Generating a Real-Time Algorithmic Trading System Prototype from Customized UML Models (a case study)

Chong Li, Gaétan Hains, Youry Khmelevsky, Brandon Potter, Jesse Gaston, Andrew Jankovic, Sam Boateng, William Lee

Generating a Real-Time Algorithmic Trading System Prototype from Customized UML Models (a case study)

Gaétan HAINS **Chong LI**
Youry KHMELEVSKY **Brandon POTTER**
Jesse GASTON **Andrew JANKOVIC**
Sam BOATENG **William LEE**

September 2012

TR–LACL–2012–09

Laboratoire d'Algorithmique, Complexité et Logique (LACL)
Université de Paris-Est Créteil (UPEC), Faculté des Science et Technologie
61, Avenue du Général de Gaulle, 94010 Créteil cedex, France
Tel.: (+33)(1) 45 17 16 47, Fax: (+33)(1) 45 17 66 01

Laboratory for Algorithmics, Complexity and Logic (LACL)
University of Paris-Est Créteil (UPEC)

Technical Report **TR–LACL–2012–09**

G. HAINS, C. LI, Y. KHMELEVSKY, et al.

Generating a Real-Time Algorithmic Trading System Prototype from Customized UML Models (a case study)

Generating a Real-Time Algorithmic Trading System Prototype from Customized UML Models (a case study)

Gaétan Hains[1,2] Chong Li[1,2] Youry Khmelevsky[3] Brandon Potter[3]
Jesse Gaston[3] Andrew Jankovic[3] Sam Boateng[3] William Lee[3]

[1] LACL, Université Paris-Est
61, Avenue du Général de Gaulle, 94010 Créteil, France

[2] EXQIM S.A.S.
24, Rue de Caumartin, 75009 Paris, France

[3] University of British Columbia
333 University Way, Kelowna, BC, Canada

Gaetan.Hains@u-pec.fr
Chong.Li@exqim.com
Youry.Khmelevsky@ubc.ca

Abstract

Real-time algorithmic trading systems are widely used by pension funds, mutual funds, some hedge funds, market makers and other institutional traders, to manage market impact and risk, to provide liquidity to the market. The technologies of real-time information processing and high-performance computing, such as the parallel bridging model - SGL [1], are essential for such systems. However, many errors can be made with todays tools, for example, the distraction of developers because they must focus both on financial algorithms, parallel computing and coding, or compiler mis-optimization, etc. In this paper, we describe practical results with the software design of a real-time algorithmic trading prototype by undergraduate students within the CoSc 319 software engineering project course at the University of British Columbia's Okanagan campus (Canada) in collaboration with a post graduate student from the University Paris-Est (France). The prototype can be modified by end-users on the UML model level and then used with automatic Java code generation and execution within the Eclipse IDE. During the case study an advanced coding environment was developed for providing a visual and declarative approach to trading algorithms development so as to generate directly portable bitcode on Low-Level Virtual Machine (LLVM)[1] from financial specification of trading strategies. During the project, Canadian students collaborated with a research engineer from a hedge fund in Paris.

Keywords: real-time systems; software engineering; documentation; scalability; source code generation; UML; EMF; Eclipse.

[1] http://llvm.org/

Contents

1 Introduction

Computer Science research has enormous potential for applications. Yet much of the current research is of an incremental stovepipe" or inward-looking nature, even though Computer Science has a unique ability to fundamentally transform virtually every other discipline [2]. This paper's objective is to discuss our experience and new challenges in Computer Science and Software Engineering (SE) Project courses at University of British Columbia Okanagan (UBC O) in Canada, in collaboration with the international student research project sponsors, Laboratoire d'Algorithmique, Complexit et Logique (LACL) in Universit Paris-Est Crteil (UPEC) in France. We are going to discuss a research project, sponsored by LACL: Real-Time Algorithmic Trading System Prototype". In this projects students attempted to develop a new approach for special-purpose programming language code generation from UML models by developing a new plugin for Eclipse IDE [2]. In the present day now it is common to see code generators produce whole sections of code for programmers reducing the time and complexity of a program creation. This technique is also commonly used in businesses where an advanced programmer is not easily accessible so that they can create a program to do what they want quickly and easily with a few clicks of a button. The easiest way to form a set of code is from a well-formed UML transferred to XML form and after that to any form of code you want. Although this is a very convenient way to create code, the problem that occurs is the transformation does not add any real functionality for the program. This is why you need an extra library of information to create the functionality of the code that the generated code could not. So our job is to add onto the current code generation tools to create the functionality that has been requested by our clients for their hedge funds. This program will allow them to create a simple UML diagram and be able to generate code to calculating moving averages" [3]. In some student research projects from 2005 students actively used industrial IDEs, tools and workbenches with code generators, wizards and templates [4]. In papers [4, 5] we discussed some SE student projects in detail. The following OSs, DBMSs, frameworks, IDEs and development tools were used in students' software projects intensively:

- **OS:** Fedora, Ubuntu, CentOS, Windows 2003/2008/XP
- **Virtualization:** VMWare ESX and Workstation virtual hosts
- **DBMS:** Oracle, MS SQL, MySQL
- **IDEs:** Eclipse, NetBeans, BlueJ, MS Visal Studio
- **Version Control:** CVS, SVN, GIT
- **Code generators** are given below. Some of them, such as Oracle Designer, JDeveloper, IBM Rational software Architect and IBM Rational Rose Enterprise allow us to generate source code from the design models or support SE projects implementation into DBMS or into application servers. They also support reverse- and forward engineering. We have started to publish student project results on SourceForge, GitHub, 1and1.com and on the College web-site as well.

1.1 SW Engineering tools and code generators

SW Engineering tools and code generators, which were traditionally used in the SW Engineering courses and student research projects:

- **JDeveloper** "provides a visual WYSIWYG editor for HTML, JSP, JSF, and allows developers to modify the layout and properties of components visually: the tool re-generates

[2] http://www.eclipse.org/

the code. Declarative features enable programmers to generate EJBs or POJOs based on tables in relational databases. JDeveloper automates the creation of Java EE artifacts"[3].

- **Oracle Designer** "is Oracle's CASE tool for designing an information system and generating it. Designer incorporates support for business process modelling, systems analysis, software design and system generation. Designer allows organizations to design and rapidly deliver scalable, client/server systems. After generating the information system one is able to edit the generated code with Oracle Developer Suite"[4]. "Designer incorporates support for business process modelling, systems analysis, software design and system generation. Designer allows organizations to design and rapidly deliver scalable, client/server systems that can adapt to changing business needs"[5].

- **Oracle Application Express (Oracle APEX)** "is a rapid web application development tool"[6]. "With Application Express, coding is declarative. No code is generated or compiled. Declarative code yields fewer differences between developers and this consistency makes Application Express applications easy to maintain and manage"[7].

- **IBM Rational Software Architect (RSA)** "provides integrated design and development support for model-driven development with UML"[8]. "You can use RSA for Java 2 Platform, Enterprise Edition (J2EE Platform) technology, you can convert designs and UML diagrams drawn in the Modeling perspective into code, and also convert the UML models you create into C++ code as needed"[9].

- **IBM Rational Rose Enterprise** "provides a common modeling language for enabling faster creation of quality software, Jump-start your Ada, ANSI C++, C++, CORBA, Java, J2EE, Visual C++ and Visual Basic applications with code generated from visual models"[10].

- **Actifsource** "is a code generator plug-in for the Eclipse IDE. Generic code templates enable code generation. Supported languages are Groovy, Java, C/C++, Scala, etc. Regeneration of code happens in real-time whenever model, meta-model or code templates change"[11].

2 Real-Time Algorithmic Trading System Prototype Design

This project idea was initiated by Youry Khmelevsky at MIT's Computer Science and Artificial Intelligence Laboratory (CSAIL), during the sabbatical in 2010-2011 and crystallized during SE course teaching at Math&STIC Doctoral School at UPEC in December 2012. The idea was supported by Gatan Hains, Professor of Computer Science, UPEC and his PhD student Chong LI (Candidate, Graduate Engineer). We decided to offer a high-performance computing and model-driven development project to Canadian undergraduate students as a possible research project in SW Engineering.
The business domain area of the project was chosen by the sponsors from UPEC: real-time algorithmic trading for a real hedge fund in Paris. Algorithmic trading is widely used by pension funds, mutual funds, some hedge funds, market makers and other institutional traders, to

[3] http://en.wikipedia.org/wiki/JDeveloper
[4] http://en.wikipedia.org/wiki/Oracle_Designer
[5] http://www.oracle.com/technetwork/developer-tools/designer/overview/index-082236.html
[6] http://www.oracle.com/technetwork/developer-tools/apex/overview/index.html
[7] http://apex.oracle.com/pls/otn/f?p=4600:6:0
[8] http://www-01.ibm.com/software/awdtools/swarchitect/
[9] http://www.ibm.com/developerworks/rational/library/05/kunal/
[10] http://www-01.ibm.com/software/awdtools/developer/rose/enterprise/
[11] http://marketplace.eclipse.org/node/1177

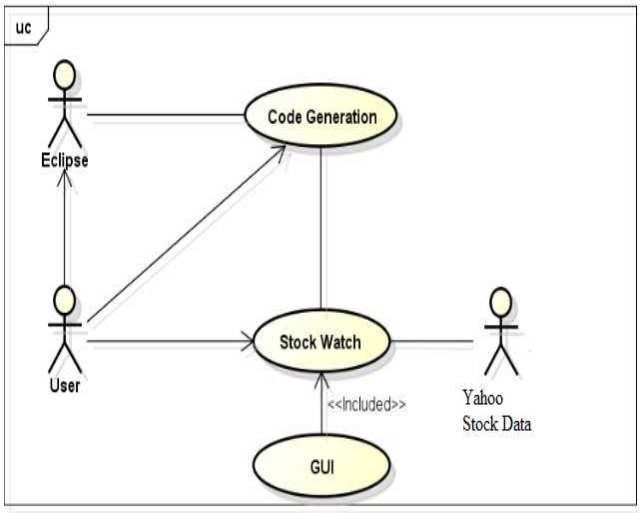

Figure 1: Use Case Diagram of the Real-Time Trading System Prototype.

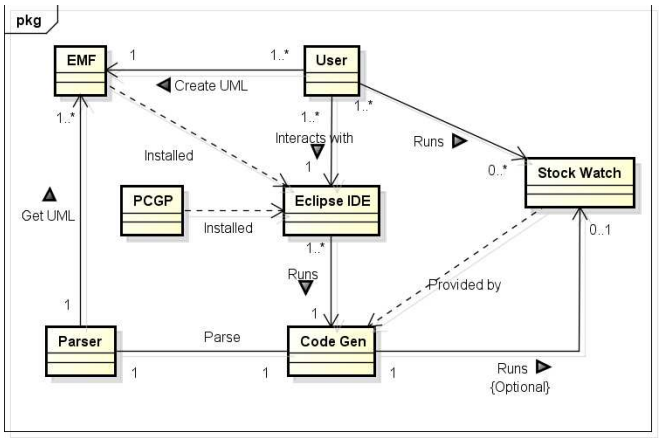

Figure 2: Simplified Domain Model of the Real-Time Trading System Prototype.

manage market impact and risk, to provide liquidity to the market. The technologies of high-performance computing, such as the parallel bridging model - SGL [1], are essential for such systems so as to ensure that complex strategies can process stock-exchange data as milli-second rates. However, many errors can be made with today's tools, for example, the distraction of developers because they must focus both on financial algorithms, parallel computing and coding, or the compiler mis-optimization, etc. The sponsors from UPEC proposed to develop an advanced coding environment for providing a visual and declarative approach to trading algorithms development which can directly generate portable bitcode on Low-Level Virtual Machine (LLVM) from financial specification of trading strategy to avoid the above issues. In this project, students should build a visual and declarative coding environment with which they can generate trading algorithms in Java. During the project, students had the possibility to collaborate with a research engineer from a hedge fund in Paris. With this project, students could also extend their knowledge in several fields: financial trading strategies like mean-reversion[12] or/and pair-trading[13]; parallel computing and HPC models (e.g. SGL model [1]), real-time trading systems, as well as compilation and code generation of programming languages, etc. The Use Case diagram for the real-time trading system prototype is shown on Figure 1, where StockWatch (see Figure 4) must process real-time data from the Yahoo Stock. The simplified Domain Model is shown on Figure 2 [3].

[12] http://en.wikipedia.org/wiki/Mean_reversion_(finance)
[13] http://en.wikipedia.org/wiki/Pairs_trade

We tried to avoid the traditional method in which we must generate different code several times from very high-level abstract to high-level programming languages, then to middle-level and low-level programming languages, until we reach machine code. Instead of that, students should use UML as front-end, and Java (as a first step before an LLVM version) as back-end to generate executable code directly from high-level specification. They firstly investigated how to specify an algorithmic trading strategy using UML. After that, they learned how to build an Abstract Syntax Tree (AST) from UML specification. Then, students prepared a module to generate code automatically from the AST. Students investigated different approaches to find a solution for the problem: model-driven development and automatic code generation for the general purpose languages.

One of the approaches is based on reverse engineering. Reverse engineering is an old activity and a young discipline [6]. "In fact, developers have always struggled with analyzing software components to gather information that the documentation leaves out, examining source code to reconstruct the underlying rationale and design choices, and inspecting data formats to maintain communications among applications. Pushing the adoption of reverse engineering techniques in the development practice is still a major need, requiring appropriate education of developers both in university courses and within industry, and to support reverse engineering techniques and tools with empirical evidence about their performance and usability and with guidelines for their adoption" [6].

In this projects students were developing a tool, which should allow them to generate code from the UML models with additional specifications. As defined in [7], model-driven engineering allows constructing software from abstractions that are more closely fitted to the problem domain and that better hide technical details of the solution space. "Code generation is used to produce executable code from these abstractions, which may result in individual concerns being scattered and tangled throughout the generated code. The challenge, then, becomes how to modularize the code-generator templates to avoid scattering and tangling of concerns within the templates themselves" [7].

Students investigated different approaches, especially related to reverse and forward SE, Eclipse Modeling Framework[14] (EMF) and Model Driven Architecture (MDA).

On the other hand, their research lacked Executable UML (xUML), which is a "rigorous, object-oriented system development method that has been successfully deployed on many strategically critical projects in sectors such as telecommunications, automotive, aerospace and defense, since 1994"[15]. xUmlCompiler[16] can be used to generate java persistence annotated (JPA) classes, which offers a software viewer for the system and generates persistence.xml resources for quick testing turnaround. The xUML as well as Shlaer-Mellor/Executable UML toolset[17], UML/SysML[18]-based modeling and simulation [8] can be used in future research projects.

3 Real-Time Algorithmic Trading System Implementation

We summarize here the results of the Real-Time Algorithmic Trading System Prototype implementation.

The code generation within the student research projects is a new trend in CoSc education. Traditionally, we stressed the study of programming languages and utilization of them, but now we can use more model-driven design and development code generation in our curricula, even within capstone student projects and sponsored projects from Industry.

This project was aimed to develop a Java code generator that takes information from a UML and created a working real-time Java application that may run in parallel. A universal java pro-

[14] http://www.eclipse.org/modeling/emf/

[15] http://www.kc.com/XUML/xumlsupportsmda.php

[16] http://www.ohloh.net/p/xuml-compiler, http://code.google.com/p/xuml-compiler/

[17] http://code.google.com/p/smexuml-tk/

[18] SysML is an OMG standard profile for UML defined for generalSystems Modelling

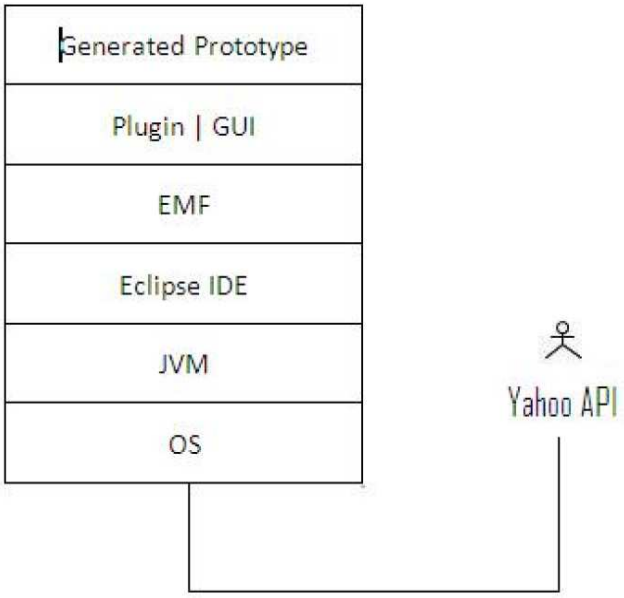

Figure 3: The System Prototype Architecture.

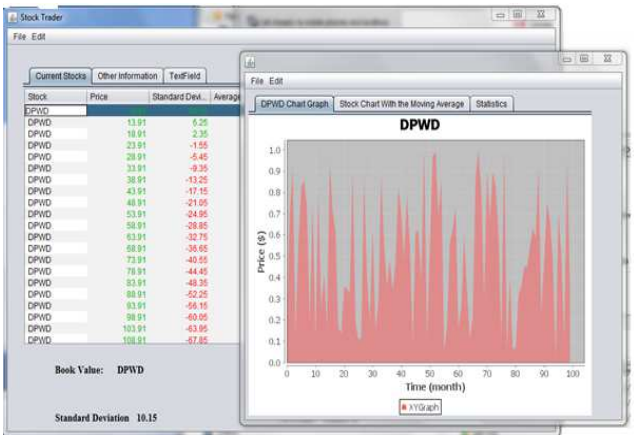

Figure 4: Real-Time Stock Watch Monitor with the Graphical Stock Information Diagram.

ducing system from a UML is impossible to do within the current UML standards, so for this project the students required a narrower focus. This project was focused much more specifically on using the UML as more than an interface and from their parsing out the required data and combined with a library of functions, producing a simple java class that took in the required data and produced a calculated value as output. Creating a java program from a UML was impossible to do universally, so instead what this project aims to do is to use a library of commands that can be called by essentially embedding them into the UML and having a library of commands produce the Java application which will in this case be a real-time trading algorithm with the sole purpose of determining, buy, sell or do nothing [3]. The System Prototype Architecture is shown on Figure 3.

As it is shown on the Figure 2, user uses EMF to create a specification for the plugin. The user calls a plugin from Eclipse with the right click menu. The code generator then calls the parser which takes UML data and then the code generator creates Stock Watch Monitor java application (Stock Watch). The user can execute Stock Watch manually or code generator can execute Stock Watch on completion automatically. The Stock Watch collects data from Yahoo in real-time and produce table with the data results, as it is shown on Figure 4 [3].

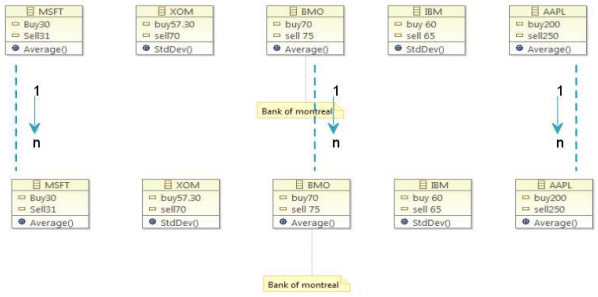

Figure 5: End-User UML Diagram of the On-line Stock Information Sources.

Figure 6: Example of the .ecore XML file.

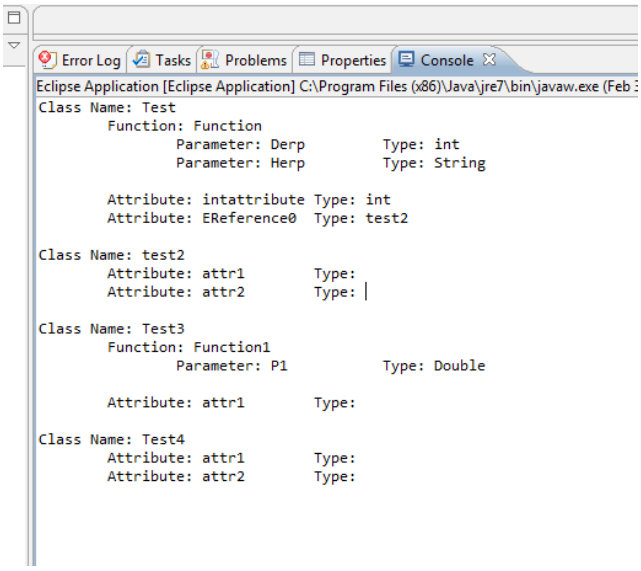

Figure 7: The Parsed Data Example.

3.1 How the Plugin Works

3.1.1 XML Parsing

Using the EMF plug-in we were able to produce a UML diagram of the specifications as it's shown on Figure 5, which is stored as an XML file known locally as an .ecore file (see example on Figure 6.). Using a simple Java parser we extracted the relevant information from the .ecore file and it then gathered the required methods from the library adding them to the generated output. The parsed data is shown on Figure 7.

3.1.2 Generating Code

When the Generate option is selected from the right click menu of a .ecore file in eclipse the GeneratePopupActionClass is called and the run method is called. The run method then creates a ConsoleGenerator object and invokes the generateConsoleApplication method. The generateConsoleApplication method creates a file named OutputClass.java which contains the main for the program to be run. The java file is then compiled and added to a .jar so it can be easily executed. The jar will be located in the root of the C drive and will be named GeneratedApp.jar.
Generation status: currently generates a standard console application based on a set of parameters specified in UML.
Parameters include (see Figure 5):

- Stock symbol of the stock to watch
- For each stock specify the statistic for decision making e.g. Average(). Note: should be declared as a function in UML.
- For each stock specify the BUY and SELL points e.g. BUY34 or BUYAVERAGE

List of Library Functions: The current list of functionality that can be specified in the UML for code generation (not case sensitive):

- average - Specified by the Average.java file in the source.
- stddev - Specified by the StdDev.java file in the source.

Data collection and mathematical calculations done with multiple threads. The GUI has also runs in it own separate thread.

Functions:
To add a function, developers need to do the following:

- Create a java class which computes the desired algorithm value.
- Specify which functions from the class are needed to be publicly accessible
- Submit the class and a description to the development team of the P-est Plugin

To add a function to the distributed library, developers need to do the following (see Figure 5):

- Create a java class which computes the desired algorithm value.
- Specify which functions from the class are needed to be publicly accessible
- Add a function in the FunctionInterface class to access the data from the new class

The Parser plugin-in Class Diagram is shown on Figure 8 execution within Eclipse IDE is shown on Figure 9.

3.1.3 Running the Generated Code

This option works in much the same way as the above Generate option with the one difference being that the created jar is executed after being compile.
The project source code can be accessed from SourceForge.net repository by using the following URL: http://sourceforge.net/projects/pestplugin/files/ [9]

3.2 Regular Usage

3.2.1 EMF Usage

- Create a new Project within Eclipse.
- Create a UML diagram (see Figure 5).

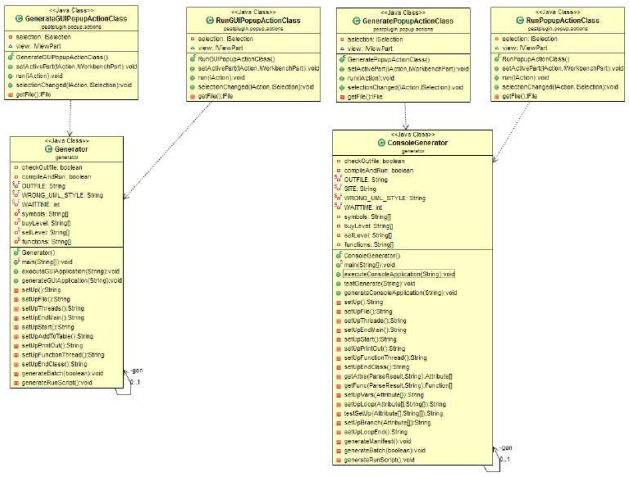

Figure 8: Parser Plug-in Class Diagram.

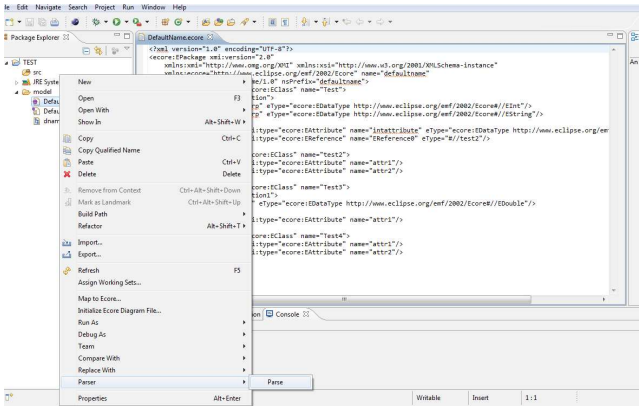

Figure 9: Parser Plug-in within Eclipse.

3.2.2 Plug-in Usage

- Right-click the .ecore file.
- Select Code Generation.
- Select Parser, Generate Code or Generate and Run.
- The output file will be called OutputClass.java and will be placed at the root of the C drive.
- If you selected Generate and Run, the code will also be packaged into a jar and executed.

3.2.3 Detailed instructions for Parsing the Data

This subsubsection explains how the prototype will generate a Java source code:

- The initial stage of the prototype requires that the user create a UML diagram that will fit with the specifications of our library. This will include the algorithms, variables and how they will interact with one another (see Figure 5).
- Upon creation of the UML, an XML is created that will state all the attributes, variables and functions that the user specified in the UML. Using this we can easily obtain information we need to generate code. This is automatically created when a UML is made (see Figure 6).

- Parsing the XML (Ecore File) - Now that we have an XML we can parse the data so that we can obtain the information we need to generate the code. We do so by right clicking our Ecore file and selecting the Parse function (see Figure 9).
- Parsed XML Data - The parsed data will supply the information above. We will have the class name (specified by the user), the functions (from our supplied Libraries), and any parameters along with their data type for storing information (see Figure 7).

4 Conclusion

We have provided here a case study for the real-time algorithmic trading system prototype with automatic Java source code generation from the end-users UML models. During the case study an advanced coding environment was developed for providing a visual and declarative approach to trading algorithms development which can generate directly portable bitcode on LLVM from financial specification of trading strategy to avoid the above issues. Due to lack of project time, the LLVM code generation was not implemented within the proposed prototype system. During the project, Canadian students collaborated with a research engineer from a hedge fund in Paris. Familiarity with existing software development tools is both necessary and encouraged by such projects. The international and collaborative dimension of our latest educational project is also important to introduce students to international, heterogeneous and asynchronous teams.
A new library of mathematical functions for stock interactions and a new plug-in for Eclipse that generates code were developed and implemented during the project.

Acknowledgment

Our thanks goes to the Team Epik from UBC Okanagan: Brandon Potter, Jesse Gaston, Andrew Jankovic, Sam Boateng and William Lee, who successfully designed and implemented the real-time system prototype within CoSc 319 SW Engineering Project in Winter 2012.

References

[1] C. Li and G. Hains. (2011, Jul.) A simple bridging model for high-performance computing. High Performance Computing and Simulation (HPCS), 2011 International Conference on , vol., no., pp.249-256. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber= 5999831&isnumber=5999789

[2] N. Feamster and A. Gray. (2008, Mar.) Can great research be taught?: independent research with cross-disciplinary thinking and broader impact. SIGCSE Bull. 40, 1, 471-475. [Online]. Available: http://doi.acm.org/10.1145/1352322.1352294

[3] B. Potter, J. Gaston, A. Jankovic, S. Boateng, and W. Lee (2012, Apr.) Parallel Code Generation Plug-in XML to Java Plug-in Project Documentation. Team Epik. CoSc 319 "SW Engineering Project" Final Project Report, Computer Science Department, University of British Columbia Okanagan, Kelowna, BC, Canada.

[4] Y. Khmelevsky (2011, May). Research and teaching strategies integration at post-secondary programs. In Proceedings of the 16th Western Canadian Conference on Computing Education (WCCCE '11). ACM, New York, NY, USA, 57-60. [Online]. Available: http://doi.acm.org/10.1145/1989622.1989638

[5] Y. Khmelevsky. (2009, May) Software development pro-jects in academia. In Proceedings of the 14th Western Canadian Conference on Computing Education (Burnaby, British Columbia, Canada, May 01 - 02, 2009). R. Brouwer, D. Cukierman,

and G. Tsiknis, Eds. WCCCE '09. ACM, New York, NY, 60-64. [Online]. Available: http://doi.acm.org/10.1145/1536274.1536292s

[6] G. Canfora, M. Di Penta and L. Cerulo. (2011, Apr.) Achievements and challenges in software reverse engineering. Commun. ACM 54, 4, 142-151. [Online]. Available: http://doi.acm.org/10.1145/1924421.1924451

[7] S. Zschaler and A. Rashid. (2011, Jul.) Towards modular code generators using symmetric language-aware aspects. In Proceedings of the 1st International Workshop on Free Composition (FREECO '11). ACM, New York, NY, USA, Article 6 , 5 pages. [Online]. Available: http://doi.acm.org/10.1145/2068776.2068782

[8] F. Mischkalla, D. He and W. Mueller. (2010, Mar.) Closing the gap between UML-based modeling, simulation and synthesis of combined HW/software systems. In Proceedings of the Conference on Design, Automation and Test in Europe (DATE '10). European Design and Automation Association, 3001 Leuven, Belgium, Belgium, 1201-1206.

[9] B. Potter, J. Gaston, A. Jankovic, S. Boateng, and W. Lee (2012, Apr.) Parallel Code Generation Plug-in XML to Java Plug-in Project Documentation: Source Code and Design Diagrams. Team Epik. CoSc 319 "SW Engineering Project" Final Project Report, Computer Science Department, University of British Columbia Okanagan, Kelowna, BC, Canada. [Online]. Available: http://sourceforge.net/projects/pestplugin/files/