2cuzoMKaDg
_fo2hwMkvEd2S2cuzoMKaDg">
          <ownedEnd xmi:id="_e_SAUMkvEd2S2cuzoMKaDg" name="card"
type="_fBpSwMVIEd2gxo2Yyu8isA" association="_cpOH4MkvEd2S2cuzoMKaDg"/>
          <ownedEnd xmi:id="_fo2hwMkvEd2S2cuzoMKaDg" name="transactions"
type="_qdRHkMkkEd2S2cuzoMKaDg" association="_cpOH4MkvEd2S2cuzoMKaDg">
            <upperValue xmi:type="uml:LiteralUnlimitedNatural"
xmi:id="_kERkMMkvEd2S2cuzoMKaDg" value="*"/>
            <lowerValue xmi:type="uml:LiteralInteger" xmi:id="_jtLS0MkvEd2S2cuzoMKaDg"/>
          </ownedEnd>
        </packagedElement>
        <packagedElement xmi:type="uml:Association"
xmi:id="_4jBocMnAEd23VbTmZ6CCEw" name="LoyaltyAccounts Membership"
memberEnd="_7Yh18MnAEd23VbTmZ6CCEw _7_txwMnAEd23VbTmZ6CCEw">
          <ownedEnd xmi:id="_7Yh18MnAEd23VbTmZ6CCEw" name="membership"
type="_ktNNcMkkEd2S2cuzoMKaDg" association="_4jBocMnAEd23VbTmZ6CCEw">
            <defaultValue xmi:type="uml:LiteralString"
xmi:id="_9Xxw4MnAEd23VbTmZ6CCEw" value=""/>
          </ownedEnd>
          <ownedEnd xmi:id="_7_txwMnAEd23VbTmZ6CCEw" name="accounts"
type="_MG7VMMVIEd2gxo2Yyu8isA" association="_4jBocMnAEd23VbTmZ6CCEw">
            <upperValue xmi:type="uml:LiteralUnlimitedNatural"
xmi:id="_BeV9oMnBEd23VbTmZ6CCEw" value="*"/>
            <lowerValue xmi:type="uml:LiteralInteger"
xmi:id="_Bvn64MnBEd23VbTmZ6CCEw"/>
          </ownedEnd>
        </packagedElement>
        <packagedElement xmi:type="uml:Association" xmi:id="_yIQdUMkkEd2S2cuzoMKaDg"
name="LoyaltyAccount Transactions" memberEnd="_C1aXIMklEd2S2cuzoMKaDg
_HkgjgMklEd2S2cuzoMKaDg">
          <ownedEnd xmi:id="_C1aXIMklEd2S2cuzoMKaDg" name="transactions"
type="_qdRHkMkkEd2S2cuzoMKaDg" association="_yIQdUMkkEd2S2cuzoMKaDg">
            <upperValue xmi:type="uml:LiteralUnlimitedNatural"
xmi:id="_TM90YMklEd2S2cuzoMKaDg" value="*"/>
          </ownedEnd>
          <ownedEnd xmi:id="_HkgjgMklEd2S2cuzoMKaDg" name="account"
type="_MG7VMMVIEd2gxo2Yyu8isA" association="_yIQdUMkkEd2S2cuzoMKaDg"/>
        </packagedElement>
        <packagedElement xmi:type="uml:Association" xmi:id="_WSNl0MocEd2R-LEScdZDXg"
name="LoyaltyPrograms ProgramPartners" memberEnd="_dNOQIMocEd2R-LEScdZDXg
_eENigMocEd2R-LEScdZDXg">
          <ownedEnd xmi:id="_dNOQIMocEd2R-LEScdZDXg" name="programs"
type="_AVjVsMVEEd2gxo2Yyu8isA" association="_WSNl0MocEd2R-LEScdZDXg">
            <upperValue xmi:type="uml:LiteralUnlimitedNatural" xmi:id="_fN9_4MocEd2R-
LEScdZDXg" value="*"/>
```

```
        </ownedEnd>
        <ownedEnd xmi:id="_eENigMocEd2R-LEScdZDXg" name="partners"
type="_ODIWQMn1Ed2R-LEScdZDXg" association="_WSNl0MocEd2R-LEScdZDXg">
          <upperValue xmi:type="uml:LiteralUnlimitedNatural" xmi:id="_jwGHQMocEd2R-
LEScdZDXg" value="*"/>
        </ownedEnd>
      </packagedElement>
      <packagedElement xmi:type="uml:Association" xmi:id="_zjGjUMknEd2S2cuzoMKaDg"
name="LoyaltyProgram ServiceLevels" memberEnd="_l9pk0MkoEd2S2cuzoMKaDg
_n8Tp4MkoEd2S2cuzoMKaDg">
        <ownedEnd xmi:id="_l9pk0MkoEd2S2cuzoMKaDg" name="program"
type="_AVjVsMVEEd2gxo2Yyu8isA" association="_zjGjUMknEd2S2cuzoMKaDg"/>
        <ownedEnd xmi:id="_n8Tp4MkoEd2S2cuzoMKaDg" name="levels"
type="_8fH4kMkmEd2S2cuzoMKaDg" isOrdered="true"
association="_zjGjUMknEd2S2cuzoMKaDg">
          <upperValue xmi:type="uml:LiteralUnlimitedNatural"
xmi:id="_wMZ30MkoEd2S2cuzoMKaDg" value="*"/>
        </ownedEnd>
      </packagedElement>
      <packagedElement xmi:type="uml:Association"
xmi:id="_gpxhEMnQEd23VbTmZ6CCEw" name="LoyaltyProgram Membership"
memberEnd="_nTWdIMnQEd23VbTmZ6CCEw _oBQE8MnQEd23VbTmZ6CCEw">
        <ownedEnd xmi:id="_nTWdIMnQEd23VbTmZ6CCEw" name="membership"
type="_ktNNcMkkEd2S2cuzoMKaDg" association="_gpxhEMnQEd23VbTmZ6CCEw"/>
        <ownedEnd xmi:id="_oBQE8MnQEd23VbTmZ6CCEw" name="program"
type="_AVjVsMVEEd2gxo2Yyu8isA" association="_gpxhEMnQEd23VbTmZ6CCEw"/>
      </packagedElement>
      <packagedElement xmi:type="uml:Association"
xmi:id="_fWAwMMktEd2S2cuzoMKaDg" name="Memberships ServiceLevel"
memberEnd="_lVke4MktEd2S2cuzoMKaDg _yHf5QMktEd2S2cuzoMKaDg">
        <ownedEnd xmi:id="_lVke4MktEd2S2cuzoMKaDg" name="memberships"
type="_ktNNcMkkEd2S2cuzoMKaDg" association="_fWAwMMktEd2S2cuzoMKaDg">
          <upperValue xmi:type="uml:LiteralUnlimitedNatural"
xmi:id="_uQ4NAMktEd2S2cuzoMKaDg" value="*"/>
        </ownedEnd>
        <ownedEnd xmi:id="_yHf5QMktEd2S2cuzoMKaDg" name="currentLevel"
type="_8fH4kMkmEd2S2cuzoMKaDg" association="_fWAwMMktEd2S2cuzoMKaDg"/>
      </packagedElement>
      <packagedElement xmi:type="uml:Association" xmi:id="_eiu8QMn1Ed2R-LEScdZDXg"
name="ProgramPartner Services" memberEnd="_gcCAMMn1Ed2R-LEScdZDXg _g-
5csMn1Ed2R-LEScdZDXg">
        <ownedEnd xmi:id="_gcCAMMn1Ed2R-LEScdZDXg" name="partner"
type="_ODIWQMn1Ed2R-LEScdZDXg" association="_eiu8QMn1Ed2R-LEScdZDXg"/>
        <ownedEnd xmi:id="_g-5csMn1Ed2R-LEScdZDXg" name="deliveredServices"
type="_bFF6cMVEEd2gxo2Yyu8isA" association="_eiu8QMn1Ed2R-LEScdZDXg">
```

```
          <upperValue xmi:type="uml:LiteralUnlimitedNatural" xmi:id="_lmsHMMn1Ed2R-
LEScdZDXg" value="*"/>
          <lowerValue xmi:type="uml:LiteralInteger" xmi:id="_lNWaIMn1Ed2R-
LEScdZDXg"/>
        </ownedEnd>
      </packagedElement>
      <packagedElement xmi:type="uml:Association" xmi:id="_geG2IMksEd2S2cuzoMKaDg"
name="Services ServiceLevel" memberEnd="_kSVVsMksEd2S2cuzoMKaDg
_qIWt4MksEd2S2cuzoMKaDg">
        <ownedEnd xmi:id="_kSVVsMksEd2S2cuzoMKaDg" name="availableServices"
type="_bFF6cMVEEd2gxo2Yyu8isA" association="_geG2IMksEd2S2cuzoMKaDg">
          <upperValue xmi:type="uml:LiteralUnlimitedNatural"
xmi:id="_mMOPgMksEd2S2cuzoMKaDg" value="*"/>
          <lowerValue xmi:type="uml:LiteralInteger" xmi:id="_nbzBAMksEd2S2cuzoMKaDg"/>
        </ownedEnd>
        <ownedEnd xmi:id="_qIWt4MksEd2S2cuzoMKaDg" name="level"
type="_8fH4kMkmEd2S2cuzoMKaDg" association="_geG2IMksEd2S2cuzoMKaDg"/>
      </packagedElement>
      <packagedElement xmi:type="uml:Association" xmi:id="_tMXmkMkuEd2S2cuzoMKaDg"
name="Service Transaction" memberEnd="_0DzYcMkuEd2S2cuzoMKaDg
_3mJd0MkuEd2S2cuzoMKaDg">
        <ownedEnd xmi:id="_0DzYcMkuEd2S2cuzoMKaDg" name="service"
type="_bFF6cMVEEd2gxo2Yyu8isA" association="_tMXmkMkuEd2S2cuzoMKaDg"/>
        <ownedEnd xmi:id="_3mJd0MkuEd2S2cuzoMKaDg" name="transaction"
type="_qdRHkMkkEd2S2cuzoMKaDg" association="_tMXmkMkuEd2S2cuzoMKaDg"/>
      </packagedElement>
      <packagedElement xmi:type="uml:DataType"
xmi:id="_MVFXUMm5Ed23VbTmZ6CCEw" name="Date">
        <ownedAttribute xmi:id="_EPbawfQqEd2SMeEUZ5YgYQ" name="now" isStatic="true"
type="_MVFXUMm5Ed23VbTmZ6CCEw"/>
        <ownedAttribute xmi:id="_73W4YPQwEd2SMeEUZ5YgYQ" name="nowString"
isStatic="true" type="_IX5VAMHiEd2UNuYhQ_ApoA"/>
        <ownedOperation xmi:id="_skwXgMkvEd2S2cuzoMKaDg" name="isAfter">
          <ownedParameter xmi:id="_t7e54MkvEd2S2cuzoMKaDg" name="aDate"
type="_MVFXUMm5Ed23VbTmZ6CCEw"/>
          <ownedParameter xmi:id="_1d1RAMkvEd2S2cuzoMKaDg"
type="_73TBsMVDEd2gxo2Yyu8isA" direction="return"/>
        </ownedOperation>
        <ownedOperation xmi:id="_2_vIoMkvEd2S2cuzoMKaDg" name="isBefore">
          <ownedParameter xmi:id="_2_vIockvEd2S2cuzoMKaDg" name="aDate"
type="_MVFXUMm5Ed23VbTmZ6CCEw"/>
          <ownedParameter xmi:id="_2_vIoskvEd2S2cuzoMKaDg"
type="_73TBsMVDEd2gxo2Yyu8isA" direction="return"/>
        </ownedOperation>
        <ownedOperation xmi:id="_gQ8boMojEd2R-LEScdZDXg" name="now"
isStatic="true">
```

```
          <ownedParameter xmi:id="_lkjDQMojEd2R-LEScdZDXg"
type="_MVFXUMm5Ed23VbTmZ6CCEw" direction="return"/>
         </ownedOperation>
         <ownedOperation xmi:id="_DDgWAPQxEd2SMeEUZ5YgYQ" name="nowAsString"
isStatic="true">
          <ownedParameter xmi:id="_Gb3Y8PQxEd2SMeEUZ5YgYQ"
type="_IX5VAMHiEd2UNuYhQ_ApoA" direction="return"/>
         </ownedOperation>
         <ownedOperation xmi:id="_Du6CcPQ8Ed2SMeEUZ5YgYQ" name="toString">
          <ownedParameter xmi:id="_I7zlIPQ8Ed2SMeEUZ5YgYQ"
type="_IX5VAMHiEd2UNuYhQ_ApoA" direction="return"/>
         </ownedOperation>
        </packagedElement>
        <packagedElement xmi:type="uml:Enumeration"
xmi:id="_2Cn4gMnREd23VbTmZ6CCEw" name="Color">
         <ownedLiteral xmi:id="_Blk-UMnSEd23VbTmZ6CCEw" name="silver"/>
         <ownedLiteral xmi:id="_C_ILMMnSEd23VbTmZ6CCEw" name="gold"/>
        </packagedElement>
       </packagedElement>
     </packagedElement>
    </packagedElement>
  </packagedElement>
  <packagedElement xmi:type="uml:PrimitiveType" xmi:id="_73TBsMVDEd2gxo2Yyu8isA"
name="Boolean"/>
  <packagedElement xmi:type="uml:PrimitiveType" xmi:id="_kJxTwMklEd2S2cuzoMKaDg"
name="Double"/>
  <packagedElement xmi:type="uml:PrimitiveType" xmi:id="_GxO5MMHiEd2UNuYhQ_ApoA"
name="Integer"/>
  <packagedElement xmi:type="uml:PrimitiveType" xmi:id="_IX5VAMHiEd2UNuYhQ_ApoA"
name="String"/>
  <packagedElement xmi:type="uml:PrimitiveType" xmi:id="_ihHtUMoJEd2R-LEScdZDXg"
name="void"/>
</uml:Package>
```

test1.uml

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<uml:Package xmi:version="2.1" xmlns:xmi="http://schema.omg.org/spec/XMI/2.1"
xmlns:uml="http://www.eclipse.org/uml2/2.1.0/UML"
xmi:id="_6pUsMMHhEd2UNuYhQ_ApoA" name="tudresden">
  <packagedElement xmi:type="uml:Package" xmi:id="_AsKYYMHiEd2UNuYhQ_ApoA"
name="ocl20">
    <packagedElement xmi:type="uml:Package" xmi:id="_BlraUMHiEd2UNuYhQ_ApoA"
name="pivot">
      <packagedElement xmi:type="uml:Package" xmi:id="_Cff-QMHiEd2UNuYhQ_ApoA"
name="examples">
        <packagedElement xmi:type="uml:Package" xmi:id="_DmIPIMHiEd2UNuYhQ_ApoA"
name="test1">
          <packagedElement xmi:type="uml:Class" xmi:id="_J0lVcMHiEd2UNuYhQ_ApoA"
name="Person">
            <ownedAttribute xmi:id="_MIZyMMHiEd2UNuYhQ_ApoA" name="name"
type="_IX5VAMHiEd2UNuYhQ_ApoA">
              <defaultValue xmi:type="uml:LiteralString" xmi:id="_Mjqx4MHiEd2UNuYhQ_ApoA"
value="John"/>
            </ownedAttribute>
            <ownedAttribute xmi:id="_Ppaa0MHiEd2UNuYhQ_ApoA" name="age"
type="_GxO5MMHiEd2UNuYhQ_ApoA"/>
            <ownedAttribute xmi:id="_f8UZEICGEd61TpGc1KcibQ"
name="socialSecurityNumber" visibility="private" type="_GxO5MMHiEd2UNuYhQ_ApoA">
              <defaultValue xmi:type="uml:LiteralInteger" xmi:id="_pHDKAIDYEd6xHIVO6jg60A"
value="1234567890"/>
            </ownedAttribute>
            <ownedAttribute xmi:id="_IMxZMIUpEd6n3uiHOXeS5A" name="grade" type="_Fh-
m4IUpEd6n3uiHOXeS5A"/>
            <ownedOperation xmi:id="_Yk3tMIUiEd6n3uiHOXeS5A" name="getAge">
              <ownedParameter xmi:id="_cvMYkIUiEd6n3uiHOXeS5A" name=""
type="_GxO5MMHiEd2UNuYhQ_ApoA" direction="return">
                <upperValue xmi:type="uml:LiteralUnlimitedNatural"
xmi:id="_o8jtUIUiEd6n3uiHOXeS5A" value="100"/>
                <lowerValue xmi:type="uml:LiteralInteger" xmi:id="_pocJoIUnEd6n3uiHOXeS5A"
value="1"/>
              </ownedParameter>
            </ownedOperation>
            <ownedOperation xmi:id="_8sDvEIUoEd6n3uiHOXeS5A" name="getGrade">
              <ownedParameter xmi:id="_AiaVkIUpEd6n3uiHOXeS5A" type="_Fh-
m4IUpEd6n3uiHOXeS5A" direction="return">
                <upperValue xmi:type="uml:LiteralUnlimitedNatural"
xmi:id="_TCV0kIUpEd6n3uiHOXeS5A" value="4"/>
                <lowerValue xmi:type="uml:LiteralInteger" xmi:id="_OIl3YIUpEd6n3uiHOXeS5A"/>
              </ownedParameter>
            </ownedOperation>
          </packagedElement>
```

```
        <packagedElement xmi:type="uml:Class" xmi:id="_Tagt4MHiEd2UNuYhQ_ApoA"
name="Student">
          <generalization xmi:id="_U2mRYMHiEd2UNuYhQ_ApoA"
general="_J0lVcMHiEd2UNuYhQ_ApoA"/>
        </packagedElement>
        <packagedElement xmi:type="uml:Class" xmi:id="_XPL5YMHiEd2UNuYhQ_ApoA"
name="Professor">
          <generalization xmi:id="_XrWQ8MHiEd2UNuYhQ_ApoA"
general="_J0lVcMHiEd2UNuYhQ_ApoA"/>
        </packagedElement>
      </packagedElement>
    </packagedElement>
  </packagedElement>
 </packagedElement>
 <packagedElement xmi:type="uml:PrimitiveType" xmi:id="_GxO5MMHiEd2UNuYhQ_ApoA"
name="int"/>
 <packagedElement xmi:type="uml:PrimitiveType" xmi:id="_IX5VAMHiEd2UNuYhQ_ApoA"
name="String"/>
 <packagedElement xmi:type="uml:PrimitiveType" xmi:id="_Fh-m4IUpEd6n3uiHOXeS5A"
name="Double"/>
</uml:Package>
```

test2.uml

```
<?xml version="1.0" encoding="UTF-8"?>
<uml:Package xmi:version="2.1" xmlns:xmi="http://schema.omg.org/spec/XMI/2.1"
xmlns:uml="http://www.eclipse.org/uml2/2.1.0/UML"
xmi:id="_6pUsMMHhEd2UNuYhQ_ApoA" name="tudresden">
 <packagedElement xmi:type="uml:Package" xmi:id="_AsKYYMHiEd2UNuYhQ_ApoA"
name="ocl20">
  <packagedElement xmi:type="uml:Package" xmi:id="_BlraUMHiEd2UNuYhQ_ApoA"
name="pivot">
   <packagedElement xmi:type="uml:Package" xmi:id="_Cff-QMHiEd2UNuYhQ_ApoA"
name="examples">
    <packagedElement xmi:type="uml:Package" xmi:id="_DmIPIMHiEd2UNuYhQ_ApoA"
name="test2">
        <packagedElement xmi:type="uml:Class" xmi:id="_J0lVcMHiEd2UNuYhQ_ApoA"
name="Person" isAbstract="true">
          <ownedAttribute xmi:id="_KLqpkIEdEd6xHIVO6jg60A" name="name"
type="_GxO5MMHiEd2UNuYhQ_ApoA"/>
        </packagedElement>
        <packagedElement xmi:type="uml:Class" xmi:id="_Tagt4MHiEd2UNuYhQ_ApoA"
name="Student">
          <generalization xmi:id="_XzejEIEdEd6xHIVO6jg60A"
general="_J0lVcMHiEd2UNuYhQ_ApoA"/>
```

```
        <ownedAttribute xmi:id="_t1BSAID5Ed6xHIVO6jg60A" name="grade"
type="_h_R9AIFPEd6-295AH3icIA"/>
      </packagedElement>
      <packagedElement xmi:type="uml:AssociationClass"
xmi:id="_blNyUID8Ed6xHIVO6jg60A" name="Married"
memberEnd="_t52nUID8Ed6xHIVO6jg60A _00TQ8ID8Ed6xHIVO6jg60A">
        <ownedAttribute xmi:id="_2UXXUIWjEd6JsuKX6y7LmQ" name="date"
type="_kYPZQIEfEd6xHIVO6jg60A"/>
        <ownedOperation xmi:id="_y4bsQIEfEd6xHIVO6jg60A" name="marry">
          <ownedParameter xmi:id="_0_-AgIEfEd6xHIVO6jg60A" name="p"
type="_J0lVcMHiEd2UNuYhQ_ApoA"/>
          <ownedParameter xmi:id="_4udEcIEfEd6xHIVO6jg60A" name="s"
type="_Tagt4MHiEd2UNuYhQ_ApoA"/>
          <ownedParameter xmi:id="_97ZDYIEfEd6xHIVO6jg60A" name="d"
type="_kYPZQIEfEd6xHIVO6jg60A"/>
        </ownedOperation>
        <ownedEnd xmi:id="_t52nUID8Ed6xHIVO6jg60A" name="Husband"
type="_J0lVcMHiEd2UNuYhQ_ApoA" association="_blNyUID8Ed6xHIVO6jg60A">
          <lowerValue xmi:type="uml:LiteralInteger" xmi:id="__L0eYID8Ed6xHIVO6jg60A"/>
        </ownedEnd>
        <ownedEnd xmi:id="_00TQ8ID8Ed6xHIVO6jg60A" name="Wife"
type="_Tagt4MHiEd2UNuYhQ_ApoA" association="_blNyUID8Ed6xHIVO6jg60A">
          <lowerValue xmi:type="uml:LiteralInteger" xmi:id="_-S_Y8ID8Ed6xHIVO6jg60A"/>
        </ownedEnd>
      </packagedElement>
    </packagedElement>
  </packagedElement>
 </packagedElement>
 <packagedElement xmi:type="uml:PrimitiveType" xmi:id="_GxO5MMHiEd2UNuYhQ_ApoA"
name="String"/>
 <packagedElement xmi:type="uml:PrimitiveType" xmi:id="_kYPZQIEfEd6xHIVO6jg60A"
name="Date"/>
 <packagedElement xmi:type="uml:PrimitiveType" xmi:id="_h_R9AIFPEd6-295AH3icIA"
name="int"/>
</uml:Package>
```

test3.uml

```
<?xml version="1.0" encoding="UTF-8"?>
<uml:Package xmi:version="2.1" xmlns:xmi="http://schema.omg.org/spec/XMI/2.1"
xmlns:uml="http://www.eclipse.org/uml2/2.1.0/UML"
xmi:id="_6pUsMMHhEd2UNuYhQ_ApoA" name="tudresden">
 <packagedElement xmi:type="uml:Package" xmi:id="_AsKYYMHiEd2UNuYhQ_ApoA"
name="ocl20">
```

```
  <packagedElement xmi:type="uml:Package" xmi:id="_BlraUMHiEd2UNuYhQ_ApoA"
name="pivot">
    <packagedElement xmi:type="uml:Package" xmi:id="_Cff-QMHiEd2UNuYhQ_ApoA"
name="examples">
      <packagedElement xmi:type="uml:Package" xmi:id="_DmIPIMHiEd2UNuYhQ_ApoA"
name="test3">
        <packagedElement xmi:type="uml:Class" xmi:id="_J0lVcMHiEd2UNuYhQ_ApoA"
name="School">
          <ownedAttribute xmi:id="_KLqpkIEdEd6xHIVO6jg60A" name="name"
type="_GxO5MMHiEd2UNuYhQ_ApoA"/>
        </packagedElement>
        <packagedElement xmi:type="uml:Class" xmi:id="_Tagt4MHiEd2UNuYhQ_ApoA"
name="Student">
          <generalization xmi:id="_XzejEIEdEd6xHIVO6jg60A"
general="_J0lVcMHiEd2UNuYhQ_ApoA"/>
          <ownedAttribute xmi:id="_t1BSAID5Ed6xHIVO6jg60A" name="grade"
type="_h_R9AIFPEd6-295AH3icIA"/>
        </packagedElement>
        <packagedElement xmi:type="uml:Association" xmi:id="_FJxCUIXFEd6N0dIhtm_qSQ"
name="School Student" memberEnd="_CImh8IXGEd6N0dIhtm_qSQ
_nFnAQIXGEd6N0dIhtm_qSQ">
          <ownedEnd xmi:id="_CImh8IXGEd6N0dIhtm_qSQ" name="s"
type="_J0lVcMHiEd2UNuYhQ_ApoA" aggregation="shared"
association="_FJxCUIXFEd6N0dIhtm_qSQ"/>
          <ownedEnd xmi:id="_nFnAQIXGEd6N0dIhtm_qSQ" name="stu"
type="_Tagt4MHiEd2UNuYhQ_ApoA" aggregation="composite"
association="_FJxCUIXFEd6N0dIhtm_qSQ">
            <upperValue xmi:type="uml:LiteralUnlimitedNatural"
xmi:id="_qyx1kIXGEd6N0dIhtm_qSQ" value="*"/>
            <lowerValue xmi:type="uml:LiteralInteger" xmi:id="_qVZ9EIXGEd6N0dIhtm_qSQ"/>
          </ownedEnd>
        </packagedElement>
      </packagedElement>
    </packagedElement>
  </packagedElement>
 </packagedElement>
 <packagedElement xmi:type="uml:PrimitiveType" xmi:id="_GxO5MMHiEd2UNuYhQ_ApoA"
name="String"/>
 <packagedElement xmi:type="uml:PrimitiveType" xmi:id="_h_R9AIFPEd6-295AH3icIA"
name="int"/>
</uml:Package>
```

Appendix B

The following are the AWL files generated by the AWL Parser integrated to the UML to AWSOME tool.

AWL file generated from simple.uml

```
package tudresden is
  package ocl20 is
    package pivot is
      package examples is
        package simple is
          class Person is
            public name : String := "y";
            public age : int;
          end class;
          class Student is Person with
          end class;
          class Professor is Person with
          end class;
        end package;
      end package;
    end package;
  end package;
  type OneToMany is range 1 .. *;
  type ZeroToMany is range 0 .. *;
  type Mandatory is range 1 .. 1;
  type Optional is range 0 .. 1;
  type int is range 0 .. *;
  type String is abstract;
end package;
```

AWL file generated from royalsandloyals.uml

```
package tudresden is
  package ocl20 is
    package pivot is
      package examples is
        package royalsandloyals is
          class Burning is Transaction with
          end class;
          class Customer is
            public age : Integer;
            public dateOfBirth : Date;
```

```
    public isMale : Boolean;
    public name : String;
    public title : String;
    public procedure birthdayHappens()
    public function getAge()  : Integer
  end class;
  class CustomerCard is
    public color : Color;
    public printedName : String;
    public valid : Boolean;
    public validFrom : Date;
    public validThru : Date;
  end class;
  class Earning is Transaction with
  end class;
  class LoyaltyAccount is
    public points : Integer;
    public totalPointsEarned : Integer;
    public number : Integer;
    public function burn(points : in Integer)  : Integer
    public function earn(points : in Integer)  : Integer
    public function getCustomerName()  : String
    public function isEmpty()  : Boolean
  end class;
  class LoyaltyProgram is
    public name : String;
    public procedure addService(aPartner : in ProgramPartner, aLevel : in ServiceLevel,
aService : in Service)
    public function enroll(c : in Customer)  : Boolean
    public function getName()  : String
    public function getServices()  : Service
  end class;
  class Membership is
  end class;
  class ProgramPartner is
    public name : String;
    public numberOfCustomers : Integer;
  end class;
  class Service is
    public condition : Boolean;
    public pointsBurned : Integer;
    public pointsEarned : Integer;
    public description : String;
    public serviceNr : Integer;
    public function calcPoints()  : Integer
    public procedure upgradePointsEarned(amount : in Integer)
```

```
end class;
class ServiceLevel is
  public name : String;
end class;
class Transaction is
  public amount : Double;
  public points : Integer;
  public date : Date;
  public function getProgram()  : LoyaltyProgram
end class;
association Customer_CustomerCards is
  role owner : Customer multiplicity Mandatory;
  role cards : CustomerCard multiplicity ZeroToMany;
end association;
association Customers_LoyaltyPrograms is
  role participants : Customer multiplicity ZeroToMany;
  role programs : LoyaltyProgram multiplicity ZeroToMany;
end association;
association CustomerCard_Membership is
  role card : CustomerCard multiplicity Mandatory;
  role membership : Membership multiplicity Mandatory;
end association;
association CustomerCard_Transactions is
  role card : CustomerCard multiplicity Mandatory;
  role transactions : Transaction multiplicity ZeroToMany;
end association;
association LoyaltyAccounts_Membership is
  role membership : Membership multiplicity Mandatory;
  role accounts : LoyaltyAccount multiplicity ZeroToMany;
end association;
association LoyaltyAccount_Transactions is
  role transactions : Transaction multiplicity OneToMany;
  role account : LoyaltyAccount multiplicity Mandatory;
end association;
association LoyaltyPrograms_ProgramPartners is
  role programs : LoyaltyProgram multiplicity OneToMany;
  role partners : ProgramPartner multiplicity OneToMany;
end association;
association LoyaltyProgram_ServiceLevels is
  role program : LoyaltyProgram multiplicity Mandatory;
  role levels :  ordered ServiceLevel multiplicity OneToMany;
end association;
association LoyaltyProgram_Membership is
  role membership : Membership multiplicity Mandatory;
  role program : LoyaltyProgram multiplicity Mandatory;
end association;
```

```
            association Memberships_ServiceLevel is
              role memberships : Membership multiplicity OneToMany;
              role currentLevel : ServiceLevel multiplicity Mandatory;
            end association;
            association ProgramPartner_Services is
              role partner : ProgramPartner multiplicity Mandatory;
              role deliveredServices : Service multiplicity ZeroToMany;
            end association;
            association Services_ServiceLevel is
              role availableServices : Service multiplicity ZeroToMany;
              role level : ServiceLevel multiplicity Mandatory;
            end association;
            association Service_Transaction is
              role service : Service multiplicity Mandatory;
              role transaction : Transaction multiplicity Mandatory;
            end association;
            class Date is
              public now : Date;
              public nowString : String;
              public function isAfter(aDate : in Date)  : Boolean
              public function isBefore(aDate : in Date)  : Boolean
              public class function now()  : Date
              public class function nowAsString()  : String
              public function toString()  : String
            end class;
            type Color is (silver, gold);
          end package;
        end package;
      end package;
    end package;
    type OneToMany is range 1 .. *;
    type ZeroToMany is range 0 .. *;
    type Mandatory is range 1 .. 1;
    type Optional is range 0 .. 1;
    type Double is delta 0.0010 range 0.0 .. *;
    type Integer is range 0 .. *;
    type String is abstract;
    type void is abstract;
end package;
```

AWL file generated from test1.uml

```
package tudresden is
```

```
  package ocl20 is
    package pivot is
      package examples is
        package test1 is
          class Person is
            public name : String := "John";
            public age : int;
            public socialSecurityNumber : int := 1234567890;
            public grade : Double;
            public function getAge()  : int_sub1
            public function getGrade()  : Double_sub2
          end class;
          class Student is Person with
          end class;
          class Professor is Person with
          end class;
        end package;
      end package;
    end package;
  type OneToMany is range 1 .. *;
  type ZeroToMany is range 0 .. *;
  type Mandatory is range 1 .. 1;
  type Optional is range 0 .. 1;
  type int is range 0 .. *;
  type String is abstract;
  type Double is delta 0.0010 range 0.0 .. *;
  type int_sub1 is range 1 .. 100;
  type Double_sub2 is delta 0.0010 range 0.0 .. 4.0;
end package;
```

AWL file generated from test2.uml

```
package tudresden is
  package ocl20 is
    package pivot is
      package examples is
        package test2 is
          class Person is abstract
            public name : String;
          end class;
          class Student is Person with
            public grade : int;
          end class;
          assocobject Married is
            role Husband : Person multiplicity Optional;
```

```
            role Wife : Student multiplicity Optional;
            public date : Date;
            public procedure marry(p : in Person, s : in Student, d : in Date)
          end assocobject;
        end package;
      end package;
    end package;
  end package;
  type OneToMany is range 1 .. *;
  type ZeroToMany is range 0 .. *;
  type Mandatory is range 1 .. 1;
  type Optional is range 0 .. 1;
  type String is abstract;
  type Date is abstract;
  type int is range 0 .. *;
end package;
```

AWL file generated from test3.uml

```
package tudresden is
  package ocl20 is
    package pivot is
      package examples is
        package test3 is
          class School is
            public name : String;
          end class;
          class Student is School with
            public grade : int;
          end class;
          aggregation School_Student is
            parent s : School multiplicity Mandatory;
            child stu : Student multiplicity ZeroToMany;
          end aggregation;
        end package;
      end package;
    end package;
  end package;
  type OneToMany is range 1 .. *;
  type ZeroToMany is range 0 .. *;
  type Mandatory is range 1 .. 1;
  type Optional is range 0 .. 1;
  type String is abstract;
  type int is range 0 .. *;
end package;
```

Appendix C

The following are the AWL files given as input to the AWSOME to OCL tool and the corresponding OCL files output.

simple.awl

```
package tudresden is
  package ocl20 is
    package pivot is
      package examples is
        package simple is
          class Person is
            public name : String := "y";
            public age : int;
          end class;
          class Student is Person with
          end class;
          class Professor is Person with
          end class;
        end package;
      end package;
    end package;
  end package;
  type OneToMany is range 1 .. *;
  type ZeroToMany is range 0 .. *;
  type Mandatory is range 1 .. 1;
  type Optional is range 0 .. 1;
  type int is range 0 .. *;
  type String is abstract;
end package;
```

OCL constraints from simple.awl

```
package tudresden::ocl20::pivot::examples::simple
context Person::name
init: 'y'
endpackage
```

royalsandloyals.awl

```
package tudresden is
  package ocl20 is
    package pivot is
      package examples is
```

```
package royalsandloyals is
  class Burning is Transaction with
  end class;
  class Customer is
    public age : Integer;
    public dateOfBirth : Date;
    public isMale : Boolean;
    public name : String;
    public title : String;
    public procedure birthdayHappens()
    guarantees
        age' = age + 1
    public function getAge()  : Integer
    invariant age >= 18
  end class;
  class CustomerCard is
    public color : Color;
    public printedName : String;
    public temp1 : String;
    public valid : Boolean := true;
    public validFrom : Date;
    public validThru : Date;
    invariant (validFrom.isBefore(validThru))
                    and temp1 = owner.title.concat(" ")
                    and printedName = temp1.concat(owner.name)
                        and (valid => (this.validFrom.isBefore(Date.now()) = true
and this.validThru.isAfter(Date.now()) = true))
                                    and (not valid =>
(this.validFrom.isBefore(Date.now()) = false and this.validThru.isAfter(Date.now()) = false))
  end class;
  class Earning is Transaction with
  end class;
  class LoyaltyAccount is
    public points : Integer := 0;
    public totalPointsEarned : Integer;
    public number : Integer;
    public function burn(points : in Integer)  : Integer
    public function earn(points : in Integer)  : Integer
    public function getCustomerName()  : String
    guarantees
        getCustomerName = membership.card.owner.name
    public function isEmpty()  : Boolean
    guarantees
        isEmpty = (points = 0)
     invariant points > 0 => exists(t : Transaction)(t in transactions => t.points > 0)
  end class;
```

```
class LoyaltyProgram is
  public name : String;
  public procedure addService(aPartner : in ProgramPartner, aLevel : in ServiceLevel,
aService : in Service)
  assumes aPartner in partners and aLevel in levels
  guarantees aService in partners.deliveredServices' and aService in
levels.availableServices'
  public function enroll(c : in Customer)  : Boolean
  assumes c.name /= ""
  guarantees participants' = participants union {c}
  and membership' = membership
  public function getName()  : String
  public function getServices()  : Service
  invariant membership.currentLevel in levels
                and forall(p : ProgramPartner) (p in partners =>
p.deliveredServices.size() >= 1)
                and forall(s:Service) ((s in  partners.deliveredServices) =>
(s.pointsEarned = 0 and s.pointsBurned = 0)) => (membership.accounts.isEmpty())
end class;
class Membership is
      invariant currentLevel.name = "Silver" => (card.color = silver) and
                  currentLevel.name = "Gold" => (card.color = gold)
end class;
class ProgramPartner is
  public name : String;
  public numberOfCustomers : Integer;
  invariant numberOfCustomers = programs.participants.size()
                and deliveredServices.transaction.points.sum() < 10000
end class;
class Service is
  public condition : Boolean;
  public pointsBurned : Integer;
  public pointsEarned : Integer;
  public description : String;
  public serviceNr : Integer;
  public function calcPoints()  : Integer
  //public procedure upgradePointsEarned(amount : in Integer)
  //guarantees  calcPoints'() = calcPoints() + amount
end class;
class ServiceLevel is
  public name : String;
end class;
class Transaction is
  public amount : Double;
  public points : Integer;
  public date : Date;
```

```
  public function getProgram()  : LoyaltyProgram
end class;
association Customer_CustomerCards is
  role owner : Customer multiplicity Mandatory;
  role cards : CustomerCard multiplicity ZeroToMany;
end association;
association Customers_LoyaltyPrograms is
  role participants : Customer multiplicity ZeroToMany;
  role programs : LoyaltyProgram multiplicity ZeroToMany;
end association;
association CustomerCard_Membership is
  role card : CustomerCard multiplicity Mandatory;
  role membership : Membership multiplicity Mandatory;
end association;
association CustomerCard_Transactions is
  role card : CustomerCard multiplicity Mandatory;
  role transactions : Transaction multiplicity ZeroToMany;
end association;
association LoyaltyAccounts_Membership is
  role membership : Membership multiplicity Mandatory;
  role accounts : LoyaltyAccount multiplicity ZeroToMany;
end association;
association LoyaltyAccount_Transactions is
  role transactions : Transaction multiplicity OneToMany;
  role account : LoyaltyAccount multiplicity Mandatory;
end association;
association LoyaltyPrograms_ProgramPartners is
  role programs : LoyaltyProgram multiplicity OneToMany;
  role partners : ProgramPartner multiplicity OneToMany;
end association;
association LoyaltyProgram_ServiceLevels is
  role program : LoyaltyProgram multiplicity Mandatory;
  role levels :  ServiceLevel multiplicity OneToMany;
end association;
association LoyaltyProgram_Membership is
  role membership : Membership multiplicity Mandatory;
  role program : LoyaltyProgram multiplicity Mandatory;
end association;
association Memberships_ServiceLevel is
  role memberships : Membership multiplicity OneToMany;
  role currentLevel : ServiceLevel multiplicity Mandatory;
end association;
association ProgramPartner_Services is
  role partner : ProgramPartner multiplicity Mandatory;
  role deliveredServices : Service multiplicity ZeroToMany;
end association;
```

```
        association Services_ServiceLevel is
          role availableServices : Service multiplicity ZeroToMany;
          role level : ServiceLevel multiplicity Mandatory;
        end association;
        association Service_Transaction is
          role service : Service multiplicity Mandatory;
          role transaction : Transaction multiplicity Mandatory;
        end association;
        class Date is
          public now : Date;
          public nowString : String;
          public function isAfter(aDate : in Date)  : Boolean
          public function isBefore(aDate : in Date)  : Boolean
          public class function now()  : Date
          public class function nowAsString()  : String
          public function toString()  : String
        end class;
        type Color is (silver, gold);
      end package;
    end package;
  end package;
 end package;
 type OneToMany is range 1 .. *;
 type ZeroToMany is range 0 .. *;
 type Mandatory is range 1 .. 1;
 type Optional is range 0 .. 1;
 type Double is delta 0.0010 range 0.0 .. *;
 type Integer is range 0 .. *;
 type String is abstract;
 type void is abstract;
end package;
```

OCL constraints from royalsandloyals.awl

```
package tudresden::ocl20::pivot::examples::royalsandloyals


context Customer::birthdayHappens()
post: (age)=((age@pre)+(1))
context Customer
inv: (age)>=(18)

context CustomerCard::valid
init: true
context CustomerCard
```

```
inv:
((((validFrom.isBefore(validThru))and((temp1)=(owner.title.concat(''))))and((printedName)=(te
mp1.concat(owner.name))))and((valid)implies(((self.validFrom.isBefore(Date.now()))=(true))an
d((self.validThru.isAfter(Date.now()))=(true)))))and((not(valid))implies(((self.validFrom.isBefor
e(Date.now()))=(false))and((self.validThru.isAfter(Date.now()))=(false))))


context LoyaltyAccount::points
init: 0
context LoyaltyAccount::getCustomerName()
post: (result)=(membership.card.owner.name@pre)
context LoyaltyAccount::isEmpty()
post: (result)=((points@pre)=(0))
context LoyaltyAccount
inv: ((points)>(0))implies(transactions->exists(t|((transactions)-
>includes(t))implies((t.points)>(0))))

context LoyaltyProgram::addService(aPartner: ProgramPartner, aLevel: ServiceLevel, aService:
Service)
pre: ((partners)->includes(aPartner))and((levels)->includes(aLevel))
post: (( partners.deliveredServices)->includes(aService@pre))and(( levels.availableServices)-
>includes(aService@pre))
context LoyaltyProgram::enroll(c: Customer)
pre: (c.name)<>('')
post: ((participants)=((participants@pre)-
>union(Set{c@pre})))and((membership)=(membership@pre))
context LoyaltyProgram
inv: ((((levels)->includes(membership.currentLevel))and(partners->forAll(p|((partners)-
>includes(p))implies((p.deliveredServices->size())>=(1)))))and(partners.deliveredServices-
>forAll(s|((partners.deliveredServices)-
>includes(s))implies(((s.pointsEarned)=(0))and((s.pointsBurned)=(0))))))implies(membership.ac
counts->isEmpty())

context Membership
inv:
(((currentLevel.name)=('Silver'))implies(((card.color)=(Color::silver))and((currentLevel.name)=
('Gold'))))implies((card.color)=(Color::gold))

context ProgramPartner
inv: ((numberOfCustomers)=(programs.participants-
>size()))and((deliveredServices.transaction.points->sum())<(10000))

endpackage
```

test1.awl

```
package tudresden is
  package ocl20 is
    package pivot is
      package examples is
        package test1 is
          class Person is
            public name : String := "John";
            public age : int;
            public socialSecurityNumber : int := 1234567890;
            public grade : Double;
            public function getAge()  : int_sub1
            public function getGrade()  : Double_sub2
                  guarantees grade < 4.0 and grade > 0.0
          end class;
          class Student is Person with
          end class;
          class Professor is Person with
          end class;
        end package;
      end package;
    end package;
  end package;
  type OneToMany is range 1 .. *;
  type ZeroToMany is range 0 .. *;
  type Mandatory is range 1 .. 1;
  type Optional is range 0 .. 1;
  type int is range 0 .. *;
  type String is abstract;
  type Double is delta 0.0010 range 0.0 .. *;
  type int_sub1 is range 1 .. 100;
  type Double_sub2 is range 0.0 .. 4.0;
end package;
```

OCL constraints from test1.awl

```
package tudresden::ocl20::pivot::examples::test1

context Person::name
init: 'John'
context Person::socialSecurityNumber
init: 1234567890
context Person::getGrade()
post: ((grade@pre)<(4.0))and((grade@pre)>(0.0))

endpackage
```