TXL Source Transformation in Practice

James R. Cordy
School of Computing, Queen's University
Kingston, Ontario, Canada K7L 2N8
Email: cordy@cs.queensu.ca

***Abstract*—The TXL source transformation system is widely used in industry and academia for both research and production tasks involving source transformation and software analysis. While it is designed to be accessible to software practitioners, understanding how to use TXL effectively takes time and has a steep learning curve. This tutorial is designed to get you over the initial hump and rapidly move you from a TXL novice to the skills necessary to use it effectively in real applications. Consisting of a combination of one hour lecture presentations followed by one hour practice sessions, this is a hands-on tutorial in which participants quickly learn the basics of how to use TXL effectively in their research or industrial practice.**

***Keywords*—TXL, source transformation, platform migration, static analysis, reverse- and re-engineering, rapid prototyping**

I. Introduction

TXL [1] is a programming language designed explicitly for authoring source transformation tasks of all kinds. Unlike most other source transformation tools, TXL is completely self-contained - all aspects of the source transformation, including the scanner, parser, transformer and output pretty-printer are all written in TXL. Because it has no dependencies on external parsers, frameworks or libraries, TXL programs are completely self-contained and easily portable and distributable across platforms and operating systems.

TXL is a mature technology that is widely used in academia and industry in tasks such as software analysis and reverse engineering, software renovation and re-engineering, platform, language and library migration, natural language requirements extraction, clone detection, code instrumentation, and rapid prototyping of DSLs and other new languages. More than 100 companies are currently using TXL worldwide, and more than 100 refereed academic papers on research using TXL have been presented at SANER and its sister conferences over the past decade.

It is the engine that powers the LS/2000 design recovery system [2], the NICAD code clone detector [3], the SIMONE model clone detector [4], commercial software modelling tools such as Embarcadero *Describe*, software analysis services such as those offered by IBM Global Services, and custom software analysis and migration tools used in companies such as Siemens, Ericsson, Lockheed Martin and General Motors.

Because of its simplicity and self-contained nature, TXL is widely used to introduce source transformation and software analysis tools and techniques in graduate and undergraduate courses. Nevertheless, it has a steep learning curve and it takes some practice to understand how to use it effectively in any particular domain of application.

II. Aims & Audience

The purpose of this one day tutorial is to address that issue by rapidly bringing participants up to speed in TXL, in order to accelerate their progress and get them "over the hill" so that they can begin to use TXL effectively in their work. The broader aim of the tutorial is to rapidly bring those interested in understanding and using TXL in particular and source transformation tools in general up to speed on how to effectively use these tools in practice. The emphasis is on the practical use of the tools rather than the theory. Example problems from the SANER domain are used throughout.

The intended audience is SANER researchers and industrial practitioners facing problems involving software code analysis, transformation or translation who are exploring technical alternatives including TXL. The tutorial is aimed at TXL novices, and no previous exposure to TXL is expected or required. Some familiarity with compiler technology basics is an asset.

III. Organization & Schedule

The tutorial is designed as a mix of tutorial presentations and hands-on practice sessions in which participants can practice the techniques presented on their own computers, under the interactive guidance of the presenter. Each presentation introduces a new level of sophistication and technology, which is then practiced in the following practice session. Content is roughly based on the TXL Cookbook [5], a collection of TXL problem schemas and paradigms of use designed to cover a range of software analysis and evolution applications.

There are three parts to the tutorial, each consisting of a presentation session followed by a hands-on practice session. In order to facilitate learning and understanding, participants work together in groups of two on lab problems in the practice sessions.

Part 1: TXL Basics

We begin with a basic introduction to TXL, for those who are new to it. Topics include the TXL paradigm, the TXL processor, anatomy of a TXL program, specification of lexical and syntactic forms, input parsing, base grammars and grammar overrides, transformation rules and functions, patterns and replacements, deconstructors and constructors, basic TXL program authoring strategy, and how to understand TXL.

The hands-on lab session for Part 1 consists of installing and running TXL, installing and working with a language grammar, making a basic TXL program, parsing inputs, and making and refining simple TXL transformations. Lab sessions use the popular PHP scripting language as the target language.

Part 2: Parser-based Techniques

Part 2 concentrates on parsing techniques, the foundation of all work in TXL. The presentation is example-based, working through several typical parser-based problems using a concrete example toy language. Topics include crafting a TXL grammar, making a basic parser, pretty-printing using TXL, designing and implementing language extensions, robust parsing, island grammars and semi-parsing, agile parsing to simplify rules.

In the lab session for Part 2, participants explore techniques for restructuring programs using TXL in the context of a concrete restructuring problem to improve information hiding in PHP classes. Transformation techniques practiced include reordering of program elements, working with multiple copies, filtering to separate elements of interest, interface extraction, program reorganization.

Part 3: Rule-based Techniques

Part 3 introduces paradigms for program transformation and analysis using TXL rules. A small set of representative problems in software restructuring, optimization, and static and dynamic analysis serve as concrete examples to demonstrate TXL programming paradigms. Planned topics include scopes of application, grammatical type extraction and filtering, negative patterns, transforming to a fixed point, dependency sorting, deep pattern match, one pass rules, context-dependent rules, iteration, attributes, grammatical abstraction, generalization and specialization, complex conditions, dynamic output, counting and statistics, program markup, fact extraction.

Part 3's lab session involves implementing two real PHP analysis problems: implementing a scoped dialect of the language to address plugin interference, and static call graph extraction for PHP. These are challenging real problems which help participants to gain experience and confidence in their use of TXL. Small prizes are offered for the best solutions crafted by teams in the tutorial.

IV. Outcomes

At the end of the tutorial participants will be well-versed in the basics of TXL and the paradigms of its application to problems in programming language parsing and source transformation, software translation and migration, language extensions and DSL rapid prototyping, static and dynamic analysis, reverse- and re-engineering, program metrics and statistics, and other kinds of software engineering tasks based on manipulation of source code.

V. Materials

Participants download a free copy of the FreeTXL software, examples and documentation from the TXL website [6], and work on their own laptop computers in a team environment in the practice lab sessions. Copies of all presentations and lab materials are available to tutorial participants. Modest prizes are awarded for the best problem solutions crafted by participants in the lab practice sessions.

About the Presenter

James Cordy is Professor and past Director of the School of Computing at Queens University at Kingston, Canada. As leader of the TXL source transformation project with hundreds of academic and industrial users worldwide, he is the author of more than 160 refereed contributions in programming languages, software engineering and artificial intelligence. From 1995-2001 he was Vice President and Chief Research Scientist at Legasys Corporation, whose TXL-based LS/2000 source code analysis system was responsible for the analysis and reprogramming of over 4.5 billion lines of financial code of the largest Canadian banks for the Year 2000 problem. Dr. Cordy is an ACM Distinguished Scientist, a senior member of the IEEE, and an IBM CAS faculty fellow.

References

[1] James R. Cordy, "The TXL Source Transformation Language", *Science of Computer Programming* 61,3 (August 2006), pages 190–210.

[2] Thomas R. Dean, James R. Cordy, Kevin A. Schneider and Andrew J. Malton, "Using Design Recovery Techniques to Transform Legacy Systems", Proc. *ICSM 2001, 17th Intl. Conf. on Software Maintenance*, pages 622–631.

[3] Chanchal K. Roy and James R. Cordy, "NICAD: Accurate Detection of Near-Miss Intentional Clones Using Flexible Pretty-Printing and Code Normalization", Proc, *ICPC 2008, 16th IEEE Intl. Conf. on Program Comprehension*, pages 172–181.

[4] James R. Cordy, "Submodel Pattern Extraction for Simulink Models", Proc. *SPLC 2013, 17th Intl. Software Product Line Conf.*, pages 7–10.

[5] James R. Cordy, "Excerpts from the TXL Cookbook", *Generative and Transformational Techniques in Software Engineering III, Lecture Notes in Computer Science* 6491 (January 2011), pages 27-91.

[6] TXL Programming Language Website, *http://www.txl.ca*