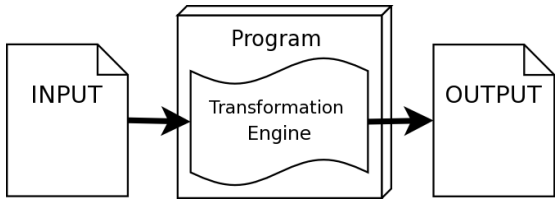
transformation languages

Introduction

Transformation languages are widely used for to process different kinds of data. They can be used to solve/support/help in the following situations:

- How do I get data in format A to format B?
- How do I change the representation of data in a certain format?
- How can I change / transform the design of a certain task without changing it's logic

The common and single work flow can in general be described as the following:



- The input data can be a well formed or/and hierarchical or/and abstract set of information. It can even be a stream of data.
- The transformation engine may have knowledge of both models/formats (and metamodels), as well it can have (kind of) transformation rules. They can be internal or external, depending of the specific use case. Stylesheets for XSLT processors can be seen as external rules.
- The output is the product of the whole process, it might be well formed or/and hierarchical or/and abstract set of information. It can even be a stream of data.

(All the definitions can be seen as optional and is depending on the use case.)

Models and metamodels

Models and metamodels play in many use cases of transformation languages an important role:

The transformation engine has to understand its input, this can be done by some hard coded model or it can be done by applying metamodel information to the input. In XML-cases this can be the DTD.

Transformation languages can even be used to transform models and metamodels of the transformation processes itself.

Examples

txl

TXL is a unique programming language specifically designed to support computer software analysis and source transformation tasks.
The TXL programming language is a hybrid functional / rule-based language with unification, implied iteration and deep pattern match.
TXL is best at tasks that involve structural analysis and transformation of formal notations such as programming languages, specification languages, structured documents and so on.

It can be used for pretty printers, syntax checking, automation in software maintenance and it supports many more kinds of transformations.

Program transformation

A program is a structured object with semantics. The structure allows us to transform a program. The semantics give us the means to compare programs and to reason about the validity of transformations.

Stratego

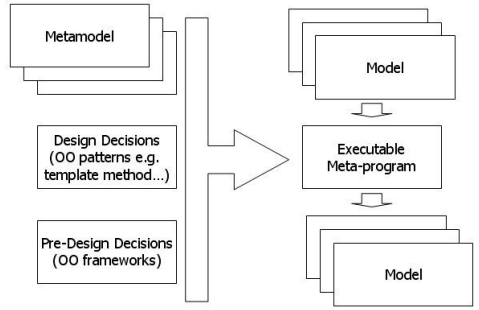
Stratego is a language for program transformation based on term rewriting with programmable rewriting strategies.

Strategic Term Rewriting:
- Select rules to use in a specific transformation
- Select strategy to apply
- Define your own strategy if necessary
- Combine strategies

So we can use use Stratego for example to refactor programs, as we can extract additional information from the structure and semantic.

Kermeta

Kermeta is a tool which enables us real model to model transformations. It uses informations of a specific metamodel and will translate its model to another model.



Conclusion

Transformation languages help us to deal with different formats or models. They can help us to transform them in other formats or models, as well they can give us different representations of a model.

(Meta-)Modeling helps a lot in the process of transformation. They help us to get a certain level of abstraction and reusability.

We can use transformation languages if we have to deal with a lot of different formats, we want different representations and so on.

Resources

- Txl: http://txl.ca/
- Stratego: http://www.program-transformation.org/Stratego/WebHome
- Kermeta: http://www.kermeta.org/
- Pierre-Alain Muller, Franck Fleurey, Didier Vojtisek, Zoé Drey, Damien Pollet, Frédéric Fondement, Philippe Studer, and Jean-Marc Jézéquel. -- On executable meta-languages applied to model transformations. -- In *Model Transformations In Practice Workshop*, Montego Bay, Jamaica, October 2005.
- Master Course Program Transformation (Material): http://www.cs.uu.nl/wiki/Pt