Code Generation for Event-B

Innopolis University

Interim report

Georgiy Krikun

5 December 2016

Abstract

In this interim report I will describe specifics of my bachelor thesis work (which worth two courses).

Contents

Goals of the project

The main objective of the project is implementation a translator from Event-B modeling language to Eiffel programming language as Rodin platform plugin.

Like it already be done by Nestor Catano and Victor Rivera for translation from Event-B to Java (or JML), before me.

But in my case I should implement plugin which translate Event-B model to Eiffel Programming Language.

As results of work should be delivered working plugin, case study (or studies) model in Event-B, translation of this model to Eiffel and proves of soundness of the translator.

Overview of the report:

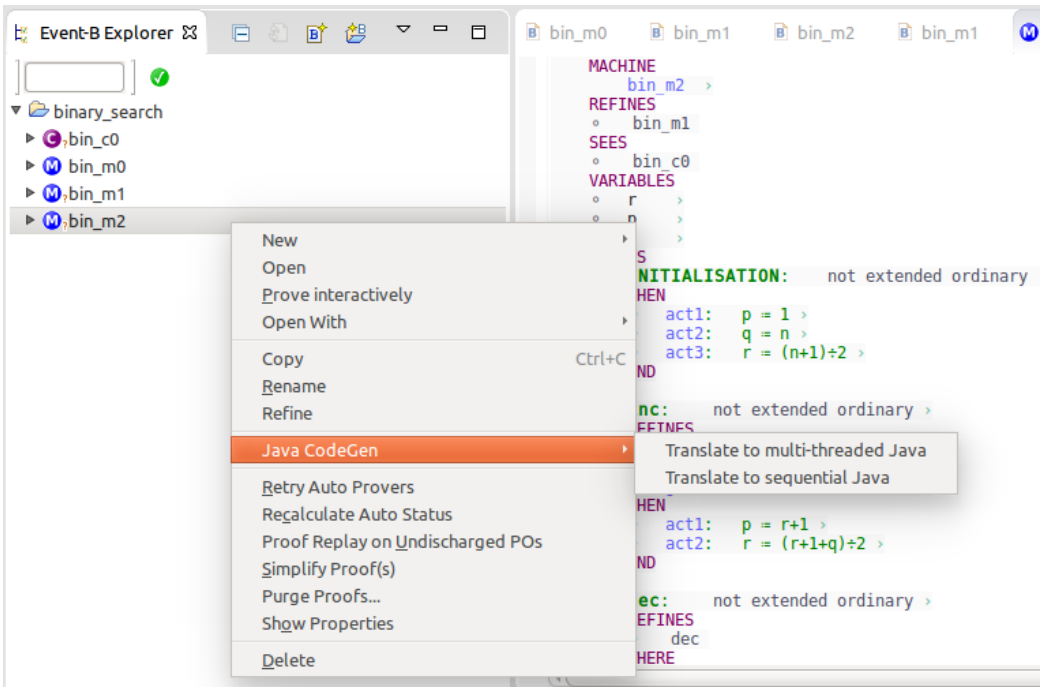
1. **Overview of System Specification** - section describes development specifics for Rodin Platform.
2. **Background Theory** - section describes formal approach, Event-B methodology, Eiffel Programming Language.
3. **Overview of Task Specification and Project Schedule** - section consider plugin development decomposition to tasks, schedule timings and so on.
4. **Review of Tasks** - section describes current status of tasks.
5. **Interim Results** - section describes interim results of thesis work.
6. **Short Term Plans** - section states next steps I will take in project.

Overview of the System Specification

Whole system is a plugin for Rodin Platform. It uses translation rules (like this one) from Event-B to Eiffel and produce Eiffel code. Translation rules has been provided to me by Victor Rivera.

Tutorial for the extension of the Rodin platform by plugin addition.

Access to plugin functionality will be granted through context menu of specific machine, like in EventB2Java plugin:



But in my case it would be `Eiffel CodeGen` and will translate to Eiffel code.

Background Theory

Formal methods

Formal methods are techniques used to model complex systems as mathematical entities.

The primary idea behind a formal method is that there is benefit in writing a precise specification of a system, and formal methods use a formal or mathematical syntax to do so. This syntax is usually textual but can be graphical.

Roughly speaking, formal design can be seen as a three step process:

1. **Formal Specification** During the formal specification phase, the engineer rigorously defines a system using a modeling language. Modeling languages are fixed grammars which allow users to model complex structures out of predefined types. This process of formal specification is similar to the process of converting a word problem into algebraic notation.

2. **Verification** As stated above, formal methods differ from other specification systems by their heavy emphasis on provability and correctness. By building a system using a formal specification, the designer is actually developing a set of theorems about his system.

3. **Implementation** Once the model has been specified and verified, it is implemented by converting the specification into code. As the difference between software and hardware design grows narrower, formal methods for developing embedded systems have been developed.

Event-B

Event-B is another formal method for modelling complete developments of discrete transition systems. Event-B was introduced by J-R. Abrial, and is derived from the B method.

Unlike in B models, the static part of Event-B models is separated from the dynamic part, and is referred to as "contexts". Thus, Event-B models are composed of machines (the dynamic part. e.g. variables, invariants, events),

and contexts (the static part. e.g. carrier sets, constants).

Three basic relationships between machines and contexts are used to structure a model:

- A machine **sees** a context
- A machine can **refine** another machine
- A context can **extend** another context

<machine_identifier >
refines
< machine_identifier >
sees
< context_identifier_list >
variables
< variable_identifier_list >
invariants
< label >: < predicate >
variants
< variant >
events
< event_list >

Figure 1: General structure of Event-B machine (taken from [1])

<context_identifier >
extends
< context_identifier_list >
sets
< set_identifier_list >
constants
< constant_identifier_list >
axioms
< label >: < predicate >

Figure 2: General structure of Event-B context

Rodin Platform

The Rodin Platform is an Eclipse-based IDE for Event-B that provides effective support for refinement and mathematical proof. The platform is open source, contributes to the Eclipse framework and is further extendable with plugins.

Overview of Task Specification and Project Schedule

List the primary tasks and sub-tasks required to carry out the project and an overview of the project schedule, giving the timings (start date, duration, amount of effort) of each task. List and discuss any changes that have been made to the initial specification of the project.

Review of Tasks

Provide a comprehensive review of the status of each task and sub-task, setting out at least: The status (not started, on-going, complete, behind schedule, ahead of schedule ...); Problems encountered and identified solutions; Anticipated problems and possible solutions; Impact on the project schedule.

Interim Results

If possible, provide examples of any interim results you have achieved.

Short Term Plans

Identify the next steps you will take in the project.

References

[1] Jean-Raymond Abrial. *Modeling in Event-B: System and Software Engineering*. Cambridge University Press, New York, 2010.