Generation of DSL Tools based on Language Definitions

Miguel Garcia
http://www.sts.tu-harburg.de/~mi.garcia
2007-03-29
Agenda

• What it means “Language Definition” of a DSL

• Existing prototypes
  – What subset of a Language Definition they can process
  – What outputs they generate
  – (and which ones they don’t, i.e.: shortcomings)

• Outlook
How complete a “Language Definition” can be

1. Offer a textual syntax

2. Containing definitions of well-formedness. Also called “static semantics”. For example, each usage is in scope of its single previous declaration.

3. Type-safe, i.e. no runtime execution may assign a value of type T2 to a location declared to hold values of type T1, with T2 not a subtype of T1.

4. Define behavior of programs written in the DSL (far fetched)

Items 2 and 3 can be captured in a metamodel with OCL invariants

Those OCL invariants can be evaluated after the user has edited a document using the (generated) DSL tools

Metamodels of Java cover item 1. What about items 2 and 3? Is it that difficult to write OCL???
Well-formedness example: EJB3QL (OO DB query language of Java EE)

Comparing values from different enumeration types makes no sense, but the grammar does not rule it out.

Solve that by adding an OCL invariant to the metamodel of EJB3QL:

context EnumCompExp

inv comparedValuesBelongToTheSameEnumerationType :
  left.type() = right.type()

(More examples at: Formalizing the well-formedness rules of EJB3QL in UML + OCL, ATEM 2006 Workshop, co-located with MoDELS / UML 2006)
Sample (far from trivial) DSLs

- **BPEL metamodel (with OCL for well-formedness)**

- **DASL, Sun Labs,**
  a “model to deploy” web application specification language,
  abstracting away from persistence, transaction, and distribution

- **SecureUML, ETH Zurich's Information Security Group,**
  “Security mechanisms are rather easy to state but difficult to realize …
  starting from access-control descriptions, code and security configuration are generated”

- **WebML, Dipartimento di Elettronica e Informazione at Politecnico di Milano,**
  “a visual notation for specifying the content, composition, and navigation features of hypertext applications”
Existing prototypes

- **xText**,  
  www.openarchitectureware.org/

- **TCS, by Frédéric Jouault (INRIA)**  
  http://wiki.eclipse.org/index.php/TCS

- **TEF, by Markus Scheidgen (Uni Humboldt)**  
  http://www2.informatik.hu-berlin.de/sam/meta-tools/tef/tool.html

- **Gymnast, by Chris J Daly (IBM)**  
  used to extend EMF Emfatic to support generics as described in  
  http://www.sts.tu-harburg.de/~mi.garcia/EIG2006/  
  ImprovementsToTheEmfaticEditor.pdf

- **SAFARI (just about to become an Eclipse project**  
  pages/safari.Documentation.html
Lessons learned from adding Generics to Emfatic (or, could an IDE generator have done that?)

All right, some use cases have a fair chance of being generated without requiring later customization:

- Hoover showing declaration for some usage, Folding

- Hyperlinks
Even an IDE generator needs OCL as input sometimes

- **Consider Ecore Validation**

- **Consider ContentAssist beyond grammar rules**
  - as required for example to fill in a type parameter: only those in scope should appear in the suggestions list
And sometimes, not even OCL is enough

- EMF Type Hierarchy

- Generation of an .ecore for the input .emf ("compilation" in the case of Emfatic)
We’ll spend some more time with Gymnast

- As of now it can generate grammars for different parser generators, thus encapsulating the parsing functionality and making parsers interchangeable
- As well as Java classes for the Concrete Syntax Tree (CST)

We plan to extend Emfatic with functionality for:

- generating EMF-compliant classes for Concrete Syntax Tree nodes. With that, using OCL (Object Constraint Language) becomes possible to check the well-formedness of CSTs before transforming them to ASTs. The author of such transformation may thus employ QVT-style technology, e.g. ATLAS ATL.

- generating an LPG grammar in addition to ANTLR and JavaCC.

- generating a visitor to unparse a CST into a pretty-printed representation, re-using hints specified in the existing Gymnast grammar format. For example,

```plaintext
sequence ifConstruct :
  "if" expr
  "then" expr
  "else" expr
  "endif"
```
Outlook (2 of 3)

We plan to extend Emfatic with functionality for (continued):

- generating a text editor with usability features. The design patterns to generate have been harvested while implementing the Emfatic extension for EMF Generics. These usability features include:
  
  - Outline view, Mark occurrences, Folding,
  - Hovers displaying declaration for a given usage,
  - Live problem and warning markers, Navigable hyperlinks,
  - “File New” wizard, AutoEdits such as SmartBrace (i.e. automatically completing with a closing terminal),
  - User-provided templates,
  - Syntax-aware Content Assist.
  - Refactoring will be explored (at the very least, by updating usages-for-declarations)
Field testing the resulting prototype. Candidate DSLs:

- an ExecutableUML-style language but EMF-based

- EJB3QL, building upon the metamodel (including well-formedness rules but not yet type-checking rules) described in http://www.sts.tu-harburg.de/~mi.garcia/pubs/atem06/EJB3QLpaper.pdf

- a human-readable notation for UML2