TETE: A Source Transformation Environment for Eclipse

Derek M. Shimozawa James R. Cordy

School of Computing, Queen's University
Kingston, Ontario
Canada, K7L 3N6
{derek, cordy}@cs.queensu.ca

Objective:

Source transformation is a programming paradigm that models a software solution as a syntactic conversion from the problem domain to the solution domain. It is rapidly becoming a mainstream paradigm for a wide range of common software problems in computer science. Transformation frameworks such as *ASF+SDF*, *Stratego, XSL/T*, *DMS Reengineering Toolkit*, and *TXL* have been applied in software engineering, data mining, document recognition, software static analysis, Internet commerce, and the semantic web. While a number of these frameworks have seen great deal of success and growth in both the commercial and academic research sectors, the lack of accessible development tools presents a formidable obstacle for new users. Furthermore, these technologies are difficult to learn, and require a radical cognition shift from procedural and object-oriented programming languages such as C/C++, Java, Perl, and BASIC, among others. To overcome these problems, we propose the *Transformation Engineering Toolkit for Eclipse* (*TETE*), a set of Eclipse plug-ins that introduces a unique programming interface specifically aimed to support learning about, authoring, maintaining, and interactively exploring transformations specified with the TXL source transformation language.

About TXL:

A TXL transformation is comprised of two subparts: an arbitrary, context-free BNF grammar describing the syntactic structure of a source artifact, and a collection of rules that defines how the artifact is to be transformed into the resulting output (Figure 1).

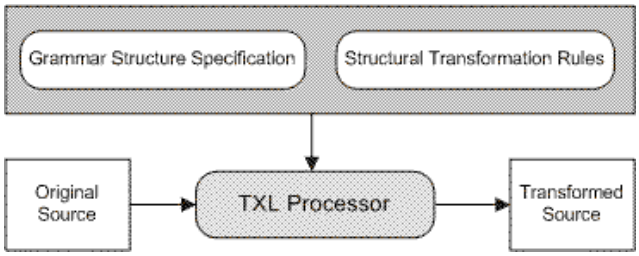

Figure 1: The TXL Processor

Transformations in TXL are executed to a fixed point by repeatedly applying a main rule to the source input until it fails. During this process, each sub-rule is invoked by its parent rule, in the style of a recursive, functional program.

Methodology:

TETE offers a collection of transformation-centric tools within the Eclipse environment. It consists primarily of a syntax-colored TXL editor, a *Transformation Topological View*, unit-testing support, a *Rapid Transformation Input Window*, and grammatical support for C++, Java, and XML transformations.

Transformation Topological View: Programming of source transformations is supported by multiple, simultaneous views of a program's source. Typically, users are only concerned with a small subset of a program's underlying grammar, which is determined by a rule's scope of application. The TETE environment features the Transformation Topological View to present BNF grammar types and nested rule sets as hierarchical tree schemas. Tree elements are linked to the transformation editor and its content outline view, allowing new users to explore the structure of their programs and focus on sections and levels of current interest.

Unit Testing Support: Interactive exploration of TXL programs is supported by a unit-testing interface that enables users to select the entry point for transformation and automatically create test cases on a rule-by-rule basis. The effectiveness of a per-rule testing approach is two-fold. Firstly, it is an interactive learning tool. Students can explore TXL rule evaluation in manageable fragments by devising their own examples. Secondly, it is a powerful debugger. Students can break down a transformation into separate test cases to find semantic errors in their rule constructs, gradually building up to the entire program.

Rapid Transformation Input Window: TETE encourages a by-example approach to learning by allowing new users to control their program launches through the Rapid Transformation Input Window. It is a powerful tool for experimenting with various parts of a transformation, and leverages TXL's pure functional style to create a reactive programming experience for users. Whereas TETE's unit testing functionality focuses on testing individual rules, the Rapid Transformation Input Window focuses on swift experimentation and rule application. Users can devise their own examples and receive instant feedback from the system to facilitate an understanding of their program's behavior and the language's semantics.

Grammatical Support: Support for transformation of C++, Java, and XML is built into TETE and the ability to add grammar modules for other languages makes the system extensible. TETE allows for the creation of "cookie-cutter" transformation projects that are pre-configured with TXL resource files. Users can add new default transformation projects to the TXL Project Wizard by adding grammar, rule set and instruction markup files to a designated resources directory in the TETE UI plug-in. This feature is useful for instructors that wish to customize the plug-in so that it comes pre-configured with specific labs or tutorials, saving significant setup time in undergraduate courses.

Demonstration Outline:

We will demonstrate the features listed above by providing a step-by-step example:

- We will add a new default project to TETE using an existing TXL grammar. A new transformation exercise will be instantiated using this grammar.
- Next, we will program a set of rules to demonstrate TETE's editing and source exploration facilities. In particular, we will use the Transformation Topological View to explore the program's nested rule structure and underlying grammar type hierarchy.
- We will develop a unit testing strategy to perform program exploration and debugging. The design of our testing strategy will be correlated to the source structure discovered using the Transformation Topological View.
- The Rapid Transformation Input Window will be used to show how we can use unit testing and rapid input in unison to effectively perform program exploration and debugging.