Dániel Darvas

Domain-specific languages (DSLs): what, how and when?

ICE Tea
21/02/2014
Outline

- **Theory**  *Concept of DSLs*

- **Technology**  *Support for DSLs*

- **Reality**  *Some details of the ST Example*
DSL | Theory / Concept
“Any fool can write code that a computer can understand. Good programmers write code that humans can understand.”

(Martin Fowler)
What is a DSL?

- “a computer (programming) language of limited expressiveness focused on a particular domain” (Fowler)
- Opposite of “general purpose language” (GPL)
- “mini-language”
DSL examples

- SQL

```sql
SELECT *
FROM accelerators
WHERE energy > 10000.0
ORDER BY name;
```
A "domain-specific language" ("DSL") is a computer language specialized to a particular application.
DSL examples

- SQL
- Wiki markup
- Refrigerator *(Bosch und Siemens Hausgeräte)*

```plaintext
compressor compartment cc {
  static compressor c1
  fan ccfan
}
```
DSL examples

- SQL
- Wiki markup
- Refrigerator
- “Accelerators” language

```plaintext
complex CERN {
    source H_source
    linear accelerator LINAC2 source: H_source
    circular accelerator PSB source: LINAC2
    circular accelerator PS source: PSB
    ...
    fixedtarget experiment nTOF source: PS
}
```
GPL or DSL?

General purpose

Domain-specific

SQL vs. ST (Structured Text)

ST vs. Java

Java vs. English

− Is it just about *programming* languages?
GPL or DSL?

- HTML?

Hypertext Markup Language (HTML) Tim Berners-Lee, Daniel Connolly, and the W3C HTML Working Group

Hypertext Markup Language (HTML)
A Representation of Textual Information and MetaInformation for Retrieval and Interchange

<h1>, <p>, <a href=''>
Motivation for DSLs

- More **expressive**, less redundant
  - ⇒ More efficient

- Good **learning curve**

- Helps the **communication** with domain experts
  - “An algorithm must be seen to be believed.” (D. Knuth)

- Can be **self-documenting**

- Domain-specific **validation**
Typical usage

- **Internal DSLs**: user is a developer
  - Usually transformed to another language

- **External, but focused on domain-specific users**
  - E.g. Mathematica
  - Processed internally

- But typically not for the broad public
  - Not graphical or “fool-proof” (free text) languages

- ~ When XML+XSD can be used
# Graphical or textual?

<table>
<thead>
<tr>
<th></th>
<th>General purpose</th>
<th>Domain specific</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Textual</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Graphical</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Learning curve
- Efficiency
- Understandability

- Depends on the goals, but graphical does not kill the textual language
Pro et contra

- Pros
  - **optimised** for humans
  - comfortable, **efficient**
  - easier **communication**

- Cons
  - cost of building:
    - Usually you need a lexer.
    - And a parser.
    - And an internal data model.
    - And an editor.
    - Preferably with syntax highlight, content assist, …
DSL / Technology

© Bill Wrigley
What do we want?

“User” level

Good/efficient syntax

Editor

GAP
(reserved for magic)

“Developer” level

Object model (AST)
## Solutions

<table>
<thead>
<tr>
<th></th>
<th>User demands</th>
<th>Developer demands</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DSL, without support</strong></td>
<td>Syntax is good, but no editor (N++)  😊😊</td>
<td>Text instead of AST 😞</td>
</tr>
<tr>
<td><strong>XML+XSD</strong></td>
<td>Syntax is more difficult, no good editor 😞</td>
<td>Object model can be obtained “easily” 😊</td>
</tr>
<tr>
<td><strong>DSL, with support</strong></td>
<td>Syntax is good, useful editor 😊</td>
<td>Object model can be obtained “easily” 😊</td>
</tr>
</tbody>
</table>
Tool support

- **Xtext** (itemis)
  - Widely used
  - Textual input
  - External parser

- **MPS** (JetBrains)
  - Textual & various graphical inputs
  - Different philosophy (no parsing)

- **Spoofax**
  - Academic (research)
  - Textual input
  - Advanced parser
Xtext

- Eclipse and EMF-based
- Big community
  - “Stackoverflow Driven Development”
  - XtextCON May 27-28, Kiel (Germany)
- Open source, actively maintained
  - Last stable release: Feb 12, 2014
- They eat their own dogfood (Xtend)
- Based on the extended grammar of the language
(Formal) grammar

- grammar = description of the language
- definition of the syntax

- elements: rules (recursive)

B.1.2.3.2 Time of day and date
PRODUCTION RULES:

```
time_of_day ::= ('TIME_OF_DAY' | 'TOD') '#' daytime
daytime ::= day_hour ':' day_minute ':' day_second
```

- from IEC 61131
Xtext grammar

- enriched grammar – not just syntax
- goal: generate “everything” from the grammar

Experiment:

```plaintext
‘experiment' name=ID
'source:' source=[Accelerator];
```

experiment NA source: SPS

Accelerator:

```plaintext
(linear?=‘linear' | circular?=‘circular')
'accelerator' name=ID 'source:' source=[SrcOrAcc];
```

circular accelerator LHC source: SPS
The full “Accelerators” grammar

grammar ch.cern.en.ice.tea.accelerator.AcceleratorGrammar with xtext.Terminals

Complex:
  'complex' name=ID '{' (items+=ComplexItem)* '}';

ComplexItem: (SourceOrAccelerator | Experiment);

SourceOrAccelerator: Source | Accelerator;

Source: 'source' name=ID;

Accelerator:
  (linear?='linear' | circular?='circular') 'accelerator' name=ID 'source:'
  source=[SourceOrAccelerator];

Experiment:
  type=('fixedtarget') 'experiment' name=ID 'source:'
  source=[SourceOrAccelerator];

complex CERN {
  source H_source
  linear accelerator LINAC2 source: H_source
  circular accelerator PSB source: LINAC2
  circular accelerator PS source: PS
  ...
  fixedtarget experiment nTOF source: PS
}
What have the Romans Xtext ever done for us?

- EMF object model
  - + parser
  - + reference handling
What have the Romans Xtext ever done for us?

```java
public void printCircularAccSrcs(Collection<Accelerator> c) {
    for (Accelerator acc : c) {
        if (acc.isCircular()){
            System.out.println(acc.getSource().getName());
        }
    }
}
```
More Xtext features

- fancy editor
  - integrated into Eclipse
More Xtext features

- fancy editor
  - integrated into Eclipse
  - content assist

- references ("jump to")

- ...
“What a magic tool!”

- Eclipse built-in features
- **Everything comes from the grammar**
  - If the grammar is bad or “weak”: manual work
  - If external data is needed: manual work

- **Example**: circular $\rightarrow$ linear transition is forbidden
  - Validation rule is needed (~ 5 lines of code)

- + 20-30 LoC to fix the content assist
Code generation

- Not part of DSLs, but typically included

- Xtext encourages **Xtend**
  - Java “dialect”
  - Compiled to Java
  - Supports **templates**

```
A="5", B="3"; A+B="8"

"A="" + a + "", B="" + b + ""; A+B="" + (a+b) + ""

'''A="«a»", B="«b»"; A+B="«a+b»"'''
```
Code generation

Source

+  

Templates

```
def static dispatch visualizeItem(Accelerator acc)
    «IF acc.circular»
        «acc.name» [label = "«acc.name»", shape=oval, fillcolor=darkolivegreen2];
    «ELSE»
        «acc.name» [label = "«acc.name»", shape=box, fillcolor=darkolivegreen3];
    «ENDIF»
    «acc.source.name» -> «acc.name»;
```

LHC [label = "LHC", shape=oval, fillcolor=darkolivegreen2];

LINAC2 [label = "LINAC2", shape=box, fillcolor=darkolivegreen3];
Code generation

H_source

LINAC2

PSB

PS

SPS

AD

EA

nTOF

LHC

NA
Difficult parts

− Validation

− Scoping
  • “Which variable i is accessed?”

− Expression handling
  • Parsing 5+ab+c^2
  • Priority

− Testing the magical black box
DSL | Reality (Some details from ST example)
ST grammar for verification purposes

- Goal: generating formal models from ST code
- Method:

```
<table>
<thead>
<tr>
<th>ST code</th>
</tr>
</thead>
<tbody>
<tr>
<td>ST object model</td>
</tr>
<tr>
<td>Intermediate model</td>
</tr>
<tr>
<td>Formal model</td>
</tr>
</tbody>
</table>
```

By Xtext

Custom Java

Custom Xtend
Example: struct vs. FB instance definition

**VAR**

```plaintext
v1 : NamedStructure; // used-defined type (UDT)
v2 : TON; // timer function block
```

**StructureVariable:**

```
name=ID ':' structure=[StructureDeclaration];
```

**FBVariable:**

```
name=ID ':' fb=[FBDeclaration];
```

- **Xtext's parser cannot distinguish between the two rules**

**StructureOrFBVariable:**

```
name=ID ':' structureOrFB=[StructureOrFBDeclaration];
```

- **Grammar of var. definitions in the standard: ~ 6 pages**
Example: variable assignment

- \( v := \text{TRUE}; \)

[Variable] '\=' Expression

- \( v[0] := \text{TRUE}; \)

[Variable] ([‘ index=INT ‘])? '\=' Expression

- \( \text{struct1.v := TRUE}; \)

- \( \text{struct1.array1[0].struct2.v := TRUE}; \)

- \( \text{DB100.struct1.array1[0,1].struct2.v := TRUE}; \)

- General grammar + many validation rules
Example: differences between the languages

<table>
<thead>
<tr>
<th>ST</th>
<th>NuSMV</th>
</tr>
</thead>
<tbody>
<tr>
<td>v1 := true; V1 := True;</td>
<td>V1 := TRUE;</td>
</tr>
</tbody>
</table>
Example: differences between the languages

<table>
<thead>
<tr>
<th>ST</th>
<th>NuSMV</th>
</tr>
</thead>
<tbody>
<tr>
<td>v1 := true; V1 := True;</td>
<td>V1 := TRUE;</td>
</tr>
<tr>
<td>v2 := 10.5;</td>
<td>V2 := 0sd32_10500;</td>
</tr>
</tbody>
</table>
## Example: differences between the languages

<table>
<thead>
<tr>
<th>ST</th>
<th>NuSMV</th>
</tr>
</thead>
<tbody>
<tr>
<td>v1 := true; V1 := True;</td>
<td>V1 := TRUE;</td>
</tr>
<tr>
<td>v2 := 10.5;</td>
<td>V2 := 0sd32_10500;</td>
</tr>
<tr>
<td>v3 := 1;</td>
<td>V3 := TRUE; V3 := 0sd16_1; V3 := 0ud32_1; V3 := 0sd32_100;</td>
</tr>
</tbody>
</table>
Example: scoping

FUNCTION_BLOCK FuncBlock
VAR
  v : INT;
END_VAR

v := 123;
DummyFunction(in := 1);
DummyFunction(in := 1, v := v);
...

But is ST a “mini-language”?
DSL | Summary and Conclusions
Summary

- The concept is useful – for small, dedicated languages
- Tools help to develop the DSL toolchain

- New languages
  - Develop one, if it helps you and it will be regularly used

- For existing languages
  - If you need the **AST** for a “small” language: go for Xtext
  - If you need just an **editor**: Xtext or “plain Eclipse editor”
  - For **big languages** (C, Java, …): Xtext is not powerful enough

- Where can it be used at CERN?
Where can be useful for us?

- Specification language

- UNICOS User templates?
  - Supporting template-based code generation is difficult
  - JET, Velocity, Xtend, T4, … won’t provide validation, syntax highlight or content assist
  - How to provide grammar and validation rules?
  - Can be possible with (not template-based) custom solution

- Hopefully in many other cases…
Links

- **Xtext**
  https://www.eclipse.org/Xtext/

- **Xtext documentation**
  https://www.eclipse.org/Xtext/documentation/2.5.0/Xtext%20Documentation.pdf

- **Xtext tutorial**
  https://www.eclipse.org/Xtext/documentation.html#FirstFiveMinutes

- **MPS**
  http://www.jetbrains.com/mps/

- **Spoofax**
  http://strategoxt.org/Spoofax
Books

- M. Fowler: **Domain-Specific Languages** (2010)
  *Available on SafariBooks*

- L. Bettini: **Implementing Domain-Specific Languages with Xtext and Xtend** (2013)
  *Available on SafariBooks*

- M. Völter: **DSL Engineering** (2013)
  *Available on dslbook.org*
“Any fool can write code that a computer can understand. Good programmers write code that humans can understand.”

(Martin Fowler)