Migrating code with SmaCC

John Brant
brant@refactoryworkers.com
Migration Strategy

- Define Parser
- Create transformation program
- Compatibility layer

✓ Normal development continues
- Keeps same design — garbage in garbage out

SmaCC
Parser Definition

- LALR(1)/LR(1) parsers
- GLR
- AST generation
- Pattern matching
AST Definition

<number> : \d+ (\s? \d)*? ;
<whitespace> : \s+;

%left "+";
%root Expression;
%suffix Node;

Expression
  : Expression 'left' "+" 'operator' Expression 'right' {{Binary}}
    | "(" 'leftParen' Expression ")" 'rightParen' {{Expression}}
    | <number> 'value' {{Number}}
  ;
Transformation Program

• Ordered list of transformation rules + methods and properties

• Declarative Pattern Rules
  Quick to write
  One-off expressions

• Imperative Code Rules
  General syntax
  Control flow
Pattern Rules

- Search expression pattern-based AST
- Replace expression is pattern-based string
Pattern Matching

<number> : \d+ (\.* \d*) ? ;
<whitespace> : \s+;
<patternToken> : `[^`]+``;

%glr:
%left "+";
%root Expression;
%suffix Node;

Expression
  : Expression 'left' "+" 'operator' Expression 'right' {{Binary}}
  | "(" 'leftParen' Expression ")" 'rightParen' {{Expression}}
  | <number> 'value' {{Number}}
  ;
Pattern Example

Source:

\[ 3 + 3 \]

\[ \text{BinaryNode: +} \]

\[ \text{NumberNode: 3} \]
\[ \text{NumberNode: 3} \]

Search Pattern:

\[ \text{\`a` + \`a`} \]

\[ \text{BinaryNode: +} \]

\[ \text{Anything: \`a`} \]
\[ \text{Anything: \`a`} \]

\[ \text{\`a` = 3} \]
Replace Expressions

- Replacement pattern is string macro
- Original source replaced with expanded macro
- Matched pattern nodes rewritten before replacement string is generated

Replacement: \`a\` * 2

Matched patterns: \`a\` = 3

Result: 3 * 2
Pattern Examples

for `a` := `b` to `c` - 1 do `d` ⇒
    for (`a` = `b`; `a` < `c`; `a`++) `\d\`

`a`/Forms.TCustomForm`.Constraints.MinHeight := `b` ⇒
   `a`.MinimumSize = new Size(`a`.MinimumSize.Width, `b`)
Code Rules

- Smalltalk expressions
- Search expressions based on AST node and code
- Replace expressions
  - Edit expressions
  - Control flow
  - General Smalltalk code
Edit Expressions

- Custom framework messages for editing source

- Replacing
  \#replace:with: \#replaceAll:with: ...

- Moving
  \#move:before: \#move:after ...

- Inserting
  \#insert:before: \#insert:afterAll: ...

- Deleting
  \#delete: \#deleteWithWhitespaceAfter: ...
Control Flow

- Normal traversal is depth first
- Change the order that nodes are traversed
  #processChild: #processChildren #continue
"{ }"
For: DelphiStatementBlockNode
When:
  true
Do:
  self replace: match beginToken with: '{'.
  self replace: match endToken with: '}'.
  self continue

"function objects"
For: PBTypeDeclarationNode
When:
  match from source sameAs: 'function_object'
Do:
  self isStatic: true.
  self classStart: match startPosition.
  self
    replace: match
    with: 'public partial class ' , self functionsClassName , ' {'
Parser Debugger

Input:

```javascript
var sqlite = require('sqlite3');
var db = new sqlite.Database('Var/lib/weex/weex.sdb');

var http = require('http'),
    url = require('url'),
    path = require('path'),
    fs = require('fs');
const PORT = 8060;
const staticDirectory = path.join(process.cwd(), 'files');

function convertToDate(value) {
    return new Date(value * 1000);
}

function convertToTs(date) {

}

getNextToken
```
Rule Debugger

Debugging PBForwardPrototypesNode

Original Source:

```java
public partial class WEventHandling : Synchrony.Forms.SpclForm {

public static WEventHandling w_eventHandling;

public string ls_crif = "\r\n";

forward prototypes
public subroutine newline ()
public subroutine clear ()
public subroutine log (string logtext)
public subroutine logevent ()
end prototypes

public subroutine newline ();
```

Run to Cursor:

```java
public static WEventHandling w_eventHandling;

public string ls_crif = "\r\n";

forward prototypes
public subroutine newline ()
public subroutine clear ()
public subroutine log (string logtext)
public subroutine logevent ()
end prototypes

public subroutine newline ();
```

New Source:

```java
public partial class WEventHandling : Synchrony.Forms.SpclForm {

public string ls_crif = "\r\n";

forward prototypes
public subroutine newline ()
public subroutine clear ()
public subroutine log (string logtext)
public subroutine logevent ()
end prototypes

public subroutine newline ();
```

Variables:

<table>
<thead>
<tr>
<th>Type</th>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>self</td>
<td>continuation</td>
<td>a PBForwardPrototypesNode(forward prototypes public subroutine newline () public subroutine clear () public subroutine log (string logtext) public subroutine logevent ())</td>
</tr>
<tr>
<td>match</td>
<td>nodes</td>
<td>a Dictionary [5 items] ['endPrototypesToken' =&gt; (prototypes(870,879,[14,116]))] 'endToken' =&gt; (end(866,868,[39,116]))] 'forwardToken' =&gt; (forward(722,728,[31,116]))]</td>
</tr>
<tr>
<td></td>
<td>rewriteEngine</td>
<td>a SmaCCRewriteEngine</td>
</tr>
<tr>
<td></td>
<td>strings</td>
<td>a Dictionary [5 items] ['endPrototypesToken' =&gt; (cachedString IfNil: cachedString =&gt; self computeStringFor: value)]</td>
</tr>
</tbody>
</table>
Questions?

http://www.refactoryworkers.com/SmaCC/

Download for Pharo:

Gofer new
  smalltalkhubUser: 'JohnBrant'
  project: 'SmaCC';
  configurationOf: 'SmaCC';
  loadBleedingEdge