What Every Xtext User Wished to Know – Industry Experience of Implementing 80+ DSLs

EclipseCon Europe 2016

2016-10-26

Roman Mitin
Avaloq

- Technology-driven financial services provider
  - 450 customer sites
  - 4 BPO centres
  - 2’200 employees
  - 22 offices in 10 countries

- Own software platform for core, web and mobile banking
  - 1’350+ certified implementation experts
Domain-specific Languages at Avaloq

- **Standard DSLs**: WSDL, RAML, XSD, XML
- **Avaloq DSLs**: PL/SQL API, Java API
- **Not a DSL**: Technical DSLs: i.e. Oracle objects, Business DSLs: BPMN, business objects & rules, Visual DSLs with tabular or graphical editors

100+ source types
Lessons of First Project with Xtext
Lessons of First Xtext Project: Editors for Existing Languages

- Favor explicit design of AST
- With Xtext we still model source code
  - Model lists as lists (order, duplicates)
  - Validate node model if needed
- Parser, recovery and partial reparse
  - Relax grammar and write checks
  - Avoid unordered groups
  - Syntactic predicates can be slow
  - No to backtracking for prototyping
- Extend Xtext if needed
  - Semantic predicates, custom lexer…
Lessons of First Xtext Project: Editors for Existing Languages

- Favor explicit design of AST
- With Xtext we still model source code
  - Model lists as lists (order, duplicates)
  - Validate node model if needed
- Parser, recovery and partial reparse
  - Relax grammar and write checks
  - Avoid unordered groups
  - Syntactic predicates can be slow
  - No to backtracking for prototyping
- Extend Xtext if needed
  - Semantic predicates, custom lexer…
Lessons of First Xtext Project: Editors for Existing Languages

- Favor explicit design of AST
- With Xtext we still model source code
  - Model lists as lists (order, duplicates)
  - Validate node model if needed
- Parser, recovery and partial reparse
  - Relax grammar and write checks
  - Avoid unordered groups
  - Syntactic predicates can be slow
  - No to backtracking for prototyping
- Extend Xtext if needed
  - Semantic predicates, custom lexer…
Scale All-In-Memory Index of Xtext

- Developer checks out few sources
- Only incremental builds in IDE

- Prebuilt index

- Sources (~90k, 600 MB text)
- Resources
- Exported objects (~2M)
- References (~10M)
- Issues

- Layered view on index
- Local H2 DB for index delta
  - Changed & invalidated models
Implementation Standards for DSLs with DSLs

Available on GitHub
Check DSL

Xbase enhanced for model validation

```java
/** SCA Check Documentation. */

live error "Procedure name too long" (int maxLength = 10)
message "Procedure name exceeds {0} of characters." {
for ProcedureDeclaration p {
    if (p.name.length > maxLength) {
        issue on p#name bind (maxLength.toString)
    }
}
}
}
raise a diagnostic

procedure xxxxxxxxxxxxxxxxxxxxxxxx
is
begin
    null;
exception
```
Format DSL
Semantic-aware source text formatting

NamedArgument:

target=[typeModel:INamedParameter|ID] "=>' value=Expression;

NamedArgument {
    rule : linewrap 0 1 1 after,
    no_linewrap after { (context.eContainer as ArgumentList).arguments.size == 1 };
    "=>' : column {FormatUtil::getLongestNameLength(context)} relative before
    { (context.eContainer as ArgumentList).arguments.size > 1 };
}

```
22  procedure bar
23  is
24  begin
25  test1(i_id=>2,
26  i_some_long_name =>'text',
27  i_comment=>'comment');
28  test2(i_id=>3);
29  end bar;
```

```
30  procedure bar
31  is
32  begin
33  test1(
34   i_id        => 2
35   ,i_some_long_name => 'text'
36   ,i_comment  => 'comment'
37  );
38  test2(i_id => 3);
39  end bar;
```
Requires stable URI fragments

- "semi-positional" syntax with selectors for not unique names
  e.g. 0/1/3(0=='foo') . 0 and 0/1/3(0=='foo') . 1 for objects 'foo'
- for unique names add '!' and omit last part: e.g. 0/1/3(0=='foo' !)

- Price: every visible declaration must have an exported object
Export DSL

Resource description strategy, fingerprints, fragment provider

Calculation method for object fingerprints

Object descriptions visible in index queries

Index entries for fine-grained invalidation with fragment structure

```typescript
import http://www.avaloq.com/tools/dsl/avqscript

interface {
    ScriptPackage = name;
    MethodDeclaration = @+modifiers, @parameters;
    FunctionDeclaration = @returnType;
    Declaration = name;
    ...
}

export lookup ScriptPackage as qualified name {
    object-fingerprint;
    data public = this.public;
}

// Only non-private methods should be exported
export MethodDeclaration as name ![this.private] {
    uri-fragment = unique attribute (name);
    object-fingerprint;
}
```

Fingerprint for fine-grained invalidation

Defines stable segment for fragment provider
Scope DSL (1/2)
Naming, scope provider, implicit references, caches

Default naming functions for EClasses

Reference-specific scope for feature “type” of EClass ObjType doing Index lookup

Invariant for resource

All EObjects of EClass Row from code_obj_type namespace
Type-specific named scope with simple Expression for KeyDef context object

Named scope with scope chain containing delegation to a named scope, index query and local expression

```java
scope (implicitVariables) INamedElement {
    context KeyDef = this.getImplicitVariables();
}

scope (decls) INamedElement {
    context CalculationBlock = this.getImplicitVariables() >> find( avqscript::ScriptPackage ) as name >> this.getLocalDeclarations();
}
```

- Delegation to a named scope
- Chain of three scopes
- Explicitly defined name function
Xtext Extensions & Tips

- Distributed Builder
- Object Fingerprints
- Testing with Xtend
- Parser Delegation
- ANTLR Semantic Predicates
- Xtext
- ...
Testing with Xtend
Easy to write and understand tests

class LabelLinkingTest extends AbstractDdkScopingTest {

    @Tag int LABEL_A

    override registerRequiredSources() {
        addSourceToWorkspace("TEST.LABEL DEF",
            "label definitions test
                label «mark(LABEL_A)» label.a
                fallback "fallback"
            end label definitions"
        );
    }

    @Test
def testLabelLinking(){
        validateLinks("TEST.E.LABEL XLATE",
            "label translations test.e
                translation «ref(LABEL_A)» label.a
                text "text"
            end label translations"
        );
    }
}
Test Failure Reporting

Clear test failure messages

```java
@Test
public void testDuplicateTranslationOfKernelLabel() {
    validateCustomerSourceStrictly("MDB$BAR.E.LABEL XLAT.", "'
        label translations mdb$bar.e
        translation LABEL.ALL
        text "bar"
    end label translations
    '
}
```
Parser Delegation

Where inheritance fails: mix DSLs with incompatible lexers

- Call parser from value converter
- Merge the node model returned by the parser

```java
AvaloqScriptExpression returns avqscript::Expression:
    PLACEHOLDER

@ValueConverter(rule = "AvaloqScriptExpression")
public IValueConverter<Expression> avaloqScriptExpression() {
    return new AvqScriptExpressionDelegateParser<Expression>(delegate);
}
```

- Editor services need to be aware of delegation
  - Validators need to be initialized with right injector
  - DocumentTokenSource need to be aware of multiple token classes

Text within placeholder [...] is handled by a different DSL.
ANTLR Semantic Predicates

Listing keywords in ID rules? ANTLR generated code too big?

- Predicate generated for keyword rule

  ```java
  /**
   * @KeywordRule(visible, invisible)
   */
  KeywordVisibleKind returns VisibleKind :
  ID
  ;
  
  public boolean isKeywordVisibleKindEnabled(final ParserContext parserContext) {
    final String text = parserContext.getInput().LT(1).getText();
    return "visible".equalsIgnoreCase(text) || "invisible".equalsIgnoreCase(text);
  }
  
  ** Custom predicate referenced from Xtext grammar **

  ```java
  /**
   * Language ID pattern: "dfltlang" .. | "german" | ("lang_" ID)
   */
  Translation :
  language=[ILanguage|Language] (text=STRING | "null")?
  ;
  ```

Adjusted Xtext workflow inserts (<predicate call>)?=> semantic predicates for ANTLR input (*.g).

Propagate predicates before ANTLR actions across all rules.
We talked about
- Context, database-backed index, layered index
- DSLs for scoping, formatting, object descriptions, and validations
- Xtend-based test suite for linking, content assist, validations etc.
- Composition of Xtext languages
- Semantic predicates for keyword rules etc.

What we could also talk about
- Headless DSL compiler
- Types outside defining resource: managing derived resources
- User-defined model validations and user-defined transformations

Want to work together on some of this in dsl-devkit?
- Try https://github.com/dsldevkit
- Contact roman.mitin@avaloq.com

Questions?
Thank you for your attention.

roman.mitin@avaloq.com
Evaluate the Sessions
Sign in and vote at eclipsecon.org

-1  0  +1