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). Actually I don't stop work on thesis, you could see latest version in my repository, and the latest version of this report as well.

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:

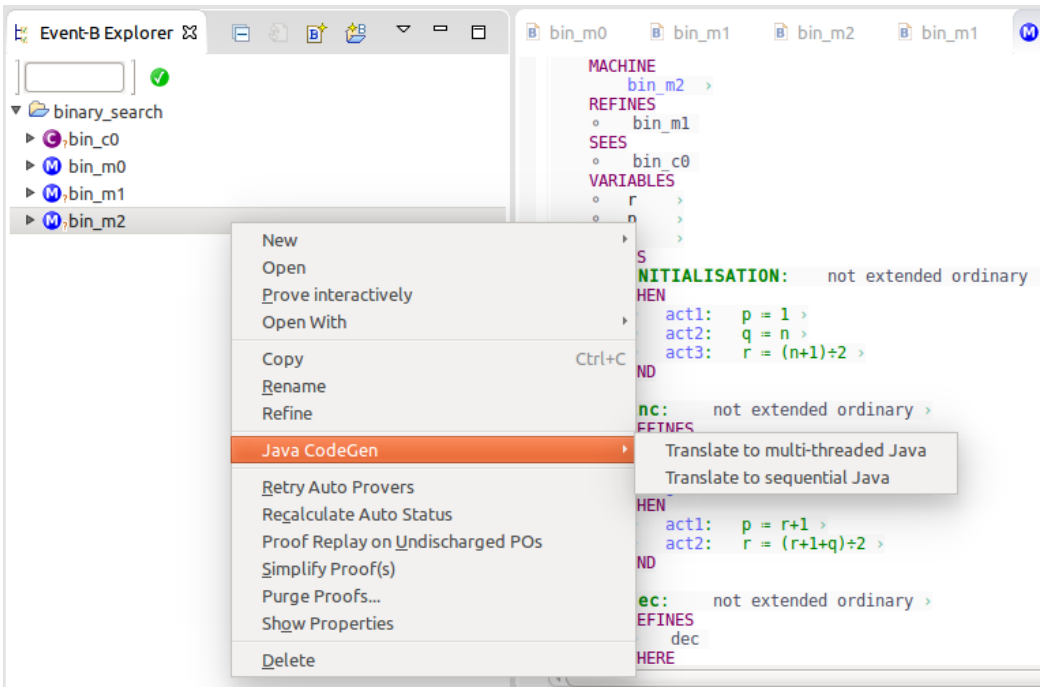
1. **Overview of System Specification** - section describes development specifics for Rodin Platform.
2. **Background Theory** - section describes formal approach, Event-B methodology, Eiffel Programming Language.
3. **Overview of Task Specification and Project Schedule** - section consider plugin development 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

General structure of models and method overall, well described by J-R. Abrial in [1]

Rodin Platform

Rigorous Open Development Environment for Complex Systems

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.

Rodin project

In Overview of System Specification, we have seen screenshot of the IDE.

Eiffel Programming Language

Eiffel is a statically typed, Object Oriented language designed by Bertrand Meyer. Eiffel features an AdaLanguage-like syntax, a robust type system, and direct language support for DesignByContract. Although overlooked by the mainstream programming community, those who actually use it are very excited about it.

It easily to find documentation about it.

Also there is EiffelStudio, IDE for this language. But I don't like to use it and all what we need - build and compile eiffel classes for testing, in the end (what job could be done through command line).

Task Specification, Project Schedule

Environment Installation	Sept. 2016	2 days	Easy
Play with examples	Sept. 2016	3 days	Easy
Read overview about formal methods	Sept. 2016	1 day	Easy
Read introduction to Event-B	Sept. 2016	1 day	Easy
Try contact with Rodin community	2016 – 2017	1 day/month	Hard
actually irc channel dead, mail list silent			
Read Event-B book [1]	Oct. 2016	1 month	Easy
Understand Event-B book [1]	Nov. 2016	2 month	Medium
Model case-study in Event-B	Dec. 2016	1 month	Medium
Implement rules of translation	Jan. 2017	$\sim$ 2 month	Hard
Implement Translator "Visitor"	Mar. 2017	$\sim$ 1 month	Hard
Prove soundness of the translator	Apr. 2017	$\sim$ 1 month	Hard
Translate case-study into Eiffel	May. 2017	$\sim$ 1 week	Medium

Review of Tasks

By now I slightly behind the schedule. Right now I think about case-study model in Event-B and couldn't say that fully get ideas of the book.

As case-study I suggested modeling translator itself. It is interesting idea but could be difficult and useless. As a plan – use existing translator to Java (EventB2Java) to get Java code and tweak it to Rodin plugin. But Victor Rivera think it could be difficult for me, and actually doing work twice. So right now I should decide whether I will model translator itself (this will prove it soundness) or I will implement translation rules in Java as plugin to Rodin, and then prove soundness of the translator.

So schedule depends on my decision and if I choose first way – I do not have to prove soundness (save about a month I think), but in any case I have to implement "the Visitor".

All other task going right on schedule.

Interim Results

- Already play with existing case-studies in Rodin.
- Install Rodin plugins, generate Java code from Event-B models.
- Read the Event-B book [1].
- Read Victor Rivera thesis (almost same theme).
- Prepare materials I will need in work (Rodin handbook, Event-B documentations, Eiffel documentations also collect Nestor Catano and Victor Rivera papers)

Short Term Plans

As next step – I will implement "Hello World" plugin (just for get some experience in plugin development). And finally decide whether I will do model of translator in Event-B or not (by trying modeling, if I will fail it could be the fate).

References

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