Migrating Overture to a different IDE

Peter W. V. Tran-Jørgensen, Tomas Kulik
Contribution

• VDM-Mode for Emacs
  – Syntax highlighting
  – Syntax validation
  – REPL
  – VDM Templates
  – Pretty printing
  – Packages accepted into MELPA
The need

• Eclipse usage is falling
• Pointed at 16th Overture workshop
• Emacs package as a case study
Why Emacs?

• Popular text editor
• Often used as IDE
• Massively extensible
• Use of third party packages
• Less than 400 LoC
Enabling Emacs packages

• **prog-mode**
  - Syntax highlighting and indentation rules

• **comint-mode**
  - REPL (interactive shell)

• **flycheck**
  - Syntax checking framework

• **yasnippet**
  - Template expansion system

• **prettify-symbols-mode**
  - Replacement of ASCII characters with Unicode
Syntax highlighting

```vdm
module Example

exports all
definitions

values

SQUARE = \lambda r : \mathbb{N}_1 \cdot r \times r;

functions

isSorted : seq of \mathbb{R} \rightarrow \mathbb{B}

isSorted (xs) ≜
  \forall i, j \in \text{inds} xs \cdot i < j \Rightarrow xs(i) \leq xs(j);

map_[@A,[@B]] : (@A \rightarrow @B) \times (seq of @A) \rightarrow (seq of @A)

map_([fun, xs]) ≜
  if xs = [] then
    []
  else
    [SQUARE(hd xs)] ≜ map_[@A,[@B]](fun, tl xs)

measure len xs;

end Example
```

Peter W. V. Tran-Jørgensen, Tomas Kulik
Syntax highlighting

• **vdm-mode derived from prog-mode**
  - Keyword, type, value highlighting
  - Pretty printing of characters
  - Characters saved as ASCII representation
  - Detection of VDM dialect

• **Comparable to Overture**
Syntax validation

module Example

exports
  functions
  fun : () → B

definitions
values

x = 42;

functions

fun : () → B
fun () ≝ 1 + 2;

end Example

~/D/t/vdm-mode-demo/example.vdmsl 14:0 All LF UTF-8 VDM mode

3018: Function returns unexpected type in 'Example'

Peter W. V. Tran-Jørgensen, Tomas Kulik
Syntax validation

• Based on flycheck
  - Errors
  - Warnings
  - In margin and echo area
  - Interactive list of errors
  - Errors as pop ups

Peter W. V. Tran-Jørgensen, Tomas Kulik
The REPL

```
> reload
Parsed 1 module in 0.005 secs. No syntax errors
Type checked 1 module in 0.013 secs. No type errors
Initialized 1 module in 0.008 secs.
Interpreter started
> p isSorted([5, 10, 99, 97, 101, 999])
= false
Executed in 0.009 secs.
> p map_=[[N_1], [N_1]](SQUARE, [1, 2, 3, 4, 5, 6])
= [1, 4, 9, 16, 25, 36]
Executed in 0.015 secs.
> 
```
The REPL

- *vdm-comint based on comint*
  - Run VDM in REPL-like fashion
  - Send parts of the model directly to the REPL
  - Load files associated with current VDM project
VDM templates

```
functions

myFunction : argTypes → resType
myFunction (argNames) △ ;
```

Flycheck mode disabled in current buffer
VDM templates

• vdm-snippets based on yasnippet
  – Generation of skeleton of VDM syntax
  – Function template expansion
  – Operation template expansion
  – Module template expansion
  – Class template expansion

• Generate and fill in placeholders
What did it take?

• Surprisingly little:

<table>
<thead>
<tr>
<th>File</th>
<th>Blank</th>
<th>Comment</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>vdm-mode.el</td>
<td>38</td>
<td>61</td>
<td>145</td>
</tr>
<tr>
<td>vdm-comint.el</td>
<td>30</td>
<td>35</td>
<td>102</td>
</tr>
<tr>
<td>vdm-mode-util.el</td>
<td>18</td>
<td>23</td>
<td>57</td>
</tr>
<tr>
<td>flycheck-vdm.el</td>
<td>15</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>vdm-snippets.el</td>
<td>14</td>
<td>25</td>
<td>13</td>
</tr>
<tr>
<td>SUM</td>
<td>115</td>
<td>176</td>
<td>349</td>
</tr>
</tbody>
</table>
How does it compare?

<table>
<thead>
<tr>
<th>Features</th>
<th>Overture</th>
<th>Emacs packages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax highlighting</td>
<td>✓</td>
<td>vdm-mode</td>
</tr>
<tr>
<td>Symbol prettyfication</td>
<td>÷</td>
<td>vdm-mode</td>
</tr>
<tr>
<td>Syntax validation</td>
<td>✓</td>
<td>flycheck-vdm</td>
</tr>
<tr>
<td>Evaluation</td>
<td>✓</td>
<td>vdm-comint</td>
</tr>
<tr>
<td>Debugging</td>
<td>✓</td>
<td>vdm-comint</td>
</tr>
<tr>
<td>POG</td>
<td>✓</td>
<td>vdm-comint</td>
</tr>
<tr>
<td>LaTeX report generation</td>
<td>✓</td>
<td>vdm-comint</td>
</tr>
<tr>
<td>Combinatorial testing</td>
<td>✓</td>
<td>vdm-comint</td>
</tr>
<tr>
<td>Code generation</td>
<td>✓</td>
<td>÷</td>
</tr>
<tr>
<td>Auto completion (limited)</td>
<td>✓</td>
<td>company-mode (simple, non-semantic)</td>
</tr>
<tr>
<td>Template expansion</td>
<td>✓</td>
<td>vdm-snippets</td>
</tr>
<tr>
<td>Standard library import</td>
<td>✓</td>
<td>÷ (must be added manually)</td>
</tr>
</tbody>
</table>
Conclusion and future plans

• Basic VDM features in few LoC
  – Syntax highlighting (145)
  – Syntax validation (32)
  – REPL (102)
  – Template expansion (13)

• Much more work required to reach full IDE

• Language Server Protocol (LSP)
  – Increase uptake of Overture
Where to find?

- Github:
  - https://github.com/peterwvj/vdm-mode