

TCS - Project plan for an Eclipse GMT component

Scope of the TCS component

Goals

The objective is to support the definition and usage of text-based Domain-Specific Languages (DSLs) using model-based tools, which provide:

- 1) means to specify textual syntaxes for metamodels;
- 2) translation capabilities from text-based DSL sentences to their equivalent model representation (in EMF), and vice-versa, based on the syntax specification;
- 3) an Eclipse text editor with: syntax highlighting, an outline, hyperlinks, and hovers, also based on the syntax specification.

Context

The text-to-model problem is classically solved by defining a grammar, and then using one of the many available parser generators (e.g., yacc, ANTLR). Model-to-text is generally handled separately by implementing a visitor that serializes its source model into an equivalent textual representation. This requires two separate encodings of the same syntax: grammar and visitor. For model-based DSLs a third non-syntactic specification (i.e., the metamodel) is also required. However, there is a significant redundancy between these elements. For instance, information already available in the metamodel needs to be duplicated in the grammar (e.g. multiplicity of elements). Parse trees then need to be converted into models either by tree walkers (i.e. visitors) or using annotations in the grammar. These are not only tedious to specify but also depend on the chosen parser generator.

Approach

TCS (Textual Concrete Syntax) is a DSL designed for the specification of textual concrete syntaxes. It works by providing means to associate syntactic elements (e.g., keywords like `if`, special symbols like `+`) to metamodel elements with little redundancy. This achieves goal 1).

Both model-to-text and text-to-model translations can be performed using a single TCS specification. A grammar can thus be generated from both the metamodel and the TCS model to perform text-to-model translation. Grammar annotations that build the model while parsing can be automatically generated. Model-to-text translation can also be performed with the same information. To this end, a generic interpreter has been defined to traverse the model following the syntactical path specified in TCS. Keywords and symbols are written alongside model information. TCS effectively bridges the modeling and grammar worlds. This achieves goal 2).

Many of the problems related to textual concrete syntaxes are already solved in the Grammarware Technical Space (TS). There is no reason to rebuild such facilities in the Model Engineering TS. What we need is a projector between these spaces. TCS is a language that allows specification and automatic generation of projectors between the Grammarware TS and the Model Engineering TS per given DSL.

An overview of the usage of the TCS language is shown in Figure 1. Assume we want to build a DSL called L. In MDE TS we provide a metamodel of L named MM_L expressed in

KM3. The definition of the concrete syntax is expressed in TCS and is denoted as CS_L . The required bridge between the two technical spaces consists of an injector and an extractor. The injector takes a model in L expressed in the textual concrete syntax of L and generates a model conforming to MM_L in the MDE TS. An example model is denoted as SM_L and it conforms to the grammar of L denoted as G_L . G_L is expressed in ANTLR. The extractor generates textual representation of models in the Model Engineering TS conforming to MM_L . Figure 1 shows an example in which a model M_L is extracted to SM_L .

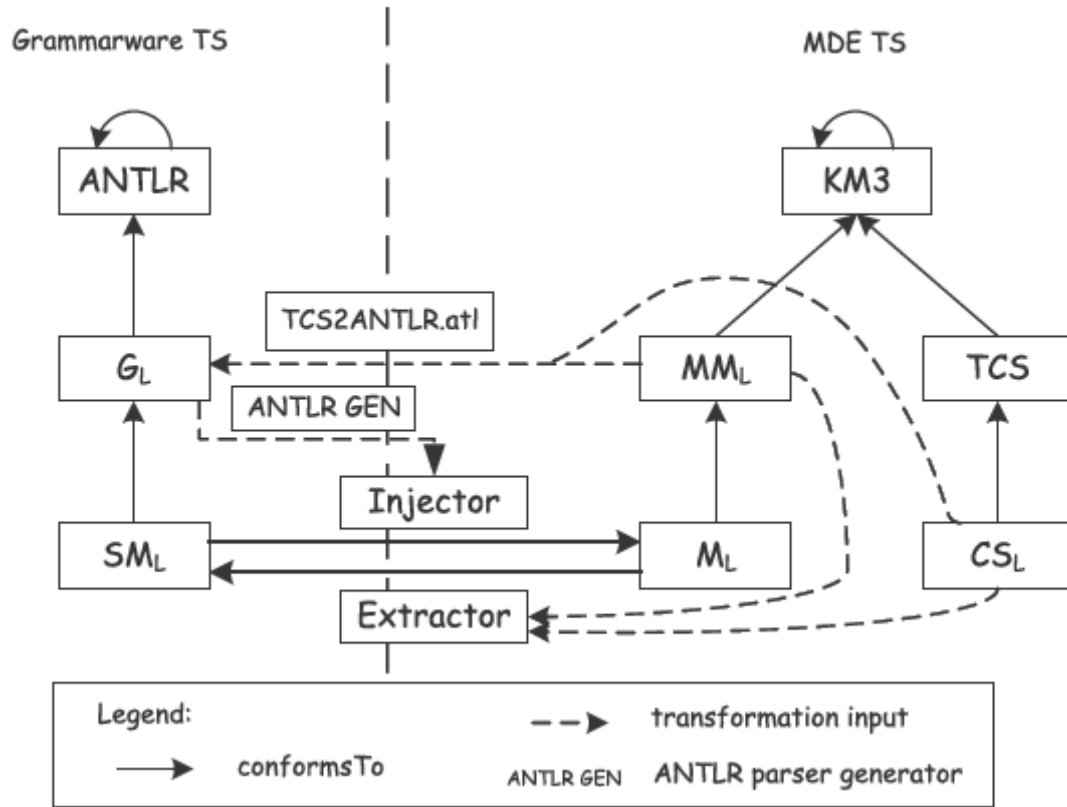


Figure 1. Overview of TCS usage

The approach we take starts with the metamodel and the concrete textual syntax description of a given language L. Our goal is to obtain three entities for L: its annotated grammar G_L expressed in ANTLR, and the couple of injector and extractor. G_L is generated by an ATL transformation named TCS2ANTLR.atl. It takes MM_L and CS_L as input (shown with dashed lines), and generates the rules and the annotations in G_L . This grammar is used to create the injector. The injector is a parser generated by the tools provided by the ANTLR technology. The generation is done by the ANTLR parser generator (denoted as ANTLR GEN).

The extractor works on the internal representation of models expressed in L and creates their textual representation. It is possible to generate an extractor per every language L. However, the current approach consists of a single extractor implemented as an interpreter that works for any language. The extractor takes a model M_L written in L, its metamodel MM_L , and its TCS syntax description CS_L and generates the textual representation SM_L of M_L .

Using TCS is typically simpler than developing ad-hoc injectors and extractors. One specification is enough for both directions. Moreover, redundancy between a TCS model and its corresponding metamodel is reduced (e.g., property multiplicity and type are omitted in TCS).

Additional features of TCS:

- **Traceability.** The current implementation of TCS provides text-to-model traceability by keeping line and column information in models.
- **Generic Editor.** Textual Generic Editor (TGE) is a tool that partly builds on TCS services. TGE provides a text editor which is parameterized by information gathered from TCS models. An outline (i.e., tree representation of a program) is generated using TCS text-to-model ability. Hyperlinks and hovers (i.e., automatic display of the target of a link) are provided using text-to-model traceability. TGE achieves goal 3).

Organization

Initial contribution

The initial contribution of TCS will be available during summer 2007, for initial prototype and a library of examples.

TCS community

TCS was developed by the ATLAS Group. It is already (and more and more) used by the Eclipse community. We will encourage user communities to use the technology and contribute to improvement through feedback. We will also encourage user and external developers to contribute examples, use cases, and code.

Work Plan and schedule

In the next year, new major releases are planned. It is hoped that new users and contributors will join the subproject as committers.

Initial Development Team

The initial core development team of TCS consists of:

- Frédéric Jouault, INRIA (Component Lead)
- Freddy Allilaire, INRIA
- Mikaël Barbero, INRIA
- William Piers, Obeo

Interested Parties

TBA

References

- Jouault, F, Bézivin, J, and Kurtev, I: [TCS: a DSL for the Specification of Textual Concrete Syntaxes in Model Engineering](#). In: GPCE'06: Proceedings of the fifth international conference on Generative programming and Component Engineering. 2006.
- Jouault, F, and Bézivin, J: [On the Specification of Textual Syntaxes for Models](#). Eclipse Modeling Symposium at the first Eclipse Summit Europe, Esslingen, October 11-12, 2006.
- Abouzahra, A, Jouault, F, Kurtev, I: [TCS Presentation made by SODIUS and INRIA ATLAS](#).