Science of Computer Programming 72 (2008) 52–70

Contents lists available at ScienceDirect

Science of Computer Programming

journal homepage: www.elsevier.com/locate/scico

Stratego/XT 0.17. A language and toolset for program transformation

Martin Bravenboer [a], Karl Trygve Kalleberg [b], Rob Vermaas [c], Eelco Visser [a,*]

[a] *Department of Software Technology, Faculty of Electrical Engineering, Mathematics and Computer Science (EWI), Delft University of Technology, Mekelweg 4, 2628 CD Delft, The Netherlands*

[b] *Department of Informatics, University of Bergen, PB 7800 N-5020 Bergen, Norway*

[c] *Machina, Utrecht, The Netherlands*

ARTICLE INFO

Article history:
Received 7 November 2005
Received in revised form 12 September 2007
Accepted 14 November 2007
Available online 1 May 2008

Keywords:
Stratego
Stratego/XT
Program transformation
Rewriting strategies
Rewrite rules
Concrete syntax
Dynamic rewrite rules

ABSTRACT

Stratego/XT is a language and toolset for program transformation. The Stratego language provides rewrite rules for expressing basic transformations, programmable rewriting strategies for controlling the application of rules, concrete syntax for expressing the patterns of rules in the syntax of the object language, and dynamic rewrite rules for expressing context-sensitive transformations, thus supporting the development of transformation components at a high level of abstraction. The XT toolset offers a collection of flexible, reusable transformation components, and tools for generating such components from declarative specifications. Complete program transformation systems are composed from these components.

This paper gives an overview of Stratego/XT 0.17, including a description of the Stratego language and XT transformation tools; a discussion of the implementation techniques and software engineering process; and a description of applications built with Stratego/XT.

1. Introduction

Automatic program transformation and *generative programming* aim at increasing programmer productivity by automating programming tasks using some form of automatic program generation or transformation, such as code generation from a domain-specific language, aspect weaving, optimization, or specialization of a generic program to a particular context. Key for achieving this aim is the construction of *tools* that implement the automating transformations. If generative programming is to become a staple ingredient of the software engineering process, the construction of generative tools itself should be automated as much as possible. This requires an infrastructure with support for the common tasks in the construction of transformation systems.

Stratego/XT is a generic infrastructure for creating stand-alone transformation systems [47,52]. It combines *Stratego*, a language for implementing transformations based on the paradigm of programmable rewriting strategies, with *XT*, a collection of reusable components and tools for the development of transformation systems. In general, Stratego/XT is intended for the *analysis*, *manipulation* and *generation* of programs, though its features make it useful for transforming any structured documents.

Reusability at all levels of granularity has been a leading principle in the design and implementation of Stratego/XT [47]. First, the focus on *transformation components* strongly promotes reuse of large-grained components. In many cases, users of Stratego/XT do not start with the development of a parser, but can immediately get started with the actual transformation.

* Corresponding author.
E-mail addresses: martin.bravenboer@gmail.com (M. Bravenboer), karltk@strategoxt.org (K.T. Kalleberg), rob.vermaas@gmail.com (R. Vermaas), visser@acm.org (E. Visser).

0167-6423/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.scico.2007.11.003

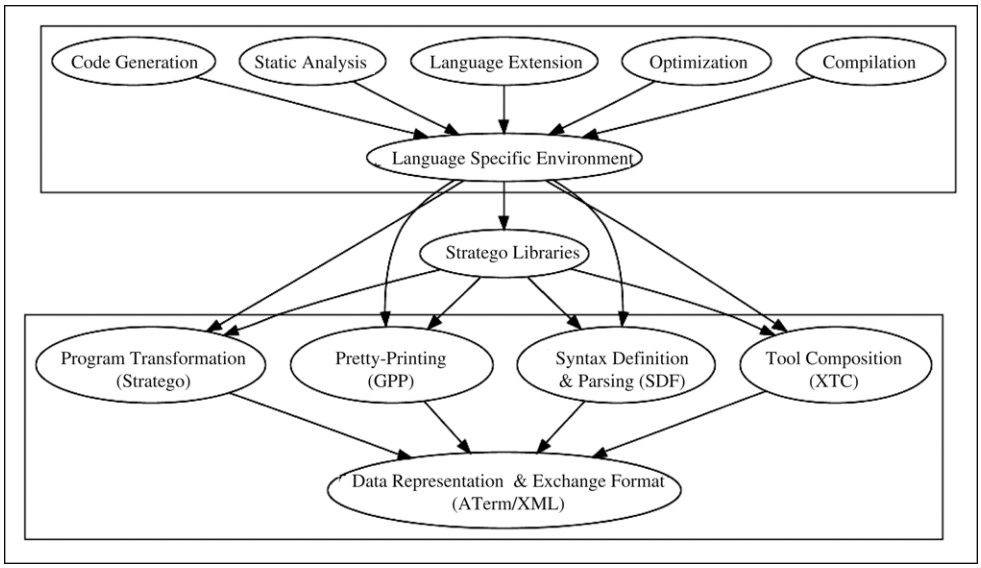

Fig. 1. Reuse layers in Stratego/XT.

Stratego/XT has a varied selection of actively developed front-ends (Section 5). Second, the use domain-specific languages (DSLs) for different phases of a transformation system is a substantial time saver. In addition to the Stratego language itself, which is a DSL for transformation proper, Stratego/XT provides and/or integrates DSLs for aspects such as syntax definition, pretty-printing, term schemas, and unit testing. In this way, implementations are more abstract, easier to maintain, and easier to read. Third, the extensive Stratego library, with its generic traversals, generic transformations for scoping, control-flow and data-flow and many other convenience functions for program transformation allows developers to write their transformations concisely.

The component layers in Fig. 1 illustrate the organization of Stratego/XT from the point of view of reusability, featuring five levels of components ranging from completely generic via language specific to application specific.

(1) At the bottom layer is the *substrate* for a transformation system, that is the data representation and exchange format, for which we use the Annotated Term Format (ATerm) [36] as basis, and XML where necessary to communicate with external tools.

(2) The *foundations* of any transformation system are syntax definition and parsing, pretty-printing, program transformation, and tool composition. The syntax definition formalism SDF provides modular syntax definition and parsing, supporting easy combination of languages. The pretty-printing package GPP supports rendering structured program representations as program text. The program transformation language Stratego supports concise implementation of program transformations by means of rewrite rules and programmable strategies for control of their application. Finally, the XTC library supports composition of transformations implemented as independently executable tools. These are all generic facilities that are needed in any transformation for any language.

(3) In the middle is a library of transformations and transformation utilities that are not specific for a language, but not usable in all transformations either. The Stratego Libraries provide a host of generic rewriting strategies and utilities for generating parts of a transformation system.

(4) Near the top are specializations of the generic infrastructure to specific object languages. Such a *language-specific environment* consists of a syntax definition for a language along with utilities such as semantic analysis, variable renaming, and module flattening.

(5) Finally, at the top are the actual *transformation systems*, such as compilers, language extensions, static analysis tools, and aspect weavers. These tools are implemented as compositions of tools from the lower layers extended with components implementing the specific transformation under consideration.

Stratego/XT has been used to build many types of transformation system, including compilers, interpreters, static analyzers, domain-specific optimizers, code generators, source code refactorers, documentation generators, and document transformers. These systems involved numerous types of transformation, including desugaring of syntactic abstractions; assimilation of language embeddings [17]; bound variable renaming; optimizations, such as function inlining; data-flow transformations, such as constant propagation, copy propagation, common-subexpression elimination, and partial evaluation [14,34]; instruction selection [16]; and several analyses including type checking [15] and escaping variables analysis.

This paper gives an overview of the design, implementation, and use of Stratego/XT 0.17. Section 2 outlines the technical foundations of the Stratego language and describes the compiler and interpreter that implement it. Section 3 describes the transformation infrastructure provided by the XT tool set. Section 4 examines the implementation techniques and