Using the AgileUML program translators

K. Lano

October 22, 2022

1 Introduction

The AgileUML toolset supports program translation from Java, C, VB6 and JavaScript to Python, C#, C++, Swift and Go [3]. It is also possible for users to define and add their own code translators to the tool, and modify existing code translators.

2 Installing AgileUML and Antlr

A complete bundle of the AgileUML and Antlr tools and auxiliary files needed for program translation is in translators.zip at: github.com/eclipse/agileuml and at agilemde.co.uk/translators.zip.

This is for Windows 10+ with Java JDK 1.8 but should run under other Windows environments.

The bundle includes:

1. *umlrsds.jar* – main AgileUML executable
2. *libraries* subdirectory – all language-specific libraries needed for target Swift, C#, etc code.
3. *cg* subdirectory – the abstraction scripts for Java, VB6, JavaScript and C.
4. *uml2py* subdirectory – the code generator for Python
5. *uml2Ca*, *uml2Cb* – code generators for C.
6. Scripts *js2py3.bat*, *java2cpp.bat*, *java2cs.bat*, *java2py3.bat*, *java2swift.bat* – execute the respective translations on the file given as argument.
7. *grun.bat* – Antlr script to execute an Antlr parser and output a parse tree. Uses $antlr - 4.8 - complete.jar$ Antlr tool and the Java, JavaScript, VisualBasic6 and C parser files for Antlr 4.
8. *output* directory containing example Java, C and JavaScript programs and their translations.

To run a translation on the command line, do:

```
js2py3 inputFile
```

The resulting Python file is written to *app.py*. Likewise for *java2py3*. The *java2cs* and *java2cpp* scripts write their output to the *output* directory.

Example files are jsOldClass5Source.txt (JavaScript), BondappSource.txt (Java), QuickSortAsc.txt (VB6) and callocSource.txt (C).

3 Running abstraction and translation from AgileUML

Apart from the translations performed by the shell scripts, other translations can be carried out by abstracting source code to UML/OCL and then code-generating to the target language.

The output/ast.txt files can be produced by running the Antlr parsers for Java, JavaScript, VB6 or C on source code, eg., as:

```
grun Java compilationUnit -tree
grun JavaScript program -tree
grun C compilationUnit -tree
grun VisualBasic6 module -tree
```

Start AgileUML:

```
java -jar umlrsds.jar
```

Necessary libraries can be loaded from the *Extensions* menu option "Import library".

The *File* menu options "From Java AST", "From JavaScript AST", "From C AST" and "From VB6 AST" abstract a parsed source program from *output/ast.txt* file, and produce a UML/OCL specification in the main AgileUML window (Figure 1).

The abstracted specification can be viewed in detail using the *Edit* menu option "Edit KM3" (Figure 2).

Quality checks such as checks for clones can be performed from the *View* menu (optional).

Type check the specification using the *Synthesis* menu option, then generate target code using *Build* menu options. Java8 code is produced by the Android app generation option, and Swift code by the iOS app generation options.

Output code is usually written to the *output* subdirectory, but Python code is written to *app.py* in the main directory.

Depending on the source code, various support libraries may need to be imported, prior to abstraction:

1. mathlib.km3 – mathematical and financial operations and byte manipulation/bitwise operators

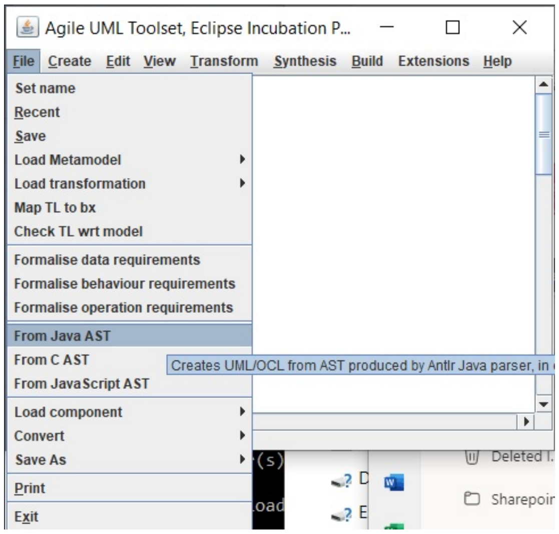

Figure 1: Abstracting programs to specifications

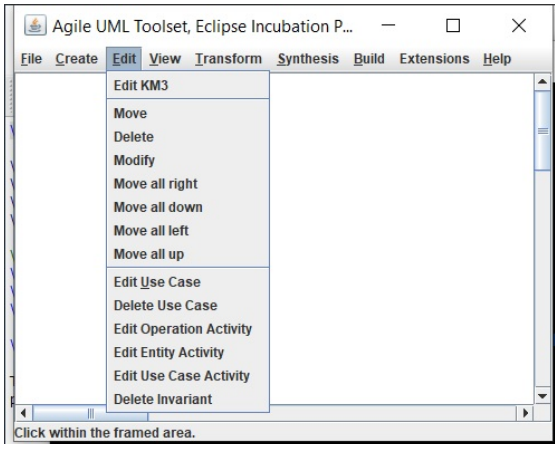

Figure 2: Viewing abstracted specification

2. ocldate.km3 – for OclDate type
3. oclexception.km3 – supports exceptions
4. oclfile.km3 – supports text files
5. oclprocess.km3 – supports processes, threads and operating system access
6. oclrandom.km3 – random number generation
7. ocltype.km3 – supports reflection
8. ocliterator.km3 – for iterators
9. ocldatasource.km3 – for databases, TCP sockets and HTTP. This depends on ocliterator, oclfile and ocldate.

3.1 Restrictions and limitations

For Java abstraction, we cover Java versions 6 and 7, and the most widely-used libraries in java.lang, java.io, java.util and java.math. Swing and other UI code is not abstracted. There is limited support for Internet/database processing. File support is primarily for text files.

For C, the ANSI C subset of [1] is covered, however pointers only translate with certain restrictions. File support is for text files.

For JavaScript, ECMAScript 2015 classes are abstracted, but not all cases of older style definitions of classes by constructor functions or object literals. The block scope of ECMAScript 2015 is adopted.

For VB6, only the core language is supported. Gotos and gosub and error handling are not supported.

4 Modifying the abstraction and code-generation tools

The abstraction and code-generation scripts can be extended or customised as required.

The abstraction scripts are in cg/Java2UML.cstl, cg/JS2UML.cstl and cg/C2UML.cstl, cg/VB2UML.cstl, together with their associated auxiliary scripts. These take as input parse trees in *output/ast.txt* produced by Antlr parsers for Java, JavaScript, C and VB6, respectively.

The CSTL scripts can be executed via the *Build* menu option "Apply CSTL to AST", which loads a selected *.cstl* file from the *cg* subdirectory, and applies this to *output/ast.txt*.

The scripts are written in $\mathcal{CSTL}$ [2] and can be edited with a text editor.

To write an abstraction script for a new programming language, use a structure similar to the VB2UML.cstl abstractor, with rulesets specific to the grammar of the new language. It is also possible to use the *antlr2cstl* script to generate an outline CSTL abstraction file for a language *L* supported by an Antlr grammar L.g4. Parse the L.g4 grammar (parser rules only) using the ANTLRv4 grammar of Antlr:

```
grun ANTLRv4 rules -tree
```

to obtain an AST of the L grammar, place this in output/ast.txt and run *antlr2cstl*.

5 Further information

The latest version of AgileUML is at https://github.com/eclipse/agileuml, code generator and abstraction scripts are in cg.zip and supporting libraries in libraries.zip.

A substantial dataset of translation examples can be found at: zenodo.org/record/7107604.

References

[1] B. Kernighan and D. Ritchie. *The C Programming Language*. Prentice Hall, 2nd edition, 1988.

[2] K. Lano, 2022. Using the code generator language CSTL, agilemde.co.uk/cgrules.pdf.

[3] K. Lano. Program translation using Model-driven engineering. In *ICSE 2022 Companion Proceedings*, 2022.