Integration of ASM with Event-B RODIN

By
Innayat Ullah
2011-NUST-MS-CCS 02

Supervisor
Dr. Osman Hasan
Department of Computing

A thesis submitted in partial fulfillment of the requirements for the degree of Masters in Computer and Communication Security (MS CCS)

In
School of Electrical Engineering and Computer Science,
National University of Sciences and Technology (NUST),
Islamabad, Pakistan.

(July 2015)

Approval

It is certified that the contents and form of the thesis entitled "**Integration of ASM with Event-B RODIN**" submitted by **Innayat Ullah** have been found satisfactory for the requirement of the degree.

Advisor: **Dr. Osman Hasan**
Signature: _____

Date: _____

Committee Member 1: **Dr. Abdul Ghafoor Abbassi**

Signature: _____
Date: _____

Committee Member 2: **Dr. Atif Mashkoor**

Signature: _____
Date: _____

Committee Member 3: **Dr. Sohail Iqbal**

Signature: _____
Date: _____

Abstract

This thesis is related to the integration of diverse formal verification tools to harness their capabilities for better modeling and verification of complex systems. In this effort we worked on integrating Abstract State Machines (ASMs) with Event-B using the RODIN platform.

Event-B is a formal verification tool that is used for real-time system-level modeling and analysis. Event-B is primarily based on the B Method, which is a formal method for the development of program code from a specification in the Abstract Machine Notation. Event-B uses sets and first-order predicate logic as its foundations. It allows different abstraction levels to be shown using refinements. Event-B is supported by the RODIN platform that uses the Eclipse IDE. ASMs are very easy to develop and manipulate but their verification is an issue. The tool set is not that rich in verification. Whereas RODIN toolset is very rich in Verification and Validation of the requirements.

In particular, in the thesis, we define Event-B and ASMs and the technique we devised to perform their translation (ASM to Event-B). Moreover there is an architectural and visual design of the tool-set that we propose to develop for this translation.

Certificate of Originality

I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any degree or diploma at NUST SEECS or at any other educational institute, except where due acknowledgement has been made in the thesis. Any contribution made to the research by others, with whom I have worked at NUST SEECS or elsewhere, is explicitly acknowledged in the thesis.

I also declare that the intellectual content of this thesis is the product of my own work, except for the assistance from others in the project's design and conception or in style, presentation and linguistics which has been acknowledged.

Author Name: **Innayat Ullah**
Signature: _____

Acknowledgment

I'm very thankful to Almighty Allah for giving me the strength and intellect to complete this thesis. I would like to say special thanks to my supervisor **Dr. Osman Hasan** for his utmost support and help to carry out the tasks. Without his help I wouldn't have been able to complete it. I'm also very thankful to the committee members for bearing with me in this research. Their help, guidance and support was a light in the dark for me. Specially **Dr. Atif Mashkoor** who was the main driving force behind this research. Thank you sir. Thanks to my family my wife and friends for their support, it was really encouraging for me in the course of my research. Thank you all.

DEDICATED
TO
MY LOVING PARENTS
MY WIFE
&
MY SUPERVISOR

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction and Motivation

"*O Lord! Increase me in knowledge.*"

Al-Quran (20:114)

This chapter presents introduction to the topic. The motivation behind choosing this topic for research. A summary of the thesis will be presented to give an abstract picture of the whole work. First we'll describe the problem statement for the research topic and our approach to address the problem. Then we'll let the readers know about the organization of the thesis chapters and their content. We will also shed some light on the research contribution attempted for a related field of this topic

Software and hardware system designing and development is major system development area these days. Computer systems are being used to design safety critical systems. These safety critical system need to be fault-less for them to be safe both financially and from life point of view. This brings Forma Methods to the front for the engineers and the scientists. Formal method is a fault avoidance and error detecting technique that uses mathematical techniques and theoretical computer science fundamentals such as:

- Particular Calculi
- Formal Languages
- Automata Theory
- Program Semantics
- Type Systems
- Algebraic Data

Therefore, Formal method can become really complex for which easy-to-use tools are needed. A relatively new tool for formal method is Event-B which is used for system-level modeling and analysis. Event-B is a simplified and extended version of the B Method. B is a formal method for the development of program code from a specification in the Abstract Machine Notation. (for,). Event-B uses sets and first-order predicate logic. It allows different abstraction levels to be shown using refinements. Event-B is provided support from the platform Rodin that uses the Eclipse IDE. This thesis is related to integration of diverse formal verification tools to harness their capabilities for better modeling and verification of complex systems. In this effort we worked on integrating Abstract State Machines (ASMs) with Event-B in RODIN platform. In the document we will define Event-B and ASMs and the technique we devised to perform their translation (ASM to Event-B). Moreover there will be an architectural and visual design of the toolset that we propose to develop for this translation.
Dr. Atif Mashkoor is Scientific Head Rigorous Methods in Software Engineering at Software Competence Center Hagenberg, Austria. He's been an active researcher in the field of formal verification. During their applied research at the Center they felt the need of a tool that can integrate ASM with Event-B. ASMs are very easy to develop and manipulate but their verification is an issue. The tool set is not that rich in verification. Whereas RODIN toolset is very rich in Verification and Validation of the requirements. There was a presentation given by *Egon Börger and Laurent Voisin* that described

Defining ASMs as Event-B Machines and Vice Versa. That was the main motivation behind this thesis work.

1.1 Problem Statement

There are various powerful techniques and tools, such as simulators, model checkers and theorem provers that have been developed for the individual state-based formal methods, such as Event-B and ASM. It would be very interested and beneficial for industry if we can use the simplicity and ease of ASM combined with the strength and power of tool-set of RODIN (Event-B)

1.2 Research Contribution

Model creation using Event-B was practiced fo a Model of *Airplane's Landing Gear System*(Boniol and Wiels,). With this model we participated in ABZ conference 2014 call for papers case study track, unfortunately our solution was rejected.

1.3 Thesis Organization

This thesis is organized in a systematic way so that the reader can get a grasp of the idea and the implementation. First of all we will present literature review in Chapter 2. Then we will have a look at the research methodology followed to carry out the research in Chapter 3. In Chapter 4 we will discuss the design and architecture for the proposed plug-in. Chapter 5 discusses the implementation details of the plug-in. Chapter 6 is the conclusion. Then we have an appendix at the end, it contains the code of main classes of the plug-in.

Chapter 2

Literature Review

This chapter presents the work that is already done in this field. A look at similar researches done in the field of integrating different tools. We will look at the documentation and resources that we consulted in order to get hold of Event-B and ASM machine development.

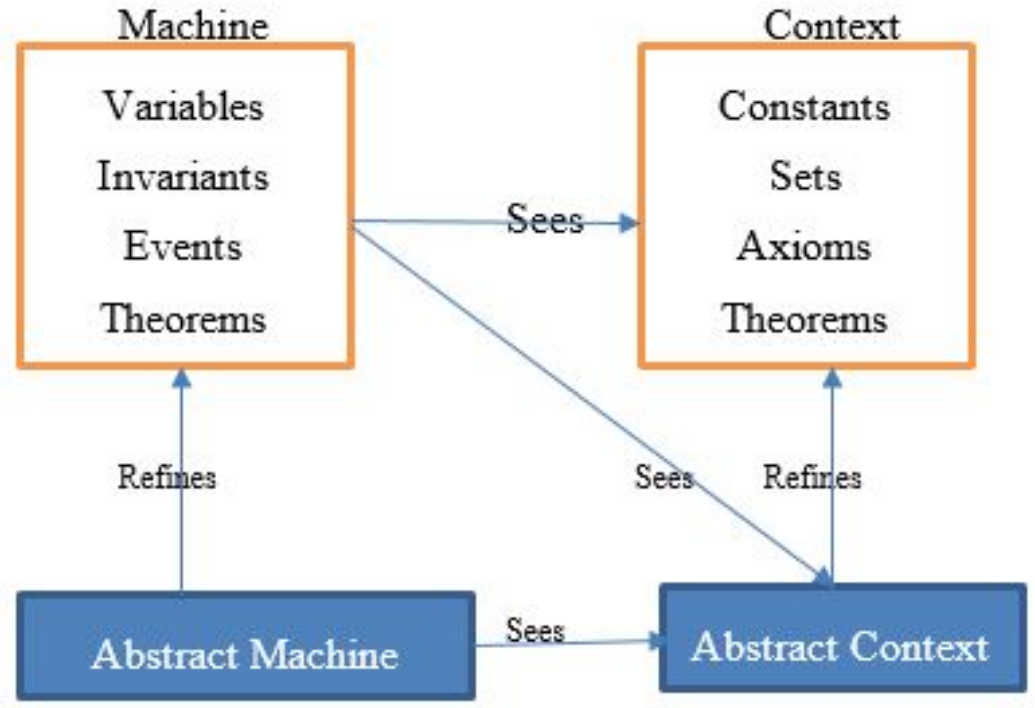

Figure 2.1: Event-B Machine Context relationship

2.1 Getting to know Event-B

Event-B is a simplified version of B Method for formal verification. B Method is aimed at *Software Development* whereas Event-B is aimed at *System design*(Jean-Raymond Abrial,). RODIN is the integrated environment that is mainly used with Event-B construct definition and verification. Event-B consists of two types of constructs Machine and Context. Machines and Contexts are refined in the process of generating a concrete machine from an abstract machine. A machine M* is a refinement of machine M and practically M* M in semantic and working. A machine M* refines M and can see multiple contexts Ci, C*i, . , Ck. As machines are refined from their abstract to concrete machines similarly Contexts are also refined from their abstract representation C to their more concrete representation C*. As shown in the figure 2.1 An Event-B machines consists of Variables (v) Invariants (Inv) Events (eve) and Theorems(C. Mtayer J.-R. Abrial,). While refining the machines (also called models) all the invariants, theorems and axioms must hold valid in the refined machine. An event in a machine basically introduces a change in the state of the machine. In order to introduce a change in the

Event.jpg

Event $Take\_Off \widehat{=}$

when

grd1 : $UplockBox\_Left = Locked\_Down \wedge UplockBox\_Right = Locked\_Down \wedge UplockBox\_Front = Locked\_Down$

grd2 : $Shock\_Absorber\_Left = OnGround \wedge Shock\_Absorber\_Right = OnGround \wedge Shock\_Absorber\_Front = OnGround$

grd3 : $Plane\_Flying = FALSE$

then

act1 : $Plane\_Flying := TRUE$

act2 : $Shock\_Absorber\_Left := InFlight$

act3 : $Shock\_Absorber\_Right := InFlight$

act4 : $Shock\_Absorber\_Front := InFlight$

end

Figure 2.2: Event-B Event

state of the machine there are different guards that can be applied to ensure the validity of the post conditions for the event to occur. There are two ways to add guards in an event. a) Any Where b) Where, in Any Where method a guard G is validated for any local variable x,y Whereas in Where type of guard a guard G is validated for global constants and/or variables. An example of an Event-B event is shown in figure 2.2

Once the machine is defined the verification of machines correctness is done through Proof Obligations (POs). In RODIN environment there are different POs automatically generated for the defined machine. Some of the proof obligations are automatically discharged others need manual discharging. Manual discharging includes but not limited to theorems and guard strengthening. For a machine to discharge all the POs it should display following properties(C. Mtayer J.-R. Abrial,):

1. All the theorems should be preserved
2. All the invariants should be preserved
3. Variables should be initialized in the initialization step
4. There is no deadlock at any point of time in machines execution

conditional flow.jpg

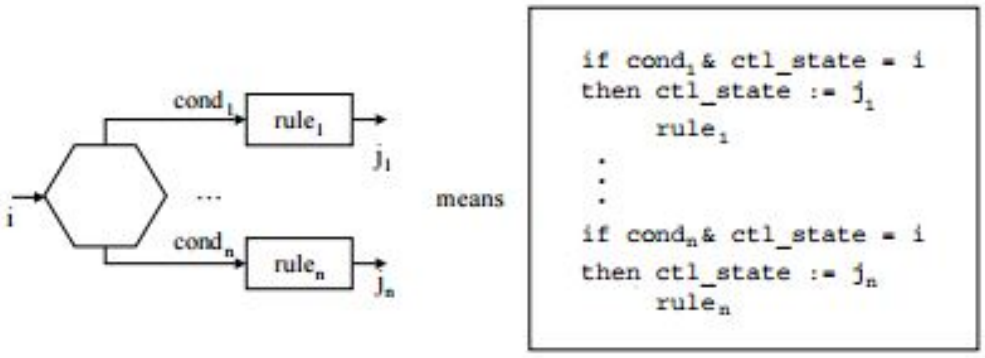

Figure 2.3: Abstract State Machine Conditional State Transition Flow

2.2 Getting to know ASML

Abstract state machines are mathematical models of real life systems. There are different states of the systems and a state transition steps takes the machine from one system state to another. These state transitions are governed by rules and constraints. Depending upon the rules a transition can take place or the other. Figure 2.3 shows a conditional state transition flow for a typical state machine.

A common language to define abstract state machines is Abstract State Machine Language (AsmL). It is .net based tool to define and refine ASMs. AsmL has its own fixed vocabulary defined for state machine definition. This vocabulary can be used to define ASMs. There are mainly two parts of an ASM a) State Operations b) State Variables. Operations are the control logic that is applied to the input state to produce a valid output state. Whereas state variables are the containers to store different values that create a state. These values can be number or strings. Each machine has a run that is the sequence of the states that are interlinked to completely depict the machine behavior. A machine consists of different statements such as conditions and assignments. A typical conditional statement is like *if cond1 then update* and the updates are the assignments of new state values to state variables and syntactically are represented as a := b. A typical run of a machine is graphically represented in the figure 2.4.

An abstract state machine in AsmL consists of following components

1. Import

run.jpg

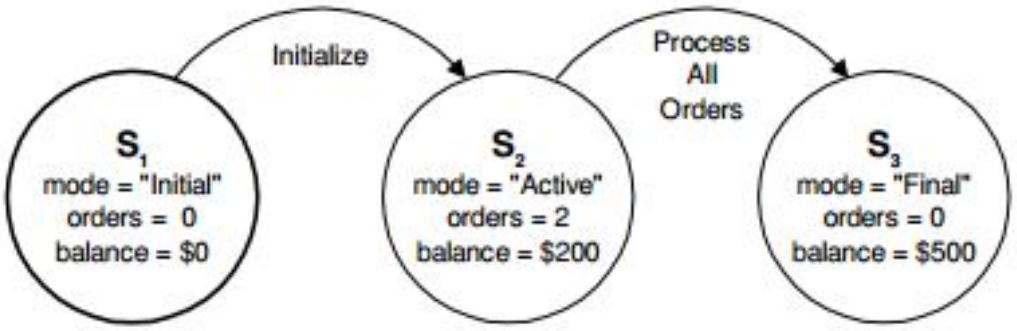

Figure 2.4: Abstract State Machine Run

2. Signature
3. Definition
4. Default initialization state

In AsmL the name of the asm should be same as the name of the asm file(Microsoft-Research,). There are standard libraries already developed for AsmL that can be used by importing them to the asm file using import keyword and path to standard library. Then there are signatures of different state variables. Definitions contain different rules and functions for the state transition of the machine. There should be atleast one main rule present in the machine in order for it to be a valid ASM. In the end there should be a default init state that should initialize variables with the initial values.

2.3 Event-B and SPIN

During my literature review I came across a very similar work. In this thesis the author integrated Event-B and Promela Language by translating Event-B models to Promela. (SPIN is a tool for promela language).

The author describes that he first converted Event-B entities into java objects and then used those java objects to transform them into Promela processes. There were some direct substitutions from Event-B to Promela as both are asynchronous. The machines from Event-B are represented as do loop in Promela. The author developed an architecture consisting of

Properties selector, Promela File Generator, UI and Promela Representation Manager. Using these components the author was able to successfully convert Event-B entities to Promela Processes.
This thesis was a great help in realizing our idea of integrating ASM with Event-B RODIN. Taking help from the thesis we were able to come up with an architecture similar to the one presented in this thesis. But we used a little different approach in intermediate objects. We used the XMI and XML files of both Event-B and ASML to translate. Moreover we have designed this plug-in as an Import wizard. As our aim is to import an ASM file from the system to the Event-B project as an Event-B entity.

Chapter 3

Research Methodology

This chapter presents the research methodology that we used to conduct this research. In this chapter we will also discuss other research methodologies and why we chose our methodology over others. A lot of that depends upon our research phases that lead us to our research approach. There were three major phases in our research. First: Identify the possibility of integration of ASM with Event-B, Second: Present an architecture for the transformation of ASM files to Event-B entities, Third: Implement the proposed architecture as a plug-in for RODIN platform

3.1 What is Research?

The word Research is composed of two syllables "*Re*" & "*Search*". In simple English "Re" means again and again whereas "Search" means finding or exploring something. Research is not only confined to science and technology but it is an active part of growth of a society. There are six key parts of research. First identify the issues, then do study on the identified issue to form some hypothesis, then perform experimentation based on that hypothesis, then observe the experimental results and analyze and compare them with other hypothesis and results with logical reasoning(S. Rajasekar,).

3.2 Difference between Research Method and Research Methodology

Research method and Research methodology are often mistaken to be the same terms but they are not. Research method are the tools, different procedures and algorithms, numerical and statistical analysis techniques whereas Research methodology is a systematic way to approach the problem in order to solve it(S. Rajasekar,).

3.3 Types of Research Method

As described in previous section there are different methods of performing research. There are mainly two major methods of research and then there is their sub division. Some research methods and their division is given below

3.3.1 Fundamental or Basic Research

Basic or fundamental research is also called "*theoretical research*(S. Rajasekar,). It is the study of natural phenomenon and may not lead to immediate application in real life but it gives new properties of different natural material or even discover new material. This is the basic scientific research with hypothesis and theories.

3.3.2 Applied Research

Applied research is the type of research in which a researcher applies his/her knowledge in a specific field(S. Rajasekar,). They tend to provide logical

reasoning and perform experiments in that particular field to analyze and compare different results to prove a hypothesis.

3.3.3 Normal Research

Normal research is a sub-part of *Basic Research*. In a particular field normal research is performed based upon the rules and procedures already defined by the scientists in that field. These rules and procedures are called *Paradigms*.

3.3.4 Revolutionary Research

In the process of normal research when there is some anomaly or unexpected results are realized. They open a new direction and make changes or create new paradigms for that field of research. This is an evolutionary process and the type of research is called *Revolutionary Research*.

3.3.5 Quantitative Research

A quantitative research is the research of quantity for a particular problem or in a general field. The research is carried out and the results are measured quantitatively (in numbers).

3.3.6 Qualitative Research

Unlike quantitative research qualitative research is concerned with the quality. It is non-numerical that is it doesn't involve numbers. It is theoretical and is expressed in words. It tries to answer *why* and *what* kind of questions.

3.4 Thesis Research Approach

In this thesis we divided our problem into three stages. First stage was to get to know the languages and their helping tools. In order to get to know Event-B I was involved in a research paper based on a case study, as a partner with two undergraduate students of SEECS. We worked on developing an Event-B model for *Airplane Landing Gear System*. Based upon the model we participated in a conference *ABZ'2014 CASE STUDY track*. Unfortunately our paper was not accepted but we were given a healthy feedback on the model and it's improvements by the reviewers. After getting to know the RODIN IDE and Event-B next task was to get to know ASMs and their structure. After getting to know the languages next task was to

Methodology.jpg

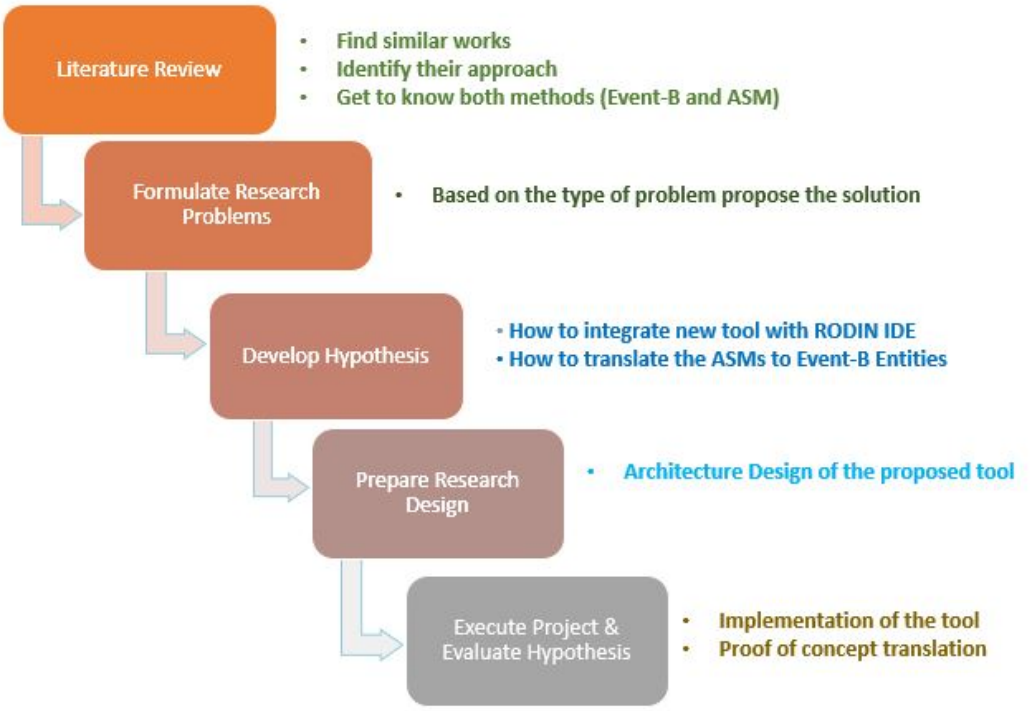

Figure 3.1: Research Methodology

find out the middle ground for the transformation. As the field of work is quite diverse we had to combine different research methods in order to get the results that we were striving for.

3.4.1 Defining the Research Area

This was the first step of our research. Dr. Atif Mashkoor has been involved in formal verification process for quite some time. He's been working with Event-B for modeling of different systems. The research problem and it's importance was discussed with my supervisor and with me. As it involved some development of the plug-in along with extensive research for finding the semantic translation between ASM and Event-B. We finally decided to go with this problem statement.

3.4.2 Literature Review

Literature review is the building block of any research. It provides the researcher the opportunity to explore what is going on in the selected research area. A researcher has to go through the research papers to find out what are the techniques that are being used in his/her field of research and what are their pros and cons. If there are any shortcomings that are identified. It helps the researcher to strengthen his/her research area and helps validate the problem statement.
For this particular thesis we tried to find out different similar work that have been done in the industry and academia. One similar work was translating Event-B to Promela language(MULLER,). It helped us allot to streamline our research problem and how to approach it. Further we had to get familiar with Event-B language and it's IDE and how the models are developed in this environment. A research paper was written in Case Study track of ABZ'14 conference. That was very instrumental in getting the grasp of Event-B models. Then we had to study ASM and how they are developed using AsmetaL.

3.4.3 Problem Statement

Our research area was already defined and the specifics of problem statement were laid down too. The research problem that we addressed in this thesis is

Various State based tools and methods are present in the industry and academia. It would be very interested and beneficial for industry if we can use the simplicity and ease of ASM combined with the strength and power of tool-set of RODIN (Event-B)

3.4.4 Hypothesis

The hypothesis after literature review was that there is no practical research done on this topic except some theories presented by *Egon Börger*. There should be a semantic translation from ASM constructs to Event-B constructs without changing the structure, meanings and working of the models.

3.4.5 Architecture Design

During this phase we designed the architecture of the tool that we had to build. The architecture of the tool was inspired by eclipse import wizard and Event-B to SPIN translation tool architecture.

3.4.6 Execution and Evaluation

In this phase of the research we developed the designed tool for RODIN. It is a wizard type tool that uses import file wizard to import the files into the Event-B project. Once the tool was developed we had to evaluate if it is generating the files in the correct way of not.

Chapter 4

System Design and Architecture

This chapter presents overall architecture of the plug-in. Here we will discuss different plug-in options that were available for us. Why we selected wizard type plug-in and its architectural design.

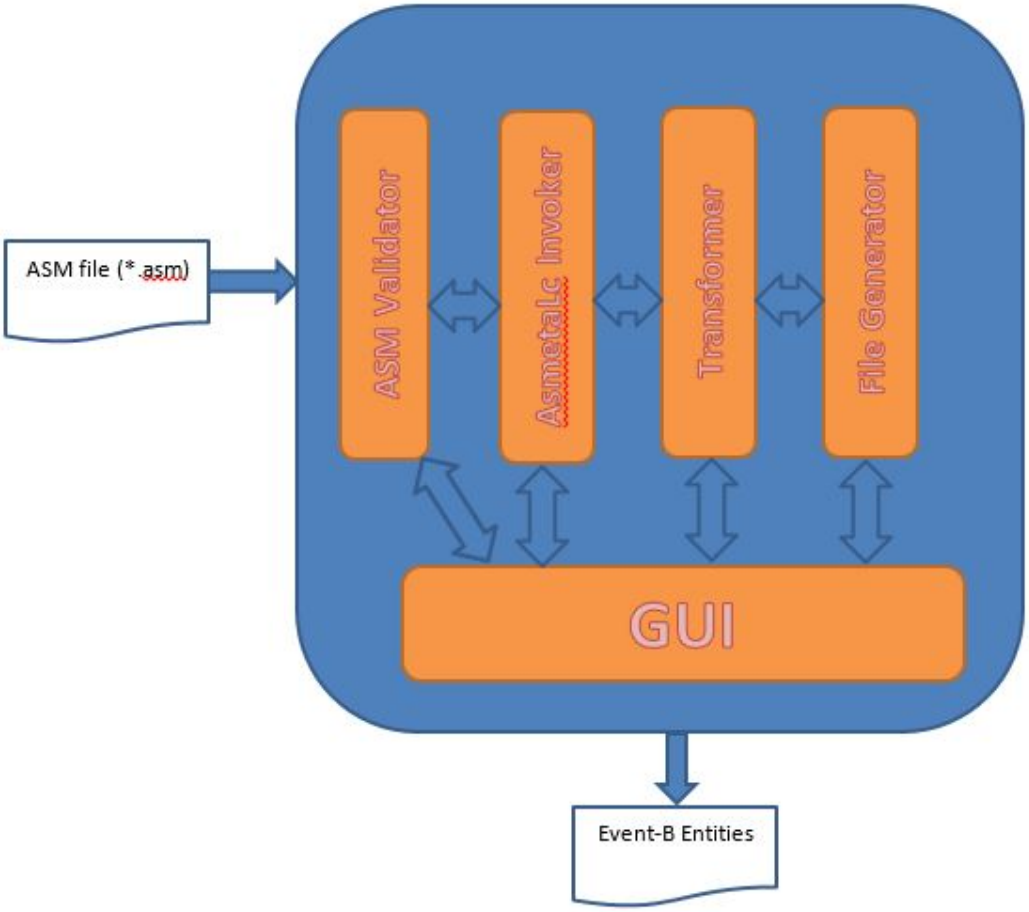

Figure 4.1: ASM to Event-B machine File generator module Architecture

4.1 Plugin Development in Eclipse

Eclipse platform provides rich support for Plugin as well as complete IDE development. There are different kinds of plug-ins that can be developed in Eclipse.

4.2 Architecture of Plugin

ASM to Event-B plug-in will plug itself into RODIN architecture. It will use extension points of Event-B, RODIN and RODIN GUI to plug and make itself functional in the Event-B RODIN domain. There are five major components of our plug-in. Figure 4.1 shows these components graphically.

1. GUI
2. ASM Validator
3. AsmetaLc Invoker
4. Transformer
5. File Generator

In the upcoming content we will discuss each component and it's responsibilities briefly. Their detailed working will be discussed in next chapter.

4.2.1 GUI

This component extends RODIN GUI to allow the user to select the files for conversion from ASM to Event-B machines. When the perspective is loaded it will show a popup window for the user to select the file that the user wants to convert to Event-B machine. It will import the file to a local storage where AsmetaLc is already stored.

4.2.2 ASM Validator

This component is responsible for validating the selected ASM file. There are a certain set of rules to define a valid ASM. It will do a schema comparison for the ASM. The Schema comparison will be done using an XML schema document. If all the schema requirements are met i.e. machine name is same as file name, there is atleast one main rule, and there is exactly one default initialization etc. Once the file is validated it will signal AsmetaLc invoker to perform further steps else it will show an error message on the GUI to let the user know that the ASM file is not a valid ASM file

4.2.3 AsmetaLc Invoker

This component is responsible for calling AsmetaLc to generate the intermediate XMI file that will be used by the Transformer to transform it to Event-B EMF file. AsmetaLc is a jar file that can be invoked from command line. The command to generate the XMI file from an ASM file is given below with their optional parameters.

```
java -jar AsmetaLc [-xmi] [-log <log4j file>]
   ↪ <model.asm>
    -xmi: generate the XMI output format.
```

```
-log: generate debug info.
<log4j file>: configuration file of the logger
<model.asm>: path name of the ASM spec.
```

4.2.4 Transformer

This component is responsible for generating the EMF that will be used for generating Event-B machine files. It will take XMI file generated by AsmetaLc invoker as the input file and will semantically translate the XMI file to EMF model. EMF model is the metamodel of Event-B machines that is designed specifically for the purpose of transforming machines from one syntax to another (cross tool support).

4.2.5 File Generator

This component is responsible for generating files from EMF Meta model of Event-B models. It is basically creating different required files in order to create a machine. A typical Event-B machine consists of five types of files .bcm, .bpr, .bpo, .bps and .bum. Each file contains relevant xml that defines the events, variables, invariants and axioms etc of the machine.

Chapter 5

Implementation

In this chapter we will have a look at the implementation details of the plug-in. We used Eclipse IDE to develop the ASM Import Wizard Plugin for RODIN. As RODIN is built on Eclipse it was the best choice for the plug-in development. Each individual java package, why it was required and it's implementation detail is discussed.

5.1 Wizard Project in Eclipse

First step was to create a project for our import wizard in eclipse. From file menu we selected "*Eclipse Plugin Project* as the type of project. As we needed some other views along with file import wizard we selected "*Customer plug-in wizard*" from the plug-in templates . The next step was to choose templates from Available Templates. We selected "**File Import Wizard**" and "**View**" from the templates and finished the "New plug-in project with custom templates" wizard. It created a project in Eclipse with following packages in it.

1. `edu.seecs.nust.asmtransformer`
2. `edu.seecs.nust.asmtransformer.importWizards`
3. `edu.seecs.nust.asmtransformer.views`

Further we had to translate the XMI files of ASM to concrete Java objects. Another package to store all the XMI entities was added. Then in order to generate the XML file for Event-B Entities there was a requirement of another package that can hold Event-B entities. These two packages are actually the models of ASM and Event-B respectively. The two more packages that were added to the project are listed below.

1. `edu.seecs.nust.asmtransformer.asm`
2. `edu.seecs.nust.asmtransformer.eventb`

In the upcoming sections we will have a look at each of these five packages and will discuss their responsibilities and intercommunication for translating ASMs to Event-B entities.

5.2 Package Implementation Details

In this section we will explore all the packages. We will see all the classes present in the package and their role in the wizard.

5.2.1 ASM Transformer

ASM transformer package is basically responsible for transforming ASM Java Objects to Event-B Java Objects. The class diagram of ASM Transformer is shown in Figure 5.3. There are a total of five(5) classes present in this Java package. These classes are:

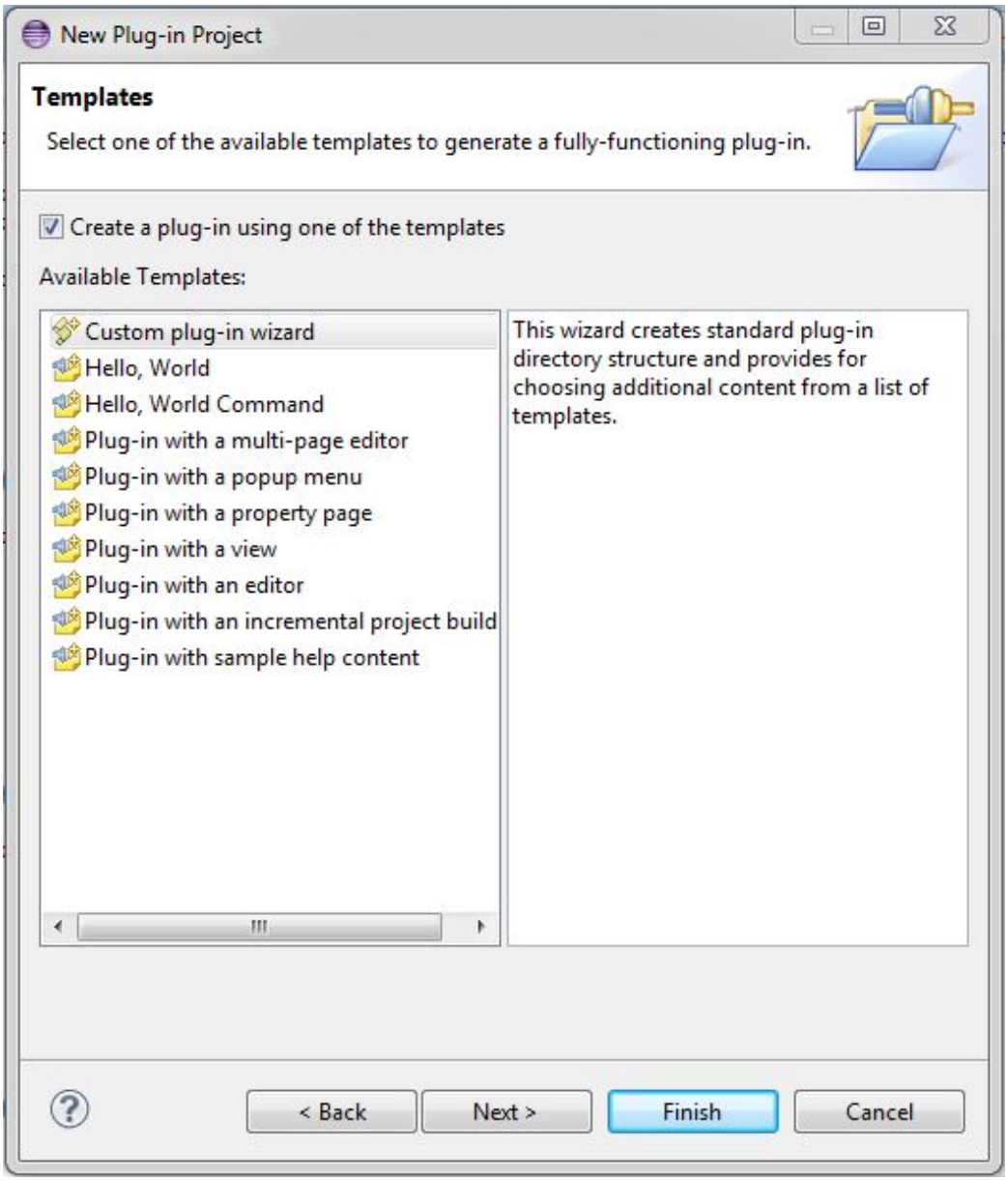

Figure 5.1: Plugin type selection

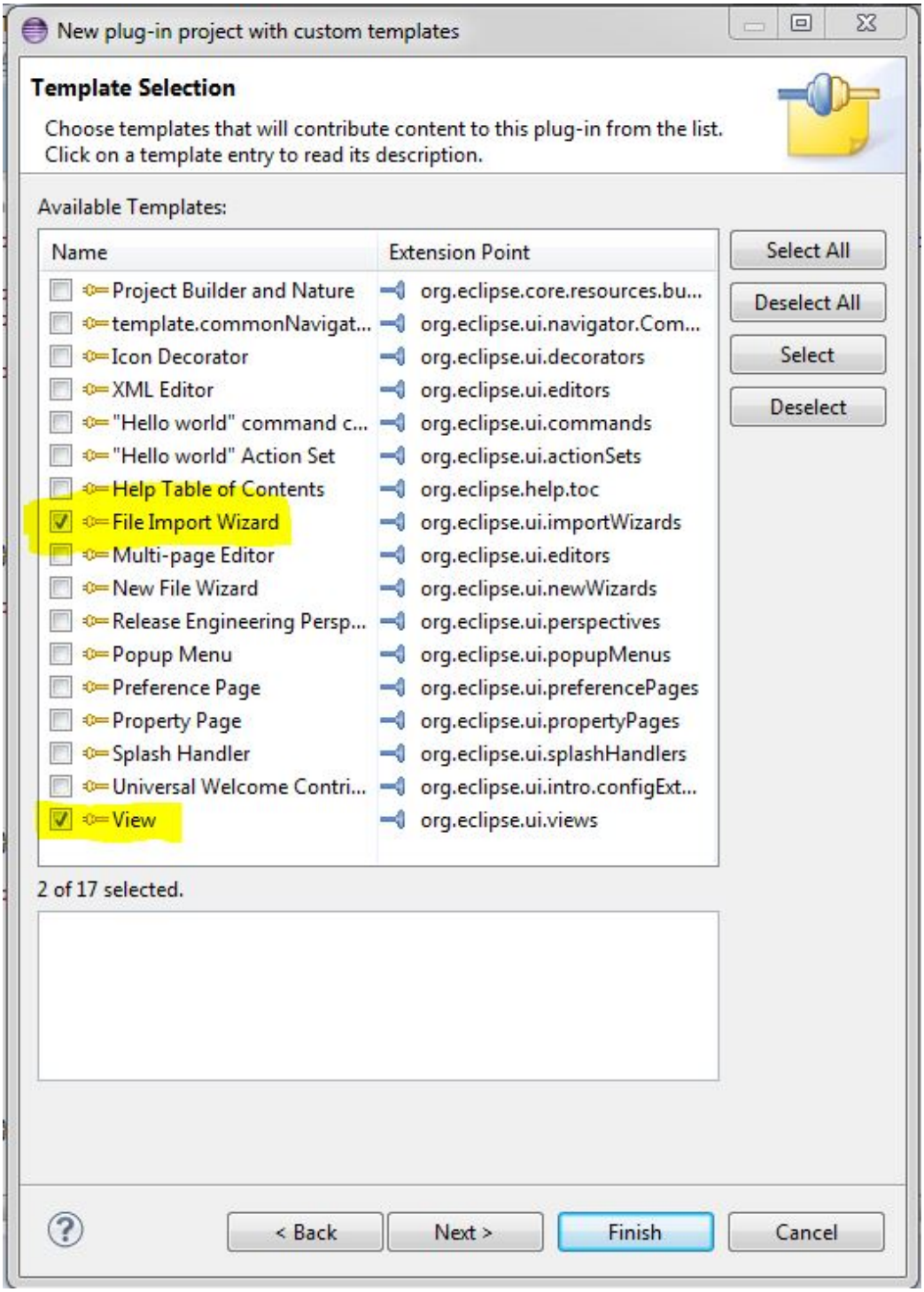

Figure 5.2: Custom wizard parts selection

1. `Activator.java`
2. `EventBFile.java`
3. `Transformer.java`
4. `Utilities.java`
5. `XmlGenerator.java`

Activator.java

This class is responsible for controlling the plug-in life cycle. It stores all the necessary information for the plug-in to activate itself and run in an IDE i.e. Eclipse, RODIN

EventBFile.java

This class is a data-model of Event-B entities. It stores two more Event-B structs "*Context*" and "*MachineFile*".

Transformer.java

This class is responsible for performing the transformation actions. It will also call other class operations i.e. GenerateXml operation for generating Event-B files.

Utilities.java

This class is a general utilities class for the whole project. It contains the common methods that are used at multiple locations within the project.

XmlGenerator.java

This class is responsible for generating XML file from the passed content with the passed name and extension.

5.2.2 ASM

ASM package contains the necessary data model classes to store the XMI in Java Object form. Each class has their own properties and/or sub entities. Figure 5.4 shows the class diagram of the package. There are seventeen (17) classes present in this package. These classes are:

Class Diagram.jpg

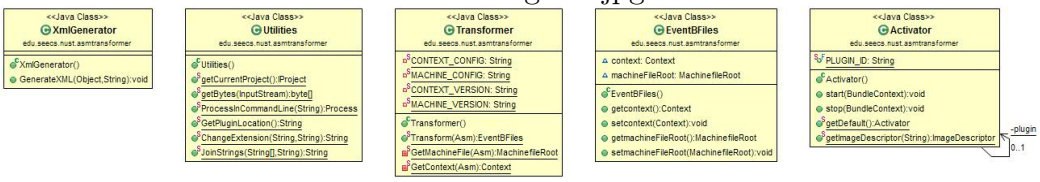

Figure 5.3: Custom wizard parts selection

1. `argument.java`
2. `Asm.java`
3. `body.java`
4. `bodySection.java`
5. `domain.java`
6. `function.java`
7. `element.java`
8. `functionDefinition.java`
9. `functionInitialization.java`
10. `headerSection.java`
11. `importClause.java`
12. `initialState.java`
13. `ruleDeclaration.java`
14. `signature.java`
15. `StaxAsmParser.java`
16. `structuredDomain.java`
17. `variable.java`

Asm.java

Asm.java class is the main class in this package. It is the main element of ASM XMI file. It stores all the other classes except StaxAsmParser.java. An Asm.java object represents an Abstract State Machine.

StaxAsmParser.java

This class is responsible for parsing the XMI file generated by the AsmetaLc.jar into ASM Java object. It reads the XMI files and fills an ASM object that is then used in the transformation process to generate Event-B Entities.

5.2.3 Event B

This package contains the data-model classes for the translated Event-B objects. Each class has it's own properties and sub entities. Figure 5.5 shows the class diagram for Event-B package. There are a total of seventeen (17) classes in this package. These classes are:

1. `action.java`
2. `axiom.java`
3. `carrierSet.java`
4. `constant.java`
5. `Context.java`
6. `contextFile.java`
7. `evet.java`
8. `extendsContext.java`
9. `guard.java`
10. `invariant.java`
11. `machineFile.java`
12. `MachineFileRoot.java`
13. `parameter.java`

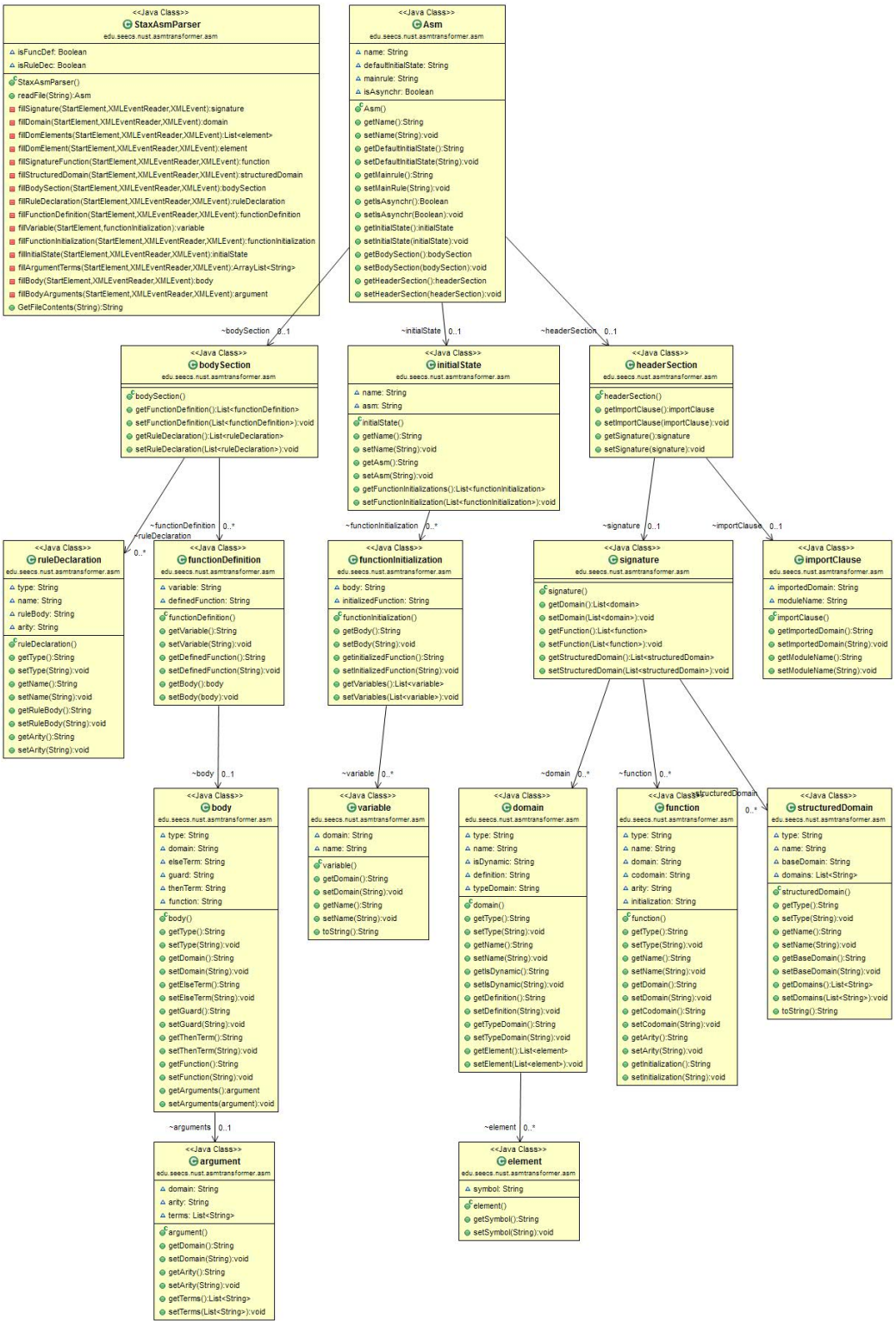

Figure 5.4: ASM Package Class Diagram

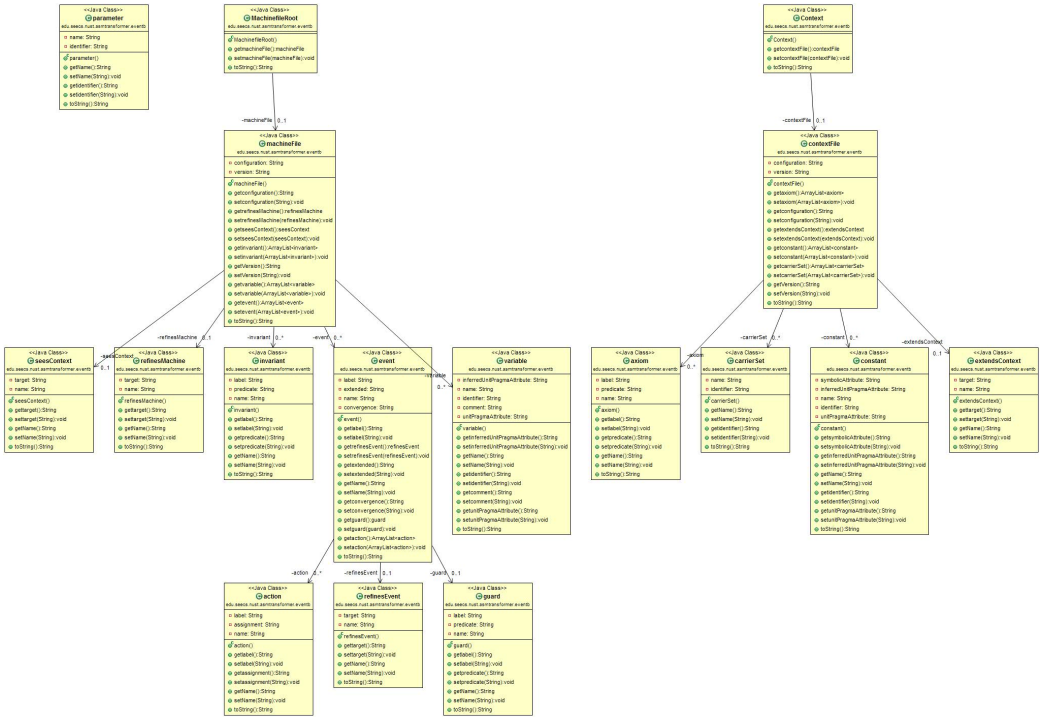

Figure 5.5: Event-B Package Class Diagram

14. `refinesEvent.java`
15. `refinesMachine.java`
16. `seesContext.java`
17. `variable.java`

Context.java

Context.java file is the main file for Event-B construct "*Context*". It stores the static components of Event-B model. It consists of lists of Constants, Axioms and Carrier Sets. This file is used to produce ".buc" xml file that is the basic file of an Event-B Context in RODIN platform.

MachineFileRoot.java

This file is the main file for Event-B construct "*Machine*". It stores the dynamic components of Event-B model like Variables, Events, Invariants and Theorems. This file is used to produce ".bum" xml file that is the xml representation of Event-B Machine Model in RODIN.

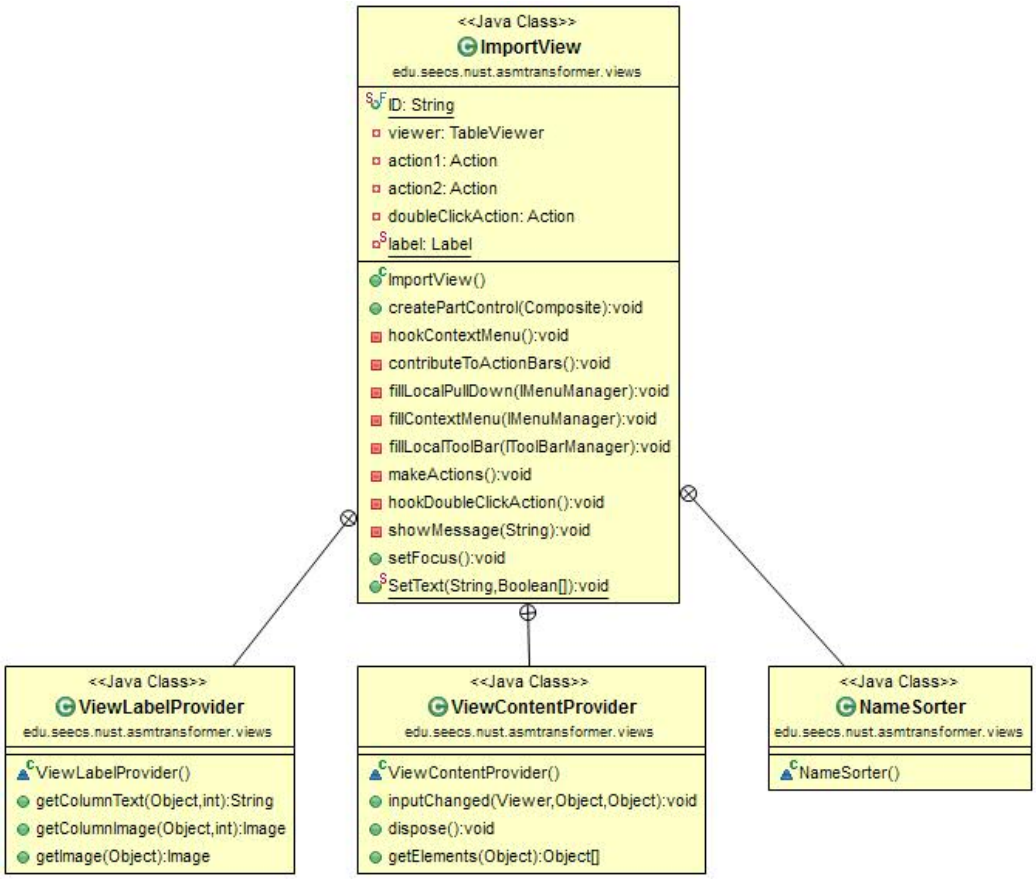

Figure 5.6: Views Package Class Diagram

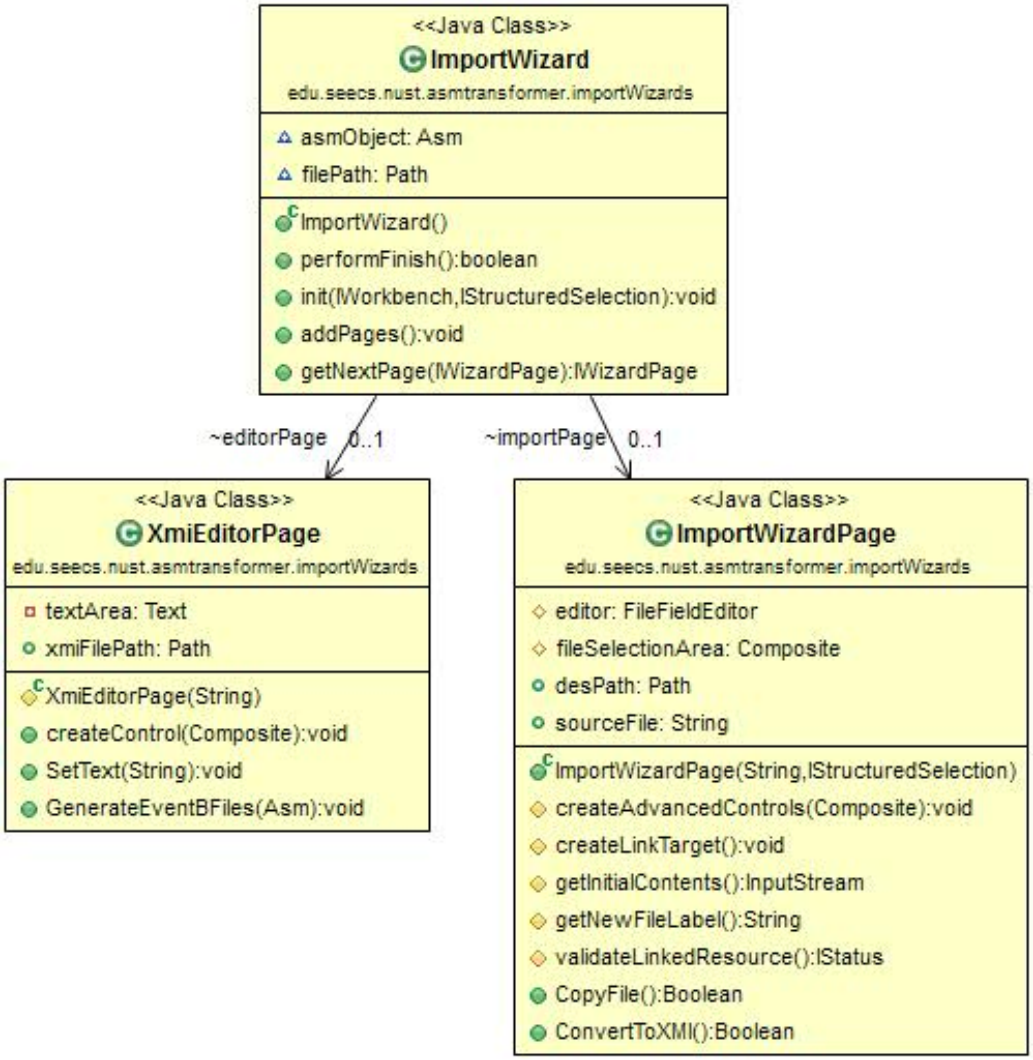

Figure 5.7: ImportWizard Package Class Diagram

5.2.4 Views

Views package contains the UI view related classes. These classes are used to display data/output to the user. The class diagram for the views package is shown in the figure 5.6. This package contains only one class `ImportView.java`.

5.2.5 Import Wizard

This is the main import wizard. It is used to import the ASM file. It is responsible for using AsmetaLc Invoker to invoke the AsmetaLc compiler jar file for XMI generation. Once the XMI file is generated it then calls the method to move the XMI file into the selected Event-B project. The class diagram for the package is shown in the figure 5.7. There are three (3) classes in this package. Once is the main import file wizard where as the other two files are the wizard pages.

1. `ImportWizard.java`
2. `ImportWizardPage.java`
3. `XmiEditorPage.java`

5.3 Translation Rules Table

As we have discussed earlier, we used AsmetaLc.jar compiler provided by Asmeta team to generate the XMI file from the selected ASM file. That XMI file is then converted into Java Objects for further rule application for translating the ASM objects to Event-B objects. We had to define a translation table in order to co the translation between these two modeling languages. The table works as a guideline and the translation is done in manner that is part of most of the well known compilers these days. The program looks for the keyword finds out it's type from the table and maps it to the output (Event-B) keyword type. Table 5.1 shows some of the translational mappings. First column is the ASM keyword (struct) type, second column is it's translation and the third column is a brief description of the translation. Note that there can be multiple keywords in one cell.

ASM entity	**Event-B entity**	**Remarks**

Anydomain	CarrierSet, Constant, axiom	Any domain is a user defined domain. It can or can not be finite. In order to accommodate this domain a CarrierSet is defined. If there are any symbols present in the anydomain those symbols will be translated as constants. To make a relationship between the symbols and the set there will be a corresponding axiom generated that will depict their relationship.
Enum	CarrierSet, Constants, Axiom	Enum is a set of multiple symbols and their relationship. In Event-B the symbols are declared as constants, Enum as The CarrierSet and their relationship is declared as Axioms.
AbstractDomain	CarrierSet, Constants, Axiom	Like Anydomain abstract domain is also declared as Carrierset constant and axioms for their relationship.
BasicDomain	Corresponding basic domain (i.e. Integer, boolean, natural number etc.)	Basic domains are directly mapped to corresponding basic structures i.e. Integer, boolean, natural, real etc.
Function	Event(s)	Function performs multiple operations on the variables etc. hence there can be one or more events for a function. Although a par rule within a function will be evaluated to one and only one event. As the assignments in a par rule are autonomous as they are in an Event-B function
Rule	Event	Rule will be mapped to an Event-B event.
invariant	Invariant	ASM invariant will be translated to Event B invariant
Variable	Variable	Directly translated to Event B variable with it's domain
PowersetDomain	Power set definition over domain d	Directly translated to powerset over a domain in Event-B

ConcreteDomain	Subset of basic domain	Subsets of basic domains such as Integers, Natural Numbers etc. It is also the same for sequence domains the only difference is the sequence domain is a sequential subset of basic domains.
UpdateRule	Update statement	Translated to assignment statement of Event-B
ConditionalRule	If - then statement	Translated to conditional statement
ForallTerm	Forall rule of Event-B	Translated to ForAll rule of event b for variable v over the domain d
ExistTerm	Exist term of Event-B	Translated to Exists rule of Event-B.
Case rule	If - Then statement (multiple events - one for each case)	Translated to Multiple events with conditional statements.
Default Init	Initialization Event	Directly translated to Initialization event of Event B
minus (unary)	-	Arthematic operator
plus (unary)	+	Arthematic operator
pwr	Xn̂	Power is multiplication of number power times.
mult	*	Arthematic operator
div	/	Arthematic operator
mod	if x < 0 then x := x * -1	Modulus is unsigned number hence if it is less than 0 just multiply it with -1
eq	=	Translated to equal to
lt	<	Translated to less than
le	<=	Translated to less than or equal to
gt	>	Translated to greater than
ge	>=	Translated to greater than or equal to
neq	/=	Translated to Not equal to
in	belongs to set	Translated to is element of a set
notin	doesn't belong to set	Translated to is not an element of a set
implies	implies sign	Translated to implication
iff	if cond1 then	Translation to if cond1 then stmt1 endif

Table 5.1: Translation Table

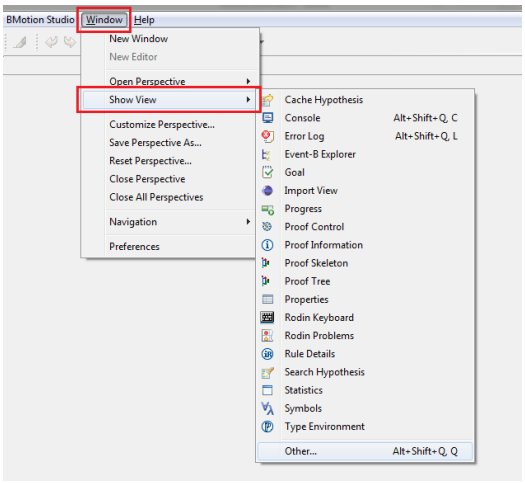

Figure 5.8: Import View Selection Step 1

5.4 Plug-in in action

Now we will present our plug-in in a step by step manner. In order to install the plug-in in the RODIN platform follow the steps

1. Copy the plug-in jar file and the AsmetaLc.jar file and paste it into your RODIN Plugins folder.

2. Restart the RODIN

Once you've restarted RODIN, you can find the plug-in in file → import wizard.

5.4.1 Open Import View

In order to open import view in RODIN IDE follow the steps shown in the figure 5.8. Go to Window→Show View→Other.. or Press **Alt+Shift+Q**. A window like Figure 5.9 will open. Type "Import View" in the filter box. You'll see a view with the name "Import View" under "Import Category". Select the view and Click "OK". Import view will be visible in the IDE.

5.4.2 Import ASM File

We will have a look at how an ASM file can be imported to any selected project. Open your RODIN IDE, create a new Event-B project (if there

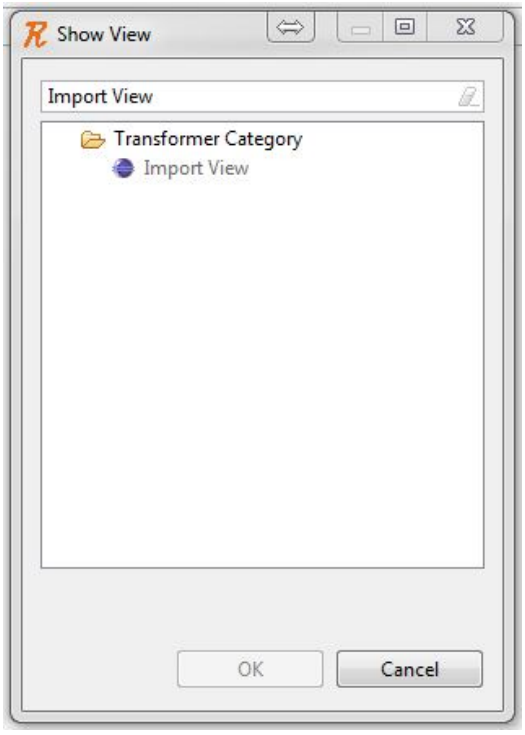

Figure 5.9: Import View Selection Step 2

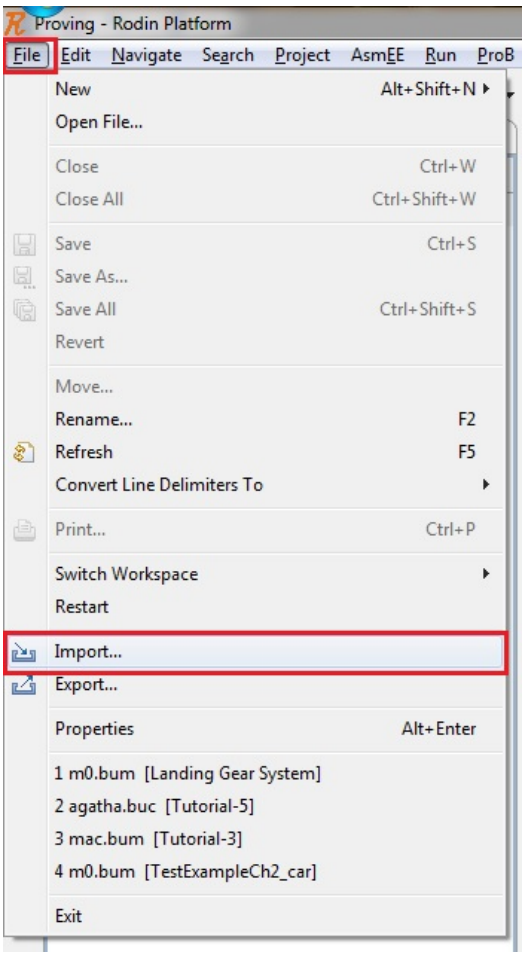

Figure 5.10: Import Wizard Step 1

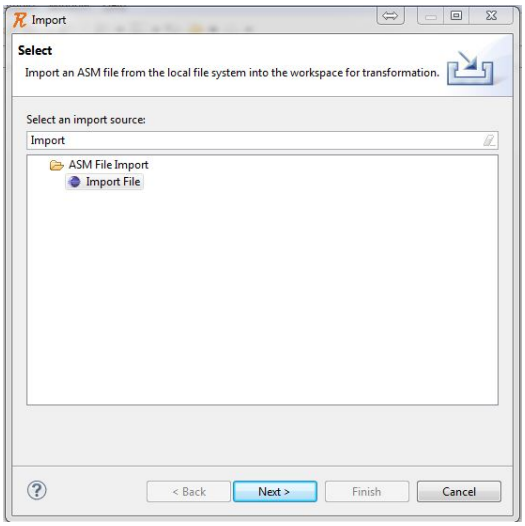

Figure 5.11: Import Wizard Step 2

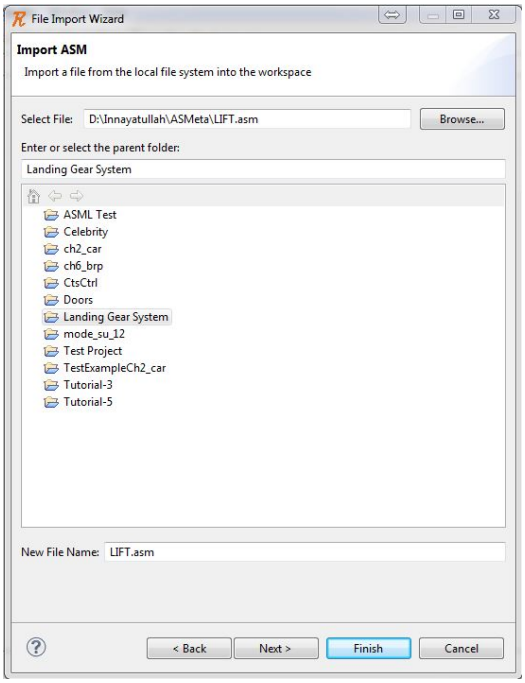

Figure 5.12: Import Wizard Step 3

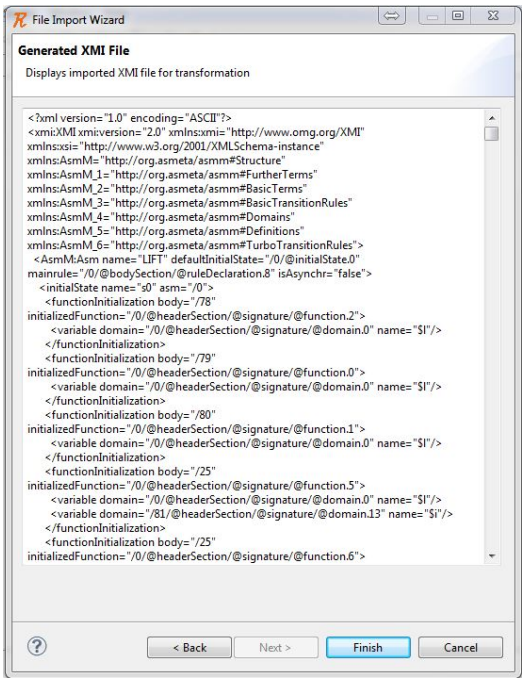

Figure 5.13: Import Wizard Step 4

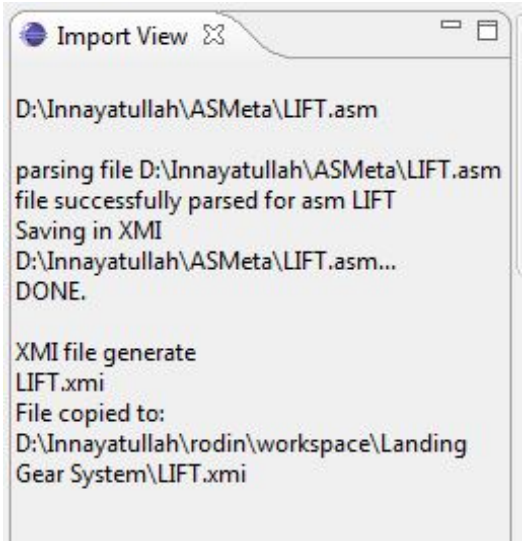

Figure 5.14: Import Wizard Step 5

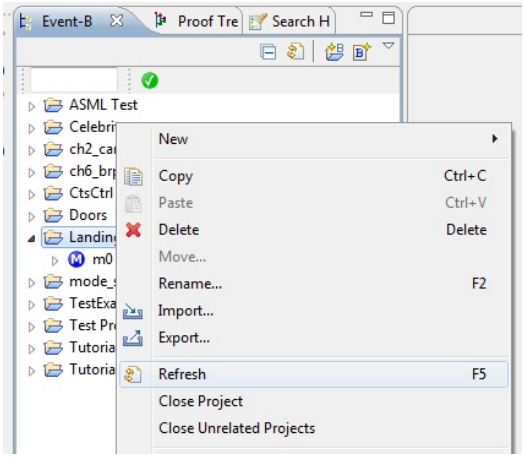

Figure 5.15: Import Wizard Step 6

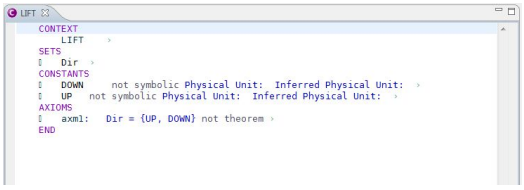

Figure 5.16: Imported context file

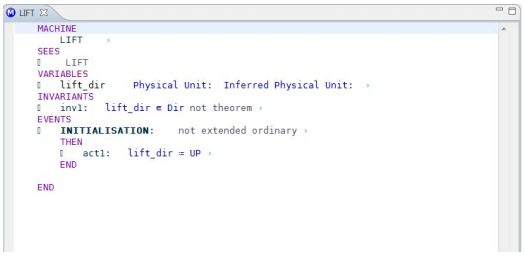

Figure 5.17: Imported machine file

Figure 5.18: Event-B Entities created from ASM file

isn't any). Select your desired Event-B project go to File→Import (Figure 5.10). An import wizard will open. In the filter box type "Import File" you'll see Import File wizard under ASM File Import folder (Figure 5.11). Select Import Wizard and click Next. Another window with the name "File Import Wizard" will open. The window will have your project selected. Click "Browse" button to select your ASM file from the hard drive. Once the file is selected it will be converted and imported at the back-end to the selected project (Figure 5.12). You'll see the output of the process in the import view that we opened earlier. The output in the import view will be (Figure 5.14). At this stage you can Finish the wizard by clicking on "Finish" button in the wizard or you can click Next to move to the next page of the wizard. Here you can see the generated XMI from the ASM file (Figure 5.13). Clicking on Finish button will close the wizard. Right click on your selected project and refresh (or select the project and press F5). The project will be refreshed and newly generated Event-B entities will be present in the project (Figure 5.18).

5.4.3 Final Out Put

Once the files are generated they will contain the constructs and the keywords of Event-B according to the ASM file. Figure 5.16 shows the Context file generated. It will contain static parts i.e. constants, axioms and sets of an Event-B entity. Figure 5.17 shows the Machine file. It contains the dynamic parts i.e. variables, invariants and events etc. of an Event-B entity.

Chapter 6

Conclusion

In this chapter we will look briefly at what we did in this research. What is remaining and the future direction for this translation plug-in.

6.1 Conclusion

Computer systems are being used to design and develop different software and hardware systems. These systems range from simple mathematical calculators to safety critical systems. Formal methods come into play when we are designing safety critical systems. They help us eradicate the design errors. They decrease development and testing costs and produce reliable systems.
Event-B and ASM are widely used formal methods. Where ASM is easy to develop, Event-B tool support is excellent. It's easy to validate and verify the models in Event-B IDE RODIN. Industry and academia require such a tool that can take ASM machines as input and can produce Event-B entities (Context, Machine). We approached this problem with a plug-in that integrates itself in the RODIN platform and semantically translates the ASM files into Event-B. A translation table is used to perform the mapping from one Event-B. As Eclipse is the base of RODIN and it provides rich support for plug-in development, we chose Eclipse for our development.

6.2 Future Direction

In this thesis work we focused on Single Agent Asynchronous ASM. In future the work can be done for multi-agent and Synchronous ASMs. This plug-in works with a limited set of ASM files. The work can be expanded to include all kinds of ASMs.

Appendix A

Java Source code

A.1 edu.seecs.nust.asmtransformer

A.1.1 Activator.java

```
package edu.seecs.nust.asmtransformer;

import org.eclipse.jface.resource.ImageDescriptor;
import org.eclipse.ui.plugin.AbstractUIPlugin;
import org.osgi.framework.BundleContext;

/**
* The activator class controls the plug-in life cycle
*/
public class Activator extends AbstractUIPlugin {

   // The plug-in ID
   public static final String PLUGIN_ID =
      ↪ "edu.seecs.nust.asmtransformer";
      ↪ //$NON-NLS-1$

   // The shared instance
   private static Activator plugin;

   /**
   * The constructor
   */
   public Activator() {
   }
```

```
/*
* (non-Javadoc)
* @see org.eclipse.ui.plugin.AbstractUIPlugin#st
   ↪ art(org.osgi.framework.BundleContext)
*/
public void start(BundleContext context) throws
   ↪ Exception {
   super.start(context);
   plugin = this;
}

/*
* (non-Javadoc)
* @see org.eclipse.ui.plugin.AbstractUIPlugin#st
   ↪ op(org.osgi.framework.BundleContext)
*/
public void stop(BundleContext context) throws
   ↪ Exception {
   plugin = null;
   super.stop(context);
}

/**
* Returns the shared instance
*
* @return the shared instance
*/
public static Activator getDefault() {
   return plugin;
}

/**
* Returns an image descriptor for the image file
   ↪ at the given
* plug-in relative path
*
* @param path the path
* @return the image descriptor
*/
public static ImageDescriptor
   ↪ getImageDescriptor(String path) {
   return imageDescriptorFromPlugin(PLUGIN_ID,
      ↪ path);
```

```
    }
}
```

A.1.2 EventBFile.java

```
package edu.seecs.nust.asmtransformer;

import edu.seecs.nust.asmtransformer.eventb.Context;
import edu.seecs.nust.asmtransformer.eventb.Machinef
    ↪ ileRoot;

public class EventBFiles {

    Context context;
    MachinefileRoot machineFileRoot;

    public Context getcontext() {
        return context;
    }

    public void setcontext(Context ctx) {
        context = ctx;
    }

    public MachinefileRoot getmachineFileRoot() {
        return machineFileRoot;
    }

    public void setmachineFileRoot(MachinefileRoot
        ↪ mfRoot) {
        machineFileRoot = mfRoot;
    }
}
```

A.1.3 Transformer.java

```
package edu.seecs.nust.asmtransformer;

import edu.seecs.nust.asmtransformer.asm.Asm;
import edu.seecs.nust.asmtransformer.eventb.*;
```

```
public class Transformer {

    private static String CONTEXT_CONFIG = "org.e
        ↪ ventb.core.fwd;de.prob.symbolic.ctxBase;de.pr
        ↪ ob.units.mchBase;org.animb.valuation.valBase";
    private static String MACHINE_CONFIG =
        ↪ "org.eventb.core.fwd;de.prob.units.mchBase";

    private static String CONTEXT_VERSION = "1";
    private static String MACHINE_VERSION = "1";

    public static EventBFiles Transform(Asm asm){
        EventBFiles ebFiles = new EventBFiles();
        Context ctx = GetContext(asm);
        MachinefileRoot mfRoot = GetMachineFile(asm);
        ebFiles.setcontext(ctx);
        ebFiles.setmachineFileRoot(mfRoot);
        return ebFiles;
    }

    private static MachinefileRoot GetMachineFile(Asm
        ↪ asm) {
        MachinefileRoot mfRoot = new MachinefileRoot();
        machineFile machineFile = new machineFile();

        machineFile.setconfiguration(MACHINE_CONFIG);
        machineFile.setVersion(MACHINE_VERSION);

        mfRoot.setmachineFile(machineFile);
        return mfRoot;
    }

    private static Context GetContext(Asm asm) {
        Context ctx = new Context();
        contextFile ctxFile = new contextFile();

        ctxFile.setconfiguration(CONTEXT_CONFIG);
        ctxFile.setVersion(CONTEXT_VERSION);

        ctx.setcontextFile(ctxFile);
        return ctx;
    }
```

```
}
```

A.1.4 Utilities.java

```
package edu.seecs.nust.asmtransformer;

import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.net.URL;

import org.eclipse.core.resources.IProject;
import org.eclipse.core.resources.IResource;
import org.eclipse.core.runtime.Platform;
import org.eclipse.jface.viewers.ISelection;
import
   ↪ org.eclipse.jface.viewers.IStructuredSelection;
import org.eclipse.ui.ISelectionService;
import org.eclipse.ui.internal.Workbench;

@SuppressWarnings("restriction")
public class Utilities {

/*
* Returns the current selected project
*/
public static IProject getCurrentProject(){
   ISelectionService selectionService =
   Workbench.getInstance().getActiveWorkbenchWindow().
      ↪ getSelectionService();

   ISelection selection =
      ↪ selectionService.getSelection();

   IProject project = null;
   if(selection instanceof IStructuredSelection) {
      Object element = ((IStructuredSelection)
         ↪ selection).getFirstElement();

      if (element instanceof IResource) {
         project= ((IResource)element).getProject();
```

```
            }
        }
        return project;
    }

    public static byte[] getBytes(InputStream is)
        ↪ throws IOException {

        int len;
        int size = 1024;
        byte[] buf;

        if (is instanceof ByteArrayInputStream) {
            size = is.available();
            buf = new byte[size];
            len = is.read(buf, 0, size);
        } else {
            ByteArrayOutputStream bos = new
                ↪ ByteArrayOutputStream();
            buf = new byte[size];
            while ((len = is.read(buf, 0, size)) != -1)
            bos.write(buf, 0, len);
            buf = bos.toByteArray();
        }
        return buf;
    }

    /*
    * Processes the passed as file in command prompt
        ↪ and returns the process
    */
    public static Process ProcessInCommandLine(String
        ↪ command){
        Runtime rt = Runtime.getRuntime();
        try {
            Process pr = rt.exec(command);
            return pr;
        } catch (IOException e) {
            e.printStackTrace();
            return null;
        }
    }
```

```
    public static String GetPluginLocation(){
        return Platform.getInstallLocation().getURL()
            ↪ .getPath();
    }

    public static String ChangeExtension(String
        ↪ source, String extension){
        String[] splittedString = source.split("\\.");
        splittedString[splittedString.length-1] =
            ↪ extension;
        return JoinStrings(splittedString, ".");
    }

    public static String JoinStrings(String[] source,
        ↪ String glueingChar){
        String dest = "";
        for(int i=1; i< source.length; i++){
            dest += source[i-1] + glueingChar +
                ↪ source[i];
        }
        return dest;
    }
}
```

A.1.5 XmlGenerator.java

```
package edu.seecs.nust.asmtransformer;
import java.io.File;
import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBException;
import javax.xml.bind.Marshaller;

public class XmlGenerator {
    public void GenerateXML(Object rootMachineFile,
        ↪ String destPath){
        try {
            File file = new File(destPath);
            JAXBContext jaxbContext =
                ↪ JAXBContext.newInstance(
                ↪ rootMachineFile.getClass());
            Marshaller jaxbMarshaller =
                ↪ jaxbContext.createMarshaller();
```

```
            // output pretty printed
            jaxbMarshaller.setProperty(Marshaller.
               ↪ JAXB_FORMATTED_OUTPUT, true);
            jaxbMarshaller.marshal(rootMachineFile,
               ↪ file);
            jaxbMarshaller.marshal(rootMachineFile,
               ↪ System.out);
         } catch (JAXBException e) {
            e.printStackTrace();
         }
      }
   }
```

A.2 edu.seecs.nust.asmtransformer.views

A.2.1 ImportView.java

```
   package edu.seecs.nust.asmtransformer.views;
   import org.eclipse.swt.widgets.Composite;
   import org.eclipse.swt.widgets.Label;
   import org.eclipse.ui.part.*;
   import org.eclipse.jface.viewers.*;
   import org.eclipse.swt.graphics.Image;
   import org.eclipse.jface.action.*;
   import org.eclipse.jface.dialogs.MessageDialog;
   import org.eclipse.ui.*;
   import org.eclipse.swt.widgets.Menu;
   import org.eclipse.swt.SWT;

   public class ImportView extends ViewPart {
      /**
      * The ID of the view as specified by the
         ↪ extension.
      */
      public static final String ID = "edu.seecs.nust.
         ↪ asmtransformer.views.ImportView";
      private static Label label;

      /**
      * The constructor.
      */
```

```
    public ImportView() {
    }

    /**
    * This is a callback that will allow us
    * to create the viewer and initialize it.
    */
    public void createPartControl(Composite parent) {
       label = new Label(parent, SWT.WRAP);
    }

    public static void SetText(String text,
       ↪ Boolean... reset){
          if( reset.length > 0 && reset[0] == true)
             label.setText("");
          label.setText(label.getText() +"\r\n" +
             ↪ text);
    }
}
```

A.3 edu.seecs.nust.asmtransformer.importWizards

A.3.1 ImportWizard.java

```
package edu.seecs.nust.asmtransformer.importWizards;
import java.io.IOException;
import java.nio.file.Path;
import
   ↪ org.eclipse.jface.viewers.IStructuredSelection;
import org.eclipse.jface.wizard.IWizardPage;
import org.eclipse.jface.wizard.Wizard;
import org.eclipse.ui.IImportWizard;
import org.eclipse.ui.IWorkbench;
import edu.seecs.nust.asmtransformer.asm.Asm;
import
   ↪ edu.seecs.nust.asmtransformer.asm.StaxAsmParser;

public class ImportWizard extends Wizard implements
   ↪ IImportWizard {
   ImportWizardPage importPage;
   XmiEditorPage editorPage;
```

```
Asm asmObject = null;
Path filePath;

public ImportWizard() {
    super();
}

/* (non-Javadoc)
* @see org.eclipse.jface.wizard.Wizard#
    ↪ performFinish()
*/
public boolean performFinish() {
    Boolean copied = false;
    Boolean converted = false;
    return true;
}

/* (non-Javadoc)
* @see org.eclipse.ui.IWorkbenchWizard#
    ↪ init(org.eclipse.ui.IWorkbench, org.eclipse
    ↪ .jface.viewers.IStructuredSelection)
*/
public void init(IWorkbench workbench,
    ↪ IStructuredSelection selection) {
    setWindowTitle("File Import Wizard");
        ↪ //NON-NLS-1
    setNeedsProgressMonitor(true);
    importPage = new ImportWizardPage("Import
        ↪ ASM", selection); //NON-NLS-1
    editorPage = new XmiEditorPage("Generated XMI
        ↪ File"); //NON-NLS-1
}

/* (non-Javadoc)
* @see org.eclipse.jface.wizard.IWizard#addPages()
*/
public void addPages() {
    super.addPages();
    addPage(importPage);
    addPage(editorPage);
}

public IWizardPage getNextPage(IWizardPage page){
```

```
        if(page.getName() == "Import ASM")
        {
            try {
                importPage.CopyFile();
                filePath = importPage.desPath;
                editorPage.xmiFilePath = filePath;
                StaxAsmParser asmParser = new
                    ↪ StaxAsmParser();
                String fileContent = asmParser.Get
                    ↪ FileContents(filePath.toString());
                editorPage.SetText(fileContent);
                if(asmObject == null){
                    asmObject = asmParser.readFile
                        ↪ (filePath.toString());
                }
                editorPage.GenerateDummyEventBFiles();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        return super.getNextPage(page);
    }
}
```

A.3.2 ImportWizardPage.java

```
package edu.seecs.nust.asmtransformer.importWizards;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.nio.file.StandardCopyOption;
import org.eclipse.core.resources.IProject;
import org.eclipse.core.runtime.IPath;
import org.eclipse.core.runtime.IStatus;
import org.eclipse.core.runtime.Path;
import org.eclipse.core.runtime.Status;
import org.eclipse.jface.preference.FileFieldEditor;
```

```
import
   ↪ org.eclipse.jface.viewers.IStructuredSelection;
import org.eclipse.swt.SWT;
import org.eclipse.swt.events.ModifyEvent;
import org.eclipse.swt.events.ModifyListener;
import org.eclipse.swt.layout.GridData;
import org.eclipse.swt.layout.GridLayout;
import org.eclipse.swt.widgets.Composite;
import org.eclipse.swt.widgets.Text;
import
   ↪ org.eclipse.ui.dialogs.WizardNewFileCreationPage;
import edu.seecs.nust.asmtransformer.Utilities;
import
   ↪ edu.seecs.nust.asmtransformer.asm.StaxAsmParser;
import
   ↪ edu.seecs.nust.asmtransformer.views.ImportView;

public class ImportWizardPage extends
   ↪ WizardNewFileCreationPage {

   protected FileFieldEditor editor;
   protected Composite fileSelectionArea;
   public java.nio.file.Path desPath;
   public String sourceFile;

   public ImportWizardPage(String pageName,
      ↪ IStructuredSelection selection) {
      super(pageName, selection);
      setTitle(pageName); //NON-NLS-1
      setDescription("Import a file from the local
         ↪ file system into the workspace");
         ↪ //NON-NLS-1
   }

   /* (non-Javadoc)
   * @see org.eclipse.ui.dialogs.WizardNewFile
      ↪ CreationPage#createAdvancedControls(org
      ↪ .eclipse.swt.widgets.Composite)
   */
   protected void createAdvancedControls(Composite
      ↪ parent) {
      fileSelectionArea = new Composite(parent,
         ↪ SWT.NONE);
```

```
        GridData fileSelectionData = new
           ↪ GridData(GridData.GRAB_HORIZONTAL
        | GridData.FILL_HORIZONTAL);
        fileSelectionArea.setLayoutData(
           ↪ fileSelectionData);

        GridLayout fileSelectionLayout = new
           ↪ GridLayout();
        fileSelectionLayout.numColumns = 3;
        fileSelectionLayout.makeColumnsEqualWidth =
           ↪ false;
        fileSelectionLayout.marginWidth = 0;
        fileSelectionLayout.marginHeight = 0;
        fileSelectionArea.setLayout(
           ↪ fileSelectionLayout);

        editor = new
           ↪ FileFieldEditor("fileSelect","Select
           ↪ File: ",fileSelectionArea); //NON-NLS-1
           ↪ //NON-NLS-2
        editor.getTextControl(fileSelectionArea)
           ↪ .addModifyListener(new ModifyListener(){
           public void modifyText(ModifyEvent e) {
           IPath path = new
              ↪ Path(ImportWizardPage.this.editor
              ↪ .getStringValue());
           setFileName(path.lastSegment());
           if(!path.lastSegment().isEmpty()){
           setPageComplete(true);
           }
           }
        });
        String[] extensions = new String[] { "*.asm"
           ↪ }; //NON-NLS-1
        editor.setFileExtensions(extensions);
        fileSelectionArea.moveAbove(null);
    }

    /* (non-Javadoc)
    * @see org.eclipse.ui.dialogs.WizardNewFile
       ↪ CreationPage#createLinkTarget()
    */
    protected void createLinkTarget() {
```

```
    }

    /* (non-Javadoc)
    * @see org.eclipse.ui.dialogs.WizardNewFile
       ↪ CreationPage#getInitialContents()
    */
    protected InputStream getInitialContents() {
       try {
          return new FileInputStream(new
             ↪ File(editor.getStringValue()));
       } catch (FileNotFoundException e) {
          return null;
       }
    }

    /* (non-Javadoc)
    * @see org.eclipse.ui.dialogs.WizardNewFile
       ↪ CreationPage#getNewFileLabel()
    */
    protected String getNewFileLabel() {
       return "New File Name:"; //NON-NLS-1
    }

    /* (non-Javadoc)
    * @see org.eclipse.ui.dialogs.WizardNewFile
       ↪ CreationPage#validateLinkedResource()
    */
    protected IStatus validateLinkedResource() {
       return new Status(IStatus.OK,
          ↪ "com.seecs.nust.asmtransformer",
          ↪ IStatus.OK, "", null); //NON-NLS-1
          ↪ //NON-NLS-2
    }

    public Boolean CopyFile() throws IOException{
       Text fileControl =
          ↪ editor.getTextControl(fileSelectionArea);
       sourceFile = fileControl.getText();
       ImportView.SetText(sourceFile,true);
       ConvertToXMI();
       sourceFile =
          ↪ Utilities.ChangeExtension(sourceFile,
          ↪ "xmi");
```

```
        ImportView.SetText("XMI file generate");
        IProject project =
            ↪ Utilities.getCurrentProject();
        if(project != null){
        java.nio.file.Path sourcePath =
            ↪ Paths.get(sourceFile);
        String fileName = getFileName();
        desPath = Paths.get(project.getLocationURI());
        fileName = Utilities.ChangeExtension(fileName,
            ↪ "xmi");
        ImportView.SetText(fileName);
        desPath = desPath.resolve(fileName);
        Files.copy(sourcePath, desPath,
            ↪ StandardCopyOption.REPLACE_EXISTING);
        ImportView.SetText("File copied to: " +
            ↪ desPath.toString());
    }

    return true;
    }

    public Boolean ConvertToXMI(){
        try{
            String command = "java -jar \"";
            command += Utilities.GetPluginLocation()
                ↪ .substring(1);
            command += "plugins/ASMetaLc.jar\"";
            command += " -xmi \"";
            command += sourceFile + "\"";
            Process process =
                ↪ Utilities.ProcessInCommandLine(command);
            ImportView.SetText(new
                ↪ String(Utilities.getBytes(
                ↪ process.getErrorStream()), "UTF-8"));
            ImportView.SetText(new
                ↪ String(Utilities.getBytes(
                ↪ process.getInputStream()), "UTF-8"));
            return true;
        }
        catch(Exception ex){
            System.console().printf(ex.getMessage());
            return false;
        }
```

```
    }
}
```

A.3.3 XmiEditorPage.java

```
package edu.seecs.nust.asmtransformer.importWizards;
import java.nio.file.Path;
import java.util.ArrayList;
import org.eclipse.jface.wizard.WizardPage;
import org.eclipse.swt.SWT;
import org.eclipse.swt.layout.GridData;
import org.eclipse.swt.layout.GridLayout;
import org.eclipse.swt.widgets.Composite;
import org.eclipse.swt.widgets.Text;
import edu.seecs.nust.asmtransformer.Transformer;
import edu.seecs.nust.asmtransformer.Utilities;
import edu.seecs.nust.asmtransformer.XmlGenerator;
import edu.seecs.nust.asmtransformer.asm.Asm;
import edu.seecs.nust.asmtransformer.eventb.*;

public class XmiEditorPage extends WizardPage {
    private Text textArea;
    public Path xmiFilePath;
    protected XmiEditorPage(String pageName) {
        super(pageName);
        setTitle(pageName); //NON-NLS-1
        setDescription("Displays imported XMI file for
            ↪ transformation");
    }

    @Override
    public void createControl(Composite parent) {
        initializeDialogUnits(parent);
        Composite composite= new Composite(parent,
            ↪ SWT.NONE);
        int nColumns= 1;
        GridLayout layout= new GridLayout();
        layout.numColumns= nColumns;
        composite.setLayout(layout);

        GridData gd= new GridData();
        gd.horizontalSpan= nColumns;
```

```
        gd.widthHint = 480;
        gd.heightHint = 480;

        textArea= new Text(composite, SWT.READ_ONLY |
           ↪ SWT.BORDER | SWT.V_SCROLL | SWT.WRAP);
        textArea.setLayoutData(gd);
        textArea.setText("");
        textArea.setBackground(parent.getShell()
           ↪ .getDisplay().getSystemColor(SWT.COLOR_LIS
           ↪ T_BACKGROUND));
        setControl(composite);
    }

    public void SetText(String text){
        textArea.setText(text);
    }

    public void GenerateEventBFiles(Asm asm){
        Transformer.Transform(asm);
    }
}
```

A.4 edu.seecs.nust.asmtransformer.asm

A.4.1 Asm.java

```
package edu.seecs.nust.asmtransformer.asm;

public class Asm {
    String name;
    String defaultInitialState;
    String mainrule;
    Boolean isAsynchr;

    initialState initialState;
    bodySection bodySection;
    headerSection headerSection;

    public String getName(){
    return name;
    }
```

```
public void setName(String n){
name = n;
}

public String getDefaultInitialState(){
return defaultInitialState;
}

public void setDefaultInitialState(String dis){
defaultInitialState = dis;
}

public String getMainrule(){
return mainrule;
}

public void setMainRule(String mr){
mainrule = mr;
}

public Boolean getIsAsynchr(){
return isAsynchr;
}

public void setIsAsynchr(Boolean ia){
isAsynchr = ia;
}

public initialState getInitialState(){
return initialState;
}

public void setInitialState(initialState is){
initialState = is;
}

public bodySection getBodySection() {
return bodySection;
}

public void setBodySection(bodySection bs) {
bodySection = bs;
```

```
    }

    public headerSection getHeaderSection() {
    return headerSection;
    }

    public void setHeaderSection(headerSection hs) {
    headerSection = hs;
    }
}
```

A.5 edu.seecs.nust.asmtransformer.eventb

A.5.1 contextFile.java

```
package edu.seecs.nust.asmtransformer.eventb;

import java.util.ArrayList;
import javax.xml.bind.annotation.*;

@XmlRootElement(name = "org.eventb.core.contextFile")
public class contextFile{
   private ArrayList<axiom> axiom;
   private String configuration;
   private extendsContext extendsContext;
   private ArrayList<constant> constant;
   private ArrayList<carrierSet> carrierSet;
   private String version;

   public ArrayList<axiom> getaxiom (){
      return axiom;
   }

   @XmlElement(name = "org.eventb.core.axiom")
   public void setaxiom (ArrayList<axiom> axiom){
      this.axiom = axiom;
   }

   public String getconfiguration (){
      return configuration;
   }
```

```
@XmlAttribute(name =
    ↪ "org.eventb.core.configuration")
public void setconfiguration (String
    ↪ configuration){
    this.configuration = configuration;
}

public extendsContext getextendsContext (){
    return extendsContext;
}

@XmlElement(name =
    ↪ "org.eventb.core.extendsContext")
public void setextendsContext (extendsContext
    ↪ extendsContext){
    this.extendsContext = extendsContext;
}

public ArrayList<constant> getconstant ()
{
return constant;
}

@XmlElement(name = "org.eventb.core.constant")
public void setconstant (ArrayList<constant>
    ↪ constant){
    this.constant = constant;
}

public ArrayList<carrierSet> getcarrierSet (){
    return carrierSet;
}

@XmlElement(name = "org.eventb.core.carrierSet")
public void setcarrierSet (ArrayList<carrierSet>
    ↪ carrierSet){
    this.carrierSet = carrierSet;
}

public String getVersion (){
    return version;
}
```

```
    @XmlAttribute
    public void setVersion (String version){
        this.version = version;
    }

    @Override
    public String toString(){
        return "contextFile [axiom = "+axiom+",
            ↪ configuration = "+configuration+",
            ↪ extendsContext = "+extendsContext+",
            ↪ constant = "+constant+", carrierSet =
            ↪ "+carrierSet+", version = "+version+"]";
    }
}
```

A.5.2 machineFile.java

```
package edu.seecs.nust.asmtransformer.eventb;

import java.util.ArrayList;

import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement(name = "org.eventb.core.machineFile")
public class machineFile{
    private String configuration;
    private refinesMachine refinesMachine;
    private seesContext seesContext;
    private ArrayList<invariant> invariant;
    private String version;
    private ArrayList<variable> variable;
    private ArrayList<event> event;

    public String getconfiguration (){
        return configuration;
    }

    @XmlAttribute(name =
        ↪ "org.eventb.core.configuration")
```

```
public void setconfiguration (String
   ↪ configuration){
   this.configuration = configuration;
}

public refinesMachine getrefinesMachine (){
   return refinesMachine;
}

@XmlElement(name =
   ↪ "org.eventb.core.refinesMachine")
public void setrefinesMachine (refinesMachine
   ↪ refinesMachine){
   this.refinesMachine = refinesMachine;
}

public seesContext getseesContext (){
   return seesContext;
}

@XmlElement(name="org.eventb.core.seesContext")
public void setseesContext (seesContext
   ↪ seesContext){
   this.seesContext = seesContext;
}

public ArrayList<invariant> getinvariant (){
   return invariant;
}

@XmlElement(name = "org.eventb.core.invariant")
public void setinvariant (ArrayList<invariant>
   ↪ invariant){
   this.invariant = invariant;
}

public String getVersion (){
   return version;
}

@XmlAttribute
public void setVersion (String version){
   this.version = version;
```

```
    }

    public ArrayList<variable> getvariable (){
        return variable;
    }

    @XmlElement(name = "org.eventb.core.variable")
    public void setvariable (ArrayList<variable>
        ↪ variable){
        this.variable = variable;
    }

    public ArrayList<event> getevent (){
        return event;
    }

    @XmlElement(name = "org.eventb.core.event")
    public void setevent (ArrayList<event> event){
        this.event = event;
    }

    @Override
    public String toString(){
        return "machineFile [configuration =
            ↪ "+configuration+", refinesMachine =
            ↪ "+refinesMachine+", seesContext =
            ↪ "+seesContext+", invariant =
            ↪ "+invariant+", version = "+version+",
            ↪ variable = "+variable+", event =
            ↪ "+event+"]";
    }
}
```

Bibliography

Formal methods wiki : http://formalmethods.wikia.com/wiki/formal_methods_wiki.

Boniol, F. and Wiels, V. *Landing gear system.*

C. Mtayer J.-R. Abrial, L. V. *Event-B Language.*

Jean-Raymond Abrial, Michael Butler, S. H. T. S. H. F. M. L. V. *RODIN: An Open Toolset for Modelling and Reasoning in Event-B.*

Microsoft-Research. *Introducing AsmL: A Tutorial for the Abstract State Machine Language.*

MULLER, T. Formal methods, model-cheking and rodin plugin development to link event-b and spin.

S. Rajasekar, P. Philominathan, V. C. *RESEARCH METHODOLOGY.*