LUÍS GONZAGA MARTINS FERREIRA

FORMALIZING MARKUP LANGUAGES FOR USER INTERFACE

Dissertação para Mestrado em Informática

TECHNICAL REPORT

Escola de Engenharia

UNIVERSIDADE DO MINHO

Braga, 2005

Contents

List of Figures

1 Overview

This document constitutes a complementary technical report for Master Thesis *Formalizing Markup Languages for User Interface*.

In Chapter 6 — Prototype and Supporting Tools — of the main document, was slightly described all supporting tools involved on the four phases of the process: *transcoding, abstraction, validation and rendering*. It presented the *uiml2vdm.xsl* stylesheet, responsible for the *UIML* formal representation (in *VDM-SL*); it described the pretty print *VDM-SL* tool *vdm2uiml.vdm*, which allows the generation of *UIML* from *VDM-SL*; it presented the verifier, which allows to test the coherency of the generated *VDM-SL* code along the transcoding process, and finally, presented some scripts involved on the *UIML* rendering process.

This document aims to present the main features of Prototype and to complement the presentation of achieved tools, their source code and application.

The first part describes the *VDM-SL* module responsible to generate *UIML* syntax for each *VDM-SL* specified element. It was presented as the *Pretty Print* tool working on second phase of formalizing process (Chapter 1 - page 10 of main document).

The second part describes the Prototype (which animates OLAP implemented *table* features), presents its source code and images of their main results, and finally describes the settings and requirements to use it.

2 *VDM-SL* to *UIML*

```
module VDM2UIML
    imports
        from IO all ,
        from UIMLSpec all
    exports all
definitions
```

```
values
    PI : UIMLSpec‘String = "    <?xmlversion    =    ’1.0’encoding    =
’ISO-8859-1’?> ";
    end-doc : UIMLSpec‘String = "</uiml> ";
    end-doc-html : UIMLSpec‘String = "  <  peers  ><  presentationhow  =
’replace’source    =    ’HTML3.2Harmonia1.0.uiml#vocab’base    =
’HTML3.2Harmonia1.0’/>\n</peers></uiml> "
```

2.1 VDM to *UIML* Functions

2.1.1 Function $uiml2str$

Specification:

```
uiml2str : UIMLSpec`Uiml → UIMLSpec`String
uiml2str (ui) ≜
  PI ↷
  " < uiml > " ↷
  head2str (ui.head) ↷
  members2str (ui.members) ↷
  end-doc-html;
```

Description:

Converts a UIML *element into String.*

Calls:

$head2str$,$members2str$

2.1.2 Function $member2str$

Specification:

```
member2str : [UIMLSpec`Member] → UIMLSpec`String
member2str (m) ≜
  if m = nil
  then ""
  else cases m :
        mk-UIMLSpec`Peers (-,-,-,-,-) → peer2str (m),
        mk-UIMLSpec`Interface (-,-,-,-,-) → interf2str (m),
        mk-UIMLSpec`Template (-,-) → templ2str (m),
        others → ""
       end;
```

Description:

Converts a Member type into String.

Calls:

$peer2str$,$interf2str$,$templ2str$

2.1.3 Function $members2str$

Specification:

```
members2str : [UIMLSpec`Member*] → UIMLSpec`String
members2str (s) △
  if s = []
  then ""
  else let x = hd (s) in
       member2str (x) ↷ members2str (tl (s));
```

Description:

Converts a Member type sequence into String.

Calls:

$member2str$

2.1.4 Function $head2str$

Specification:

```
head2str : [UIMLSpec`Head] → UIMLSpec`String
head2str (h) △
  if h = nil
  then ""
  else "< head >" ↷
       metas2str (h.meta) ↷
       "</head>";
```

Description:

Converts a Head element into String.

Calls:

$metas2str$

2.1.5 Function $metas2str$

Specification:

```
metas2str : UIMLSpec`Meta* → UIMLSpec`String
metas2str (ms) △
  if ms = []
  then ""
  else let x = hd (ms) in
       meta2str (x) ↷ metas2str (tl (ms));
```

Description:

Converts a Meta elements sequence into String.

Calls:

$meta2str$

2.1.6 Function $meta2str$

Specification:

$$
\begin{array}{l}
meta2str : [UIMLSpec`Meta] \to UIMLSpec`String \\
meta2str\,(m) \triangleq \\
\quad \textsf{if } m = \textsf{nil} \\
\quad \textsf{then } \text{""} \\
\quad \textsf{else } \text{"} < metaname = \text{'"} \curvearrowright m.name \curvearrowright \text{"'} content = \text{'"} \curvearrowright m.content \curvearrowright \\
\qquad \text{"'} / > \text{";}
\end{array}
$$

Description:

Converts a Meta element into String.

Calls:

Standard VDM-SL only

2.1.7 Function $iterator2str$

Specification:

$$
\begin{array}{l}
iterator2str : [UIMLSpec`Iterator] \to UIMLSpec`String \\
iterator2str\,(i) \triangleq \\
\quad \textsf{if } i = \textsf{nil} \\
\quad \textsf{then } \text{""} \\
\quad \textsf{else } \text{"} < iterator \text{"} \curvearrowright id2str\,(i.id) \curvearrowright \text{"} > \text{"} \curvearrowright \\
\qquad iteratorOptions2str\,(i.iterator) \curvearrowright \\
\qquad \text{"} < /iterator > \text{";}
\end{array}
$$

Description:

Converts an Iterator element into String.

Calls:

$id2str$,$iteratorOptions2str$

2.1.8 Function $iteratorOptions2str$

Specification:

```
iteratorOptions2str : [UIMLSpec`IteratorOptions] → UIMLSpec`String
iteratorOptions2str (i) △
  if i = nil
  then ""
  else cases i :
         mk-UIMLSpec`Constant (-,-,-,-,-,-,-) → constant2str (i),
         mk-UIMLSpec`Property (-,-,-,-,-,-,-,-,-) → property2str (i),
         mk-UIMLSpec`Call (-,-) → call2str (i),
         [i] → [i],
         others → ""
       end;
```

Description:

Converts an IteratorOptions element into String.

Calls:

$constant2str$,$property2str$,$call2str$

2.1.9 Function $repeats2str$

Specification:

```
repeats2str : UIMLSpec`Repeat* → UIMLSpec`String
repeats2str (sr) △
  if sr = []
  then ""
  else let x = hd (sr) in
       repeat2str (x) ⌢ repeats2str (tl (sr));
```

Description:

Converts a sequence of Repeat elements into String.

Calls:

$repeat2str$

2.1.10 Function $repeat2str$

Specification:

```
repeat2str : [UIMLSpec`Repeat] → UIMLSpec`String
repeat2str (r) △
  if r = nil
  then ""
  else "< repeat > " ↷
       iterator2str (r.iterator) ↷
       parts2str (r.parts) ↷
       "< /repeat > ";
```

Description:

Converts a Repeat element into String.

Calls:

$iterator2str$,$parts2str$

2.2 Functions for Peer Components

2.2.1 Function $peer2str$

Specification:

```
peer2str : [UIMLSpec`Peers] → UIMLSpec`String
peer2str (p) △
  if p = nil
  then ""
  else "< peers" ↷ id2str (p.id) ↷
       source2str (p.source) ↷
       how2str (p.how) ↷
       export2str (p.export) ↷ " > " ↷
       prelogs2str (p.prelog) ↷
       "< /peers > ";
```

Description:

Converts a Peers element into String.

Calls:

$prelogs2str$,$id2str$,$source2str$,$how2str$,$export2str$

2.2.2 Function $prelog2str$

Specification:

```
prelog2str : (UIMLSpec`Presentation | UIMLSpec`Logic) → UIMLSpec`String
prelog2str (e) ≜
  cases e :
    mk-UIMLSpec`Presentation (-,-,-,-,-,-) → presentation2str (e),
    mk-UIMLSpec`Logic (-,-,-,-,-) → logic2str (e),
    others → ""
  end;
```

Description:

Converts a EqualCPR into String.

Calls:

$presentation2str$,$logic2str$

2.2.3 Function $prelogs2str$

Specification:

```
prelogs2str : (UIMLSpec`Presentation | UIMLSpec`Logic)* → UIMLSpec`String
prelogs2str (s) ≜
  if s = []
  then ""
  else let x = hd (s) in
       prelog2str (x) ⌢ prelogs2str (tl (s));
```

Description:

Converts a (Presentation — Logic) sequence into String.

Calls:

$prelog2str$

2.2.4 Function $presentation2str$

Specification:

```
presentation2str : UIMLSpec`Presentation → UIMLSpec`String
presentation2str (p) ≜
  " < presentation" ↷
  id2str (p.id) ↷
  source2str (p.source) ↷
  how2str (p.how) ↷
  export2str (p.export) ↷
  base2str (p.base) ↷
  " > " ↷
  dclasss2str (p.dclass) ↷
  " < /presentation > ";
```

Description:

Converts a Presentation element into String.

Calls:

$base2str$,$id2str$,$source2str$,$how2str$,$export2str$,$dclasss2str$

2.2.5 Function $dclass2str$

Specification:

```
dclass2str : [UIMLSpec`D-class] → UIMLSpec`String
dclass2str (d) ≜
  if d = nil
  then ""
  else " < d-class" ↷ id2str (d.id) ↷ source2str (d.source) ↷
       how2str (d.how) ↷
       export2str (d.export) ↷
       att2str ("maps-to", d.maps-to) ↷
       mapstype2str (d.maps-type) ↷
       used-in-tag2str (d.used-in-tag) ↷
       " > " ↷
       dmethods2str (d.dmethod) ↷
       dproperties2str (d.dproperty) ↷
       events2str (d.event) ↷
       listeners2str (d.listener) ↷
       " < /d-class > ";
```

Description:

Converts a D-class element into String.

Calls:

Standard VDM-SL only

2.2.6 Function $dclasss2str$

Specification:

$$
\begin{array}{l}
dclasss2str : UIMLSpec`D\text{-}class^* \to UIMLSpec`String \\
dclasss2str\,(s) \triangleq \\
\quad \textsf{if } s = [] \\
\quad \textsf{then } "" \\
\quad \textsf{else let } x = \textsf{hd } (s) \textsf{ in} \\
\qquad dclass2str\,(x) \curvearrowright dclasss2str\,(\textsf{tl } (s));
\end{array}
$$

Description:

Converts a D-class sequence into String.

Calls:

$part2str$

2.2.7 Function $logic2str$

Specification:

$$
\begin{array}{l}
logic2str : [UIMLSpec`Logic] \to UIMLSpec`String \\
logic2str\,(l) \triangleq \\
\quad \textsf{if } l = \textsf{nil} \\
\quad \textsf{then } "" \\
\quad \textsf{else } "< logic" \curvearrowright \\
\qquad id2str\,(l.id) \curvearrowright \\
\qquad source2str\,(l.source) \curvearrowright how2str\,(l.how) \curvearrowright \\
\qquad export2str\,(l.export) \curvearrowright \\
\qquad "/>" \curvearrowright \\
\qquad dcomponents2str\,(l.dcomponent) \curvearrowright \\
\qquad "</logic>";
\end{array}
$$

Description:

Converts a Logic element into String.

Calls:

Standard VDM-SL only

2.2.8 Function $script2str$

Specification:

$$
\begin{array}{l}
script2str : [UIMLSpec`Script] \to UIMLSpec`String \\
script2str\,(s) \triangleq \\
\quad \text{if } s = \text{nil} \\
\quad \text{then } "" \\
\quad \text{else } " < script" \curvearrowright \\
\qquad id2str\,(s.id) \curvearrowright \\
\qquad source2str\,(s.source) \curvearrowright how2str\,(s.how) \curvearrowright \\
\qquad export2str\,(s.export) \curvearrowright \\
\qquad att2str\,("type", s.type) \curvearrowright \\
\qquad " / > " \curvearrowright \\
\qquad s.data \curvearrowright \\
\qquad " < /script > ";
\end{array}
$$

Description:

Converts a Script element into String.

Calls:

Standard VDM-SL only

2.3 Functions for Interface Components

2.3.1 Function $interf2str$

Specification:

$$
\begin{array}{l}
interf2str : [UIMLSpec`Interface] \to UIMLSpec`String \\
interf2str\,(i) \triangleq \\
\quad \text{if } i = \text{nil} \\
\quad \text{then } "" \\
\quad \text{else } " < interface" \curvearrowright id2str\,(i.id) \curvearrowright \\
\qquad source2str\,(i.source) \curvearrowright \\
\qquad how2str\,(i.how) \curvearrowright \\
\qquad export2str\,(i.export) \curvearrowright \\
\qquad " > " \curvearrowright \\
\qquad inteles2str\,(i.intele) \curvearrowright \\
\qquad " < /interface > ";
\end{array}
$$

Description:

Converts a Interface element into String.

Calls:

Standard VDM-SL only

2.3.2 Function $inteles2str$

Specification:

$$
\begin{array}{l}
inteles2str : UIMLSpec`InterfaceElements^* \to UIMLSpec`String \\
inteles2str\,(s) \triangleq \\
\quad \textsf{if } s = [] \\
\quad \textsf{then } "" \\
\quad \textsf{else let } x = \textsf{hd } (s) \textsf{ in} \\
\qquad \textsf{cases } x : \\
\qquad\quad \textsf{mk-}UIMLSpec`Structure\,(\textsf{-},\textsf{-},\textsf{-},\textsf{-},\textsf{-}) \to stru2str\,(x) \curvearrowright \\
inteles2str\,(\textsf{tl } (s)), \\
\qquad\quad \textsf{mk-}UIMLSpec`Style\,(\textsf{-},\textsf{-},\textsf{-},\textsf{-},\textsf{-}) \to style2str\,(x) \curvearrowright \\
inteles2str\,(\textsf{tl } (s)), \\
\qquad\quad \textsf{mk-}UIMLSpec`Content\,(\textsf{-},\textsf{-},\textsf{-},\textsf{-},\textsf{-}) \to content2str\,(x) \curvearrowright \\
inteles2str\,(\textsf{tl } (s)), \\
\qquad\quad \textsf{mk-}UIMLSpec`Behavior\,(\textsf{-},\textsf{-},\textsf{-},\textsf{-},\textsf{-}) \to behav2str\,(x) \curvearrowright \\
inteles2str\,(\textsf{tl } (s)) \\
\qquad \textsf{end};
\end{array}
$$

Description:

Converts an InterfaceElements set into String.

Calls:

$stru2str$,$style2str$,$content2str$,$behav2str$

2.3.3 Function $stru2str$

Specification:

$$
\begin{array}{l}
stru2str : UIMLSpec`Structure \to UIMLSpec`String \\
stru2str\,(s) \triangleq \\
\quad "\ < \ structure" \curvearrowright id2str\,(s.id) \curvearrowright source2str\,(s.source) \curvearrowright \\
how2str\,(s.how) \curvearrowright export2str\,(s.export) \curvearrowright " > " \curvearrowright \\
\quad parts2str\,(s.parts) \curvearrowright \\
\quad " < /structure > ";
\end{array}
$$

Description:

Converts a Structure element into String.

Calls:

$srcatt2str$,$parts2str$

2.3.4 Function $parts2str$

Specification:

$$
\begin{array}{l}
parts2str : UIMLSpec`Part^* \to UIMLSpec`String \\
parts2str\,(s) \triangleq \\
\quad \textsf{if } s = [] \\
\quad \textsf{then } "" \\
\quad \textsf{else let } x = \textsf{hd } (s) \textsf{ in} \\
\qquad part2str\,(x) \curvearrowright parts2str\,(\textsf{tl } (s));
\end{array}
$$

Description:

Converts a Part sequence into String.

Calls:

$part2str$

2.3.5 Function $part2str$

Specification:

$$
\begin{array}{l}
part2str : [UIMLSpec`Part] \to UIMLSpec`String \\
part2str\,(p) \triangleq \\
\quad \textsf{if } p = \textsf{nil} \\
\quad \textsf{then } "" \\
\quad \textsf{else } "<part" \curvearrowright id2str\,(p.id) \curvearrowright \\
\qquad source2str\,(p.source) \curvearrowright \\
\qquad how2str\,(p.how) \curvearrowright \\
\qquad export2str\,(p.export) \curvearrowright \\
\qquad class2str\,(p.class) \curvearrowright \\
\qquad where2str\,(p.where) \curvearrowright \\
\qquad wherepart2str\,(p.where\text{-}part) \curvearrowright \\
\qquad ">" \curvearrowright \\
\qquad style2str\,(p.style) \curvearrowright \\
\qquad content2str\,(p.content) \curvearrowright \\
\qquad behav2str\,(p.behavior) \curvearrowright \\
\qquad parts2str\,(p.parts) \curvearrowright \\
\qquad repeats2str\,(p.repeats) \curvearrowright \\
\qquad "</part>";
\end{array}
$$

Description:

Converts a Part element into String.

Calls:

$srcatt2str$,$classs2str$,$style2str$,$content2str$,$behav2str$,$parts2str$

2.3.6 Function $class2str$

Specification:

$class2str : [UIMLSpec`String] \to UIMLSpec`String$
$class2str\,(c) \triangleq$
  if $c =$ nil
  then ""
  else "$class =$ '" $\curvearrowright$ c $\curvearrowright$ "'";

Description:

Converts a class attribute into String.

Calls:

$class2str$

2.3.7 Function $properties2str$

Specification:

$properties2str : UIMLSpec`Property^* \to UIMLSpec`String$
$properties2str\,(s) \triangleq$
  if $s = []$
  then ""
  else let $x =$ hd (s) in
    $property2str\,(x) \curvearrowright properties2str\,(\text{tl}\ (s))$;

Description:

Converts a Property set into String.

Calls:

$property2str$

2.3.8 Function $property2str$

Specification:

```
property2str : [UIMLSpec`Property] → UIMLSpec`String
property2str (p) ≜
  if p = nil
  then ""
  else "< property" ↷
       name2str (p.name) ↷
       source2str (p.source) ↷
       how2str (p.how) ↷
       export2str (p.export) ↷
       att2str ("part-name", p.p-name) ↷
       att2str ("part-class", p.p-class) ↷
       att2str ("event-name", p.e-name) ↷
       att2str ("event-class", p.e-class) ↷
       "> " ↷
       proptypes2str (p.property) ↷
       "</property> "
```

Description:

Converts a Property element into String.

Calls:

$proptypes2str$

2.3.9 Function $proptypes2str$

Specification:

```
PropertyTypes = UIMLSpec`String | UIMLSpec`Constant |
UIMLSpec`Property |
                UIMLSpec`Reference | UIMLSpec`Call |
UIMLSpec`Iterator
proptypes2str : PropertyTypes* → UIMLSpec`String
proptypes2str (s) ≜
  if s = []
  then ""
  else let x = hd (s) in
       proptype2str (x) ↷ proptypes2str (tl (s));
```

Description:

Converts a Property type sequence into String.

Calls:

$property2str$

2.3.10 Function $proptype2str$

Specification:

$$
\begin{array}{l}
proptype2str : [PropertyTypes] \to UIMLSpec`String \\
proptype2str\,(p) \triangleq \\
\quad \text{if } p = \text{nil} \\
\quad \text{then ""} \\
\quad \text{else cases } p : \\
\qquad \text{mk-}UIMLSpec`Constant\,(\text{-},\text{-},\text{-},\text{-},\text{-},\text{-},\text{-}) \to constant2str\,(p), \\
\qquad \text{mk-}UIMLSpec`Property\,(\text{-},\text{-},\text{-},\text{-},\text{-},\text{-},\text{-},\text{-},\text{-}) \to property2str\,(p), \\
\qquad \text{mk-}UIMLSpec`Reference\,(\text{-},\text{-}) \to refer2str\,(p), \\
\qquad \text{mk-}UIMLSpec`Call\,(\text{-},\text{-}) \to call2str\,(p), \\
\qquad \text{mk-}UIMLSpec`Iterator\,(\text{-},\text{-}) \to iterator2str\,(p), \\
\qquad \text{others} \to p \\
\quad \text{end};
\end{array}
$$

Description:

Converts a Property type into String.

Calls:

$constant2str$,$property2str$,$refer2str$

2.3.11 Function $style2str$

Specification:

$$
\begin{array}{l}
style2str : [UIMLSpec`Style] \to UIMLSpec`String \\
style2str\,(s) \triangleq \\
\quad \text{if } s = \text{nil} \\
\quad \text{then ""} \\
\quad \text{else "} < style \text{"} \curvearrowright \\
\qquad id2str\,(s.id) \curvearrowright \\
\qquad source2str\,(s.source) \curvearrowright \\
\qquad how2str\,(s.how) \curvearrowright \\
\qquad export2str\,(s.export) \curvearrowright \\
\qquad \text{" > "} \curvearrowright \\
\qquad properties2str\,(s.property) \curvearrowright \\
\qquad \text{" } < /style > \text{";}
\end{array}
$$

Description:

Converts a Style element into String.

Calls:

$srcatt2str$,$properties2str$

2.3.12 Function $content2str$

Specification:

$$
\begin{array}{l}
content2str : [UIMLSpec`Content] \to UIMLSpec`String \\
content2str\,(c) \triangleq \\
\quad \textsf{if } c = \textsf{nil} \\
\quad \textsf{then } "" \\
\quad \textsf{else } " < content" \curvearrowright id2str\,(c.id) \curvearrowright source2str\,(c.source) \curvearrowright \\
\qquad how2str\,(c.how) \curvearrowright \\
\qquad export2str\,(c.export) \curvearrowright " > " \curvearrowright \\
\qquad constants2str\,(c.constant) \curvearrowright \\
\qquad " < /content > ";
\end{array}
$$

Description:

Converts a Content element into String.

Calls:

$srcatt2str$,$constants2str$

2.3.13 Function $constants2str$

Specification:

$$
\begin{array}{l}
constants2str : UIMLSpec`Constant^* \to UIMLSpec`String \\
constants2str\,(s) \triangleq \\
\quad \textsf{if } s = [] \\
\quad \textsf{then } "" \\
\quad \textsf{else let } x = \textsf{hd } (s) \textsf{ in} \\
\qquad constant2str\,(x) \curvearrowright constants2str\,(\textsf{tl } (s));
\end{array}
$$

Description:

Converts a Constant sequence into String.

Calls:

$constant2str$

2.3.14 Function $constant2str$

Specification:

```
constant2str : [UIMLSpec`Constant] → UIMLSpec`String
constant2str (c) △
  if c = nil
  then ""
  else "< constant" ↷
       id2str (c.id) ↷
       source2str (c.source) ↷
       how2str (c.how) ↷
       export2str (c.export) ↷
       model2str (c.model) ↷
       value2str (c.value) ↷
       "> " ↷
       constants2str (c.constant) ↷
       "</constant>";
```

Description:

Converts a Constant element into String.

Calls:

$srcatt2str$,$constant2str$

2.3.15 Function $refer2str$

Specification:

```
refer2str : [UIMLSpec`Reference] → UIMLSpec`String
refer2str (r) △
  if r = nil
  then ""
  else "<reference" ↷ constantname2str (r.constant-name) ↷
urlname2str (r.url-name) ↷ "/>";
```

Description:

Converts a Reference into String.

Calls:

Standard VDM-SL only

2.3.16 Function $behav2str$

Specification:

```
behav2str : [UIMLSpec`Behavior] → UIMLSpec`String
behav2str (b) △
  if b = nil
  then ""
  else "<behavior" ↷ id2str (b.id) ↷ source2str (b.source) ↷
       how2str (b.how) ↷
       export2str (b.export) ↷ ">" ↷
       rules2str (b.rules) ↷
       "</behavior>";
```

Description:

Converts a Behavior element into String.

Calls:

$srcatt2str$,$rules2str$

2.3.17 Function $rules2str$

Specification:

```
rules2str : UIMLSpec`Rule* → UIMLSpec`String
rules2str (s) △
  if s = []
  then ""
  else let x = hd (s) in
       rule2str (x) ↷ rules2str (tl (s));
```

Description:

Converts a Rule set into String.

Calls:

$rule2str$

2.3.18 Function $rule2str$

Specification:

```
rule2str : [UIMLSpec`Rule] → UIMLSpec`String
rule2str (r) ≜
  if r = nil
  then ""
  else "<rule" ↷ id2str (r.id) ↷ source2str (r.source) ↷
       how2str (r.how) ↷
       export2str (r.export) ↷ ">" ↷
       condition2str (r.condition) ↷
       action2str (r.action) ↷
       "</rule>";
```

Description:

Converts a Rule into String.

Calls:

$rule2str$

2.3.19 Function $action2str$

Specification:

```
action2str : UIMLSpec`Action → UIMLSpec`String
action2str (a) ≜
  "<action>" ↷
  cases a :
    mk-UIMLSpec`ActionType1 (-,-) → acttypes2str (a.type) ↷
event2str (a.event),
    mk-UIMLSpec`ActionType2 (-,-,-) →
      whentrue2str (a.whentrue) ↷
      whenfalse2str (a.whenfalse) ↷
      bydefault2str (a.bydefault),
    others → ""
  end ↷
  "</action>"
```

Description:

Converts an Action into String.

Calls:

$acttypes2str$,$event2str$

2.3.20 Function $acttypes2str$

Specification:

```
ActionType = UIMLSpec`Property | UIMLSpec`Call | UIMLSpec`Restructure
acttypes2str : ActionType* → UIMLSpec`String
acttypes2str (s) △
  if s = []
  then ""
  else let x = hd (s) in
       acttype2str (x) ⌢ acttypes2str (tl (s));
```

Description:

Converts an Action type sequence into String.

Calls:

$acttype2str$

2.3.21 Function $acttype2str$

Specification:

```
acttype2str : ActionType → UIMLSpec`String
acttype2str (s) △
  cases s :
    mk-UIMLSpec`Property (-,-,-,-,-,-,-,-,-) → property2str (s),
    mk-UIMLSpec`Call (-,-) → call2str (s),
    mk-UIMLSpec`Restructure (-,-,-,-,-,-) → restructure2str (s),
    others → ""
  end;
```

Description:

Converts a Action type set into String.

Calls:

$acttype2str$

2.3.22 Function $events2str$

Specification:

```
events2str : UIMLSpec`Event* → UIMLSpec`String
events2str (s) ≜
  if s = []
  then ""
  else let x = hd (s) in
       event2str (x) ↷ events2str (tl (s));
```

Description:

Converts an Event sequence into String.

Calls:

$event2str$

2.3.23 Function $event2str$

Specification:

```
event2str : [UIMLSpec`Event] → UIMLSpec`String
event2str (e) ≜
  if e = nil
  then ""
  else "< event" ↷
       att2str ("name", e.name) ↷
       att2str ("class", e.class) ↷
       att2str ("part-name", e.p-name) ↷
       att2str ("part-class", e.p-class) ↷
       "/>";
```

Description:

Converts a Event into String.

Calls:

Standard VDM-SL only

2.3.24 Function $condition2str$

Specification:

```
condition2str : [UIMLSpec`Condition] → UIMLSpec`String
condition2str (c) △
  if c = nil
  then ""
  else cases c.type :
        mk-UIMLSpec`Equal (-,-) → "  <  condition  >  " ↷
equal2str (c.type) ↷ " < /condition > ",
        mk-UIMLSpec`Event (-,-,-,-) → "  <  condition  >
" ↷ event2str (c.type) ↷
                                    " < /condition > "
      end;
```

Description:

Converts a Condition into String.

Calls:

$equal2str$,$event2str$

2.3.25 Function $equal2str$

Specification:

```
equal2str : [UIMLSpec`Equal] → UIMLSpec`String
equal2str (e) △
  if e = nil
  then ""
  else " < equal > " ↷
       event2str (e.event) ↷
       cpro2str (e.other) ↷
       " < /equal > "
```

Description:

Converts a Equal into String.

Calls:

$event2str$,$cpr2str$

2.3.26 Function $cpro2str$

Specification:

```
EqualCPRO = UIMLSpec`Constant | UIMLSpec`Property |
UIMLSpec`Reference | UIMLSpec`Op
cpro2str : EqualCPRO → UIMLSpec`String
cpro2str (e) ≜
  cases e :
    mk-UIMLSpec`Constant (-,-,-,-,-,-,-) → constant2str (e),
    mk-UIMLSpec`Property (-,-,-,-,-,-,-,-,-) → property2str (e),
    mk-UIMLSpec`Reference (-,-) → refer2str (e),
    mk-UIMLSpec`Op (-,-) → op2str (e),
    others → ""
  end;
```

Description:

Converts a EqualCPR into String.

Calls:

$constant2str$,$property2str$,$refer2str$

2.3.27 Function $call2str$

Specification:

```
call2str : [UIMLSpec`Call] → UIMLSpec`String
call2str (c) ≜
  if c = nil
  then ""
  else "<call" ↷
       att2str ("name", c.name) ↷
       "/>" ↷
       params2str (c.params) ↷
       "</call>";
```

Description:

Converts a Call element into String.

Calls:

Standard VDM-SL only

2.3.28 Function $params2str$

Specification:

```
params2str : UIMLSpec`Param* → UIMLSpec`String
params2str (s) △
  if s = []
  then ""
  else let x = hd (s) in
       param2str (x) ⌢ params2str (tl (s));
```

Description:

Converts a Params type sequence into String.

Calls:

$param2str$

2.3.29 Function $param2str$

Specification:

```
param2str : [UIMLSpec`Param] → UIMLSpec`String
param2str (s) △
  if s = nil
  then ""
  else cases s.type :
         mk-UIMLSpec`Constant (-,-,-,-,-,-,-) → constant2str (s.type),
         mk-UIMLSpec`Property (-,-,-,-,-,-,-,-,-) → property2str (s.type),
         mk-UIMLSpec`Reference (-,-) → refer2str (s.type),
         mk-UIMLSpec`Op (-,-) → op2str (s.type),
         mk-UIMLSpec`Event (-,-,-,-) → event2str (s.type),
         mk-UIMLSpec`Call (-,-) → call2str (s.type),
         mk-UIMLSpec`Iterator (-,-) → iterator2str (s.type),
         others → s.type
       end;
```

Description:

Converts a Param element into String.

Calls:

Standard VDM-SL only

2.4 Functions for Reusable Interface Components

2.4.1 Function $dmethods2str$

Specification:

```
dmethods2str : UIMLSpec`D-method* → UIMLSpec`String
dmethods2str (s) △
  if s = []
  then ""
  else let x = hd (s) in
       dmethod2str (x) ↷ dmethods2str (tl (s));
```

Description:

Converts a D-method sequence into String.

Calls:

$dmethod2str$

2.4.2 Function $dmethod2str$

Specification:

```
dmethod2str : [UIMLSpec`D-method] → UIMLSpec`String
dmethod2str (d) △
  if d = nil
  then ""
  else " < d-method > " ↷
       "/ > " ↷
       dparams2str (d.dparam) ↷
       script2str (d.script) ↷
       " < /d-method > ";
```

Description:

Converts a D-method element into String.

Calls:

Standard VDM-SL only

2.4.3 Function $dproperties2str$

Specification:

```
dproperties2str : UIMLSpec`D-property* → UIMLSpec`String
dproperties2str (s) △
  if s = []
  then ""
  else let x = hd (s) in
       dproperty2str (x) ↷ dproperties2str (tl (s));
```

Description:

Converts a D-property set into String.

Calls:

$dproperty2str$

2.4.4 Function $dproperty2str$

Specification:

```
dproperty2str : [UIMLSpec`D-property] → UIMLSpec`String
dproperty2str (d) △
  if d = nil
  then ""
  else "< d-property" ↷
       id2str (d.id) ↷
       att2str ("maps-to", d.maps-to) ↷
       mapstype2str (d.maps-type) ↷
       att2str ("return-type", d.return-type) ↷
       "/ > " ↷
       "< /d-property > ";
```

Description:

Converts a D-property element into String.

Calls:

Standard VDM-SL only

2.4.5 Function $dcomponents2str$

Specification:

```
dcomponents2str : UIMLSpec`D-component* → UIMLSpec`String
dcomponents2str (s) △
  if s = []
  then ""
  else let x = hd (s) in
       dcomponent2str (x) ↷ dcomponents2str (tl (s));
```

Description:

Converts a D-param sequence into String.

Calls:

$dproperty2str$

2.4.6 Function $dcomponent2str$

Specification:

```
dcomponent2str : [UIMLSpec`D-component] → UIMLSpec`String
dcomponent2str (d) △
  if d = nil
  then ""
  else "< d-component" ↷
       id2str (d.id) ↷
       source2str (d.source) ↷ how2str (d.how) ↷
       export2str (d.export) ↷
       att2str ("maps-to", d.maps-to) ↷
       att2str ("location", d.location) ↷
       "/ > " ↷
       dmethods2str (d.dmethod) ↷
       "< /d-component > ";
```

Description:

Converts a D-component element into String.

Calls:

Standard VDM-SL only

2.4.7 Function $dparams2str$

Specification:

```
dparams2str : UIMLSpec`D-param* → UIMLSpec`String
dparams2str (s) △
  if s = []
  then ""
  else let x = hd (s) in
       dparam2str (x) ↷ dparams2str (tl (s));
```

Description:

Converts a D-param sequence into String.

Calls:

$dproperty2str$

2.4.8 Function $dparam2str$

Specification:

```
dparam2str : [UIMLSpec`D-param] → UIMLSpec`String
dparam2str (d) △
  if d = nil
  then ""
  else "< d-param" ↷
       id2str (d.id) ↷
       att2str ("type", d.type) ↷
       "/ > " ↷
       d.data ↷
       "< /d-param > ";
```

Description:

Converts a D-param element into String.

Calls:

Standard VDM-SL only

2.4.9 Function $templ2str$

Specification:

```
templ2str : [UIMLSpec`Template] → UIMLSpec`String
templ2str (t) ≜
  if t = nil
  then ""
  else cases t.src-ele :
        mk-UIMLSpec`Behavior (-,-,-,-,-) → behav2str (t.src-ele),
        mk-UIMLSpec`Structure (-,-,-,-,-) → stru2str (t.src-ele),
        mk-UIMLSpec`Style (-,-,-,-,-) → style2str (t.src-ele),
        mk-UIMLSpec`Content (-,-,-,-,-) → content2str (t.src-ele),
        mk-UIMLSpec`Constant (-,-,-,-,-,-,-) → constant2str (t.src-ele),
        mk-UIMLSpec`Property (-,-,-,-,-,-,-,-,-) → property2str (t.src-ele),
        mk-UIMLSpec`Peers (-,-,-,-,-) → peer2str (t.src-ele),
        mk-UIMLSpec`Presentation (-,-,-,-,-,-) → presentation2str (t.src-ele),
        mk-UIMLSpec`Logic (-,-,-,-,-) → logic2str (t.src-ele),
        mk-UIMLSpec`Part (-,-,-,-,-,-,-,-,-,-,-) → part2str (t.src-ele),
        mk-UIMLSpec`Restructure (-,-,-,-,-,-) → restructure2str (t.src-ele),
        mk-UIMLSpec`Interface (-,-,-,-,-) → interf2str (t.src-ele),
        mk-UIMLSpec`Rule (-,-,-,-,-,-) → rule2str (t.src-ele),
        mk-UIMLSpec`Script (-,-,-,-,-,-) → script2str (t.src-ele),
        mk-UIMLSpec`D-class (-,-,-,-,-,-,-,-,-,-,-) → dclass2str (t.src-ele),
        mk-UIMLSpec`D-component (-,-,-,-,-,-,-) → dcomponent2str (t.src-ele),
        others → ""
      end;
```

Description:

Converts a Template element into String.

Calls:

Standard VDM-SL only

2.4.10 Function $listeners2str$

Specification:

```
listeners2str : UIMLSpec`Listener* → UIMLSpec`String
listeners2str (s) ≜
  if s = []
  then ""
  else let x = hd (s) in
       listener2str (x) ⌢ listeners2str (tl (s));
```

Description:

Converts a Listener sequence into String.

Calls:

$dmethod2str$

2.4.11 Function $listener2str$

Specification:

$$
\begin{array}{l}
listener2str : [UIMLSpec`Listener] \to UIMLSpec`String \\
listener2str\,(l) \triangleq \\
\quad \text{if } l = \text{nil} \\
\quad \text{then ""} \\
\quad \text{else "} < listener\text{"} \curvearrowright \\
\qquad att2str\,(\text{"}class\text{"}, l.class) \curvearrowright \\
\qquad att2str\,(\text{"}attacher\text{"}, l.attacher) \curvearrowright \\
\qquad \text{"}/ > \text{";}
\end{array}
$$

Description:

Converts a Listener element into String.

Calls:

Standard VDM-SL only

2.4.12 Function $whentrue2str$

Specification:

$$
\begin{array}{l}
whentrue2str : [UIMLSpec`When\text{-}true] \to UIMLSpec`String \\
whentrue2str\,(w) \triangleq \\
\quad \text{"} < when\text{-}true\text{"} \curvearrowright whentruetypes2str\,(w.type) \curvearrowright \\
restructure2str\,(w.restructure) \curvearrowright \\
\qquad op2str\,(w.op) \curvearrowright \\
\qquad equal2str\,(w.equal) \curvearrowright event2str\,(w.event) \curvearrowright \\
\qquad \text{"} < /when\text{-}true > \text{";}
\end{array}
$$

Description:

Converts a When-true element into String.

Calls:

$whentruetypes2str$,$restructure2str$,$op2str$,$equal2str$,$event2str$

2.4.13 **Function** $whenfalse2str$

Specification:

$$
\begin{array}{l}
whenfalse2str : [UIMLSpec`When\text{-}false] \to UIMLSpec`String \\
whenfalse2str\,(w) \triangleq \\
\quad \text{"} < when\text{-}false \text{"} \curvearrowright whentruetypes2str\,(w.type) \curvearrowright \\
restructure2str\,(w.restructure) \curvearrowright \\
\quad op2str\,(w.op) \curvearrowright \\
\quad equal2str\,(w.equal) \curvearrowright event2str\,(w.event) \curvearrowright \\
\quad \text{"} < /when\text{-}false > \text{"};
\end{array}
$$

Description:

Converts a When-false element into String.

Calls:

$whentruetypes2str$,$restructure2str$,$op2str$,$equal2str$,$event2str$

2.4.14 **Function** $bydefault2str$

Specification:

$$
\begin{array}{l}
bydefault2str : [UIMLSpec`By\text{-}default] \to UIMLSpec`String \\
bydefault2str\,(w) \triangleq \\
\quad \text{"} < by\text{-}default \text{"} \curvearrowright whentruetypes2str\,(w.type) \curvearrowright \\
restructure2str\,(w.restructure) \curvearrowright \\
\quad op2str\,(w.op) \curvearrowright \\
\quad equal2str\,(w.equal) \curvearrowright event2str\,(w.event) \curvearrowright \\
\quad \text{"} < /by\text{-}default > \text{"};
\end{array}
$$

Description:

Converts a By-default element into String.

Calls:

$whentruetypes2str$,$restructure2str$,$op2str$,$equal2str$,$event2str$

2.4.15 **Function** $restructure2str$

Specification:

```
restructure2str : [UIMLSpec`Restructure] → UIMLSpec`String
restructure2str (r) ≜
  if r = nil
  then ""
  else "<restructure" ↷
       att2str ("at-part", r.at-part) ↷
       how2str (r.how) ↷
       where2str (r.where) ↷
       att2str ("where-part", r.where-part) ↷
       source2str (r.source) ↷
       "/>" ↷
       templ2str (r.template) ↷
       "</restructure>";
```

Description:

Converts a Restructure element into String.

Calls:

Standard VDM-SL only

2.4.16 Function *whentruetype2str*

Specification:

```
whentruetype2str : (UIMLSpec`Property | UIMLSpec`Call)
→ UIMLSpec`String
whentruetype2str (c) ≜
  cases c :
    mk-UIMLSpec`Property (-,-,-,-,-,-,-,-,-) → property2str (c),
    mk-UIMLSpec`Call (-,-) → call2str (c)
  end;
```

Description:

Converts a When-true type element into String.

Calls:

Standard VDM-SL only

2.4.17 Function $whentruetypes2str$

Specification:

```
    whentruetypes2str : (UIMLSpec`Property | UIMLSpec`Call)*
→ UIMLSpec`String
    whentruetypes2str (s) △
      if s = []
      then ""
      else let x = hd (s) in
           whentruetype2str (x) ↷ whentruetypes2str (tl (s));
```

Description:

Converts a When-true type sequence into String.

Calls:

Standard VDM-SL only

2.4.18 Function $op2str$

Specification:

```
    op2str : [UIMLSpec`Op] → UIMLSpec`String
    op2str (o) △
      if o = nil
      then ""
      else "<op>" ↷
           optypes2str (o.type) ↷
           name2str (o.name) ↷
           "</op>";
```

Description:

Converts a Op element into String.

Calls:

Standard VDM-SL only

2.4.19 Function $optype2str$

Specification:

```
optype2str : OpType → UIMLSpec`String
optype2str (o) ≜
  cases o :
    mk-UIMLSpec`Constant (-,-,-,-,-,-,-) → constant2str (o),
    mk-UIMLSpec`Property (-,-,-,-,-,-,-,-,-) → property2str (o),
    mk-UIMLSpec`Reference (-,-) → refer2str (o),
    mk-UIMLSpec`Op (-,-) → op2str (o),
    mk-UIMLSpec`Event (-,-,-,-) → event2str (o),
    mk-UIMLSpec`Call (-,-) → call2str (o),
    others → ""
  end
```

Description:

Converts a Op type element into String.

Calls:

Standard VDM-SL only

2.4.20 Function $optypes2str$

Specification:

```
OpType = UIMLSpec`Property | UIMLSpec`Call |
UIMLSpec`Reference | UIMLSpec`Constant |
        UIMLSpec`Event | UIMLSpec`Op
optypes2str : OpType* → UIMLSpec`String
optypes2str (s) ≜
  if s = []
  then ""
  else let x = hd (s) in
       op2str (x) ↷ optypes2str (tl (s));
```

Description:

Converts an Op type sequence into String.

Calls:

$acttype2str$

2.5 Attributes handling functions

2.5.1 Function $id2str$

Specification:

```
id2str : [UIMLSpec`ID] → UIMLSpec`String
id2str (id) △
  cases id :
    mk-UIMLSpec`ID (i) → "id = '" ↷ i ↷ "'",
    others → ""
  end;
```

Description:

Converts a Id attribute into String.

Calls:

Standard VDM-SL only

2.5.2 Function $name2str$

Specification:

```
name2str : UIMLSpec`String → UIMLSpec`String
name2str (s) △
  if s = ""
  then ""
  else "name = '" ↷ s ↷ "'";
```

Description:

Converts a Name attribute into String.

Calls:

Standard VDM-SL only

2.5.3 Function $base2str$

Specification:

```
base2str : UIMLSpec`String → UIMLSpec`String
base2str (s) △
  if s = ""
  then ""
  else "base = '" ↷ s ↷ "'";
```

Description:

Converts a Base attribute into String.

Calls:

Standard VDM-SL only

2.5.4 Function $source2str$

Specification:

```
source2str : UIMLSpec`String → UIMLSpec`String
source2str (s) △
  if s = ""
  then ""
  else "source = '" ↷ s ↷ "'";
```

Description:

Converts a Source attribute into String.

Calls:

Standard VDM-SL only

2.5.5 Function $how2str$

Specification:

```
how2str : UIMLSpec`SourcesModes → UIMLSpec`String
how2str (s) △
  cases s :
    APPEND → "how = 'append'",
    CASCADE → "how = 'cascade'",
    REPLACE → "how = 'replace'",
    UNION → "how = 'union'",
    others → ""
  end;
```

Description:

Converts a how attribute into String.

Calls:

Standard VDM-SL only

2.5.6 Function $export2str$

Specification:

```
export2str : UIMLSpec`ExportOptions → UIMLSpec`String
export2str (s) △
  cases s :
    HIDDEN → "export = 'hidden'",
    OPTIONAL → "export = 'optional'",
    REQUIRED → "export = 'required'",
    others → ""
  end;
```

Description:

Converts a Export attribute into String.

Calls:

Standard VDM-SL only

2.5.7 Function $where2str$

Specification:

```
where2str : UIMLSpec`WhereOptions → UIMLSpec`String
where2str (s) △
  cases s :
    FIRST → "where = 'first'",
    LAST → "where = 'last'",
    BEFORE → "where = 'before'",
    AFTER → "where = 'after'",
    others → ""
  end;
```

Description:

Converts a Where attribute into String.

Calls:

Standard VDM-SL only

2.5.8 Function $mapstype2str$

Specification:

```
mapstype2str : UIMLSpec`mapsTypes → UIMLSpec`String
mapstype2str (s) △
  cases s :
    ATTRIBUTE → "maps-type = 'attribute'",
    TAG → "maps-type = 'tag'",
    CLASS → "maps-type = 'class'",
    others → ""
  end;
```

Description:

Converts a maps-to *attribute into String.*

Calls:

Standard VDM-SL only

2.5.9 Function $used\_in\_tag2str$

Specification:

```
used-in-tag2str : UIMLSpec`used-in-tagTypes → UIMLSpec`String
used-in-tag2str (s) △
  cases s :
    EVENT → "used-in-tag = 'event'",
    LISTENER → "used-in-tag = 'listener'",
    PART → "used-in-tag = 'part'",
    others → ""
  end;
```

Description:

Converts a used-in-tag *attribute into String.*

Calls:

Standard VDM-SL only

2.5.10 Function $value2str$

Specification:

```
value2str : UIMLSpec`String → UIMLSpec`String
value2str (s) △
  if s = ""
  then ""
  else "value = '" ⌢ s ⌢ "'";
```

Description:

Converts a Value attribute into String.

Calls:

Standard VDM-SL only

2.5.11 Function $model2str$

Specification:

```
model2str : UIMLSpec`String → UIMLSpec`String
model2str (s) △
  if s = ""
  then ""
  else "model = '" ⌢ s ⌢ "'";
```

Description:

Converts a Model attribute into String.

Calls:

Standard VDM-SL only

2.5.12 Function $wherepart2str$

Specification:

```
wherepart2str : UIMLSpec`String → UIMLSpec`String
wherepart2str (s) △
  if s = ""
  then ""
  else "where-part = '" ⌢ s ⌢ "'";
```

Description:

Converts a Where-part attribute into String.

Calls:

Standard VDM-SL only

2.5.13 Function $constantname2str$

Specification:

$constantname2str : UIMLSpec`String \to UIMLSpec`String$
$constantname2str\,(s) \triangleq$
  if $s =$ ""
  then ""
  else "$constant\text{-}name =$ '" $\frown$ s $\frown$ "'";

Description:

Converts a Constant-name attribute into String.

Calls:

Standard VDM-SL only

2.5.14 Function $urlname2str$

Specification:

$urlname2str : UIMLSpec`String \to UIMLSpec`String$
$urlname2str\,(s) \triangleq$
  if $s =$ ""
  then ""
  else "$url\text{-}name =$ '" $\frown$ s $\frown$ "'";

Description:

Converts a Url-name attribute into String.

Calls:

Standard VDM-SL only

2.5.15 Function $att2str$

Specification:

$$
\begin{array}{l}
att2str : UIMLSpec`String \times [UIMLSpec`String] \to UIMLSpec`String \\
att2str\,(e, v) \triangleq \\
\quad \text{if } v = \text{nil} \\
\quad \text{then ""} \\
\quad \text{else if } v = \text{""} \\
\qquad \text{then ""} \\
\qquad \text{else ""} \curvearrowright e \curvearrowright \text{" = '"} \curvearrowright v \curvearrowright \text{"'"};
\end{array}
$$

Description:

Converts a pair (attribute(a),value(v)) a to String a='v'.

Calls:

Standard VDM-SL only

2.5.16 Function $toFileHTML$

Specification:

$$
\begin{array}{l}
toFileHTML : UIMLSpec`Uiml \times UIMLSpec`String \to \mathbb{B} \\
toFileHTML\,(ui, f) \triangleq \\
\quad IO`fecho\,(f, PI, \text{START}) \wedge \\
\quad IO`fecho\,(f, \text{" <!}DOCTYPE\,uiml\,PUBLIC\text{'-//}UIT\text{//}DTD\,UIML\text{"} \curvearrowright \\
\qquad\qquad \text{"2.0}Draft\text{//}EN\text{'}\backslash n\text{"}, \\
\qquad\qquad \text{APPEND}) \wedge \\
\quad IO`fecho\,(f, \text{"'}UIML2_0g.dtd\text{'} > \backslash n\text{"}, \text{APPEND}) \wedge \\
\quad IO`fecho\,(f, \text{" < }uiml\text{ > "}, \text{APPEND}) \wedge \\
\quad IO`fecho\,(f, head2str\,(ui.head), \text{APPEND}) \wedge \\
\quad IO`fecho\,(f, members2str\,(ui.members), \text{APPEND}) \wedge \\
\quad IO`fecho\,(f, end\text{-}doc\text{-}html, \text{APPEND})
\end{array}
$$

Description:

Output an UIML *element to HTML file f.*

Calls:

$head2str$,$members2str$

2.6 Operations

2.6.1 Operation $writeF$

Specification:

$$
\begin{array}{l}
FileName = UIMLSpec`String \\
writeF : FileName \times UIMLSpec`Uiml \xrightarrow{o} \mathbb{B} \\
writeF\,(fn, ui) \triangleq \\
\quad IO`fecho(fn, uiml2str\,(ui), \text{START})
\end{array}
$$

Description:

Writes the String generated from the UIML *element in file fn.*

Calls:

$fecho$,$uiml2str$

---

3 Tests

VDM-SL methods call in expression:

$VDM2UIML`toFileHTML(UIMLSpecTab`TU2U(UIMLSpecTab`t)$, "t.uiml")

Test Suite : UIML.tc
Module : VDM2UIML

Name	#Calls	Coverage
VDM2UIML`id2str	39	37%
VDM2UIML`op2str	0	0%
VDM2UIML`writeF	0	0%
VDM2UIML`att2str	52	47%
VDM2UIML`how2str	52	63%
VDM2UIML`base2str	0	0%
VDM2UIML`call2str	0	0%
VDM2UIML`cpro2str	0	0%
VDM2UIML`head2str	1	38%
VDM2UIML`meta2str	0	0%
VDM2UIML`name2str	13	90%
VDM2UIML`part2str	20	98%
VDM2UIML`peer2str	0	0%
VDM2UIML`rule2str	0	0%
VDM2UIML`stru2str	1	$\surd$
VDM2UIML`uiml2str	0	0%
VDM2UIML`behav2str	20	14%

Name	#Calls	Coverage
VDM2UIML‘class2str	20	90%
VDM2UIML‘equal2str	0	0%
VDM2UIML‘event2str	0	0%
VDM2UIML‘logic2str	0	0%
VDM2UIML‘metas2str	0	0%
VDM2UIML‘model2str	0	0%
VDM2UIML‘param2str	0	0%
VDM2UIML‘parts2str	41	√
VDM2UIML‘refer2str	0	0%
VDM2UIML‘rules2str	0	0%
VDM2UIML‘style2str	20	√
VDM2UIML‘templ2str	0	0%
VDM2UIML‘value2str	0	0%
VDM2UIML‘where2str	20	63%
VDM2UIML‘action2str	0	0%
VDM2UIML‘dclass2str	0	0%
VDM2UIML‘dparam2str	0	0%
VDM2UIML‘events2str	0	0%
VDM2UIML‘export2str	52	66%
VDM2UIML‘interf2str	1	97%
VDM2UIML‘member2str	1	52%
VDM2UIML‘optype2str	0	0%
VDM2UIML‘params2str	0	0%
VDM2UIML‘prelog2str	0	0%
VDM2UIML‘repeat2str	0	0%
VDM2UIML‘script2str	0	0%
VDM2UIML‘source2str	52	50%
VDM2UIML‘toFileHTML	1	√
VDM2UIML‘acttype2str	0	0%
VDM2UIML‘content2str	20	14%
VDM2UIML‘dclasss2str	0	0%
VDM2UIML‘dmethod2str	0	0%
VDM2UIML‘dparams2str	0	0%
VDM2UIML‘inteles2str	2	42%
VDM2UIML‘members2str	2	√
VDM2UIML‘optypes2str	0	0%
VDM2UIML‘prelogs2str	0	0%
VDM2UIML‘repeats2str	20	27%
VDM2UIML‘urlname2str	0	0%
VDM2UIML‘acttypes2str	0	0%
VDM2UIML‘constant2str	0	0%
VDM2UIML‘dmethods2str	0	0%
VDM2UIML‘iterator2str	0	0%
VDM2UIML‘listener2str	0	0%

Name	#Calls	Coverage
VDM2UIML‘mapstype2str	0	0%
VDM2UIML‘property2str	13	98%
VDM2UIML‘proptype2str	13	30%
VDM2UIML‘whentrue2str	0	0%
VDM2UIML‘bydefault2str	0	0%
VDM2UIML‘condition2str	0	0%
VDM2UIML‘constants2str	0	0%
VDM2UIML‘dproperty2str	0	0%
VDM2UIML‘listeners2str	0	0%
VDM2UIML‘proptypes2str	26	√
VDM2UIML‘whenfalse2str	0	0%
VDM2UIML‘wherepart2str	20	50%
VDM2UIML‘dcomponent2str	0	0%
VDM2UIML‘properties2str	30	√
VDM2UIML‘dcomponents2str	0	0%
VDM2UIML‘dproperties2str	0	0%
VDM2UIML‘restructure2str	0	0%
VDM2UIML‘used-in-tag2str	0	0%
VDM2UIML‘constantname2str	0	0%
VDM2UIML‘presentation2str	0	0%
VDM2UIML‘whentruetype2str	0	0%
VDM2UIML‘whentruetypes2str	0	0%
VDM2UIML‘iteratorOptions2str	0	0%
Total Coverage		**23%**

end *VDM2UIML*

4 Prototype

In order to experiment the animation of VDM methods, mainly *OLAP* functions, we have decided to create a HTML prototype whereby we can visualize all table transformations. This application uses our VDM specifications, *UIMLSpec* and *UIMLSpecTab*. From CGI HTML forms behavior, the *VDM-SL* methods are called, than UIML is generated and render again to HTML.

The prototype was developed using CGI mechanism, with PHP technology and Java Applets.

Figure 1 depicts the architecture of our prototype and Figure 2 depicts its front-end.

The methods prototyped are:

- Rotate
- Partitioning and Projection
- Get and Set column

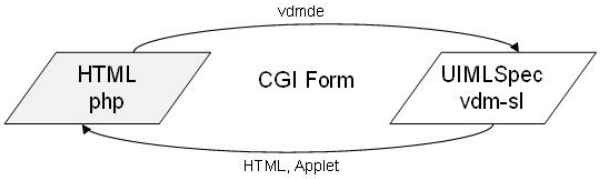

Figure 1: Prototype architecture

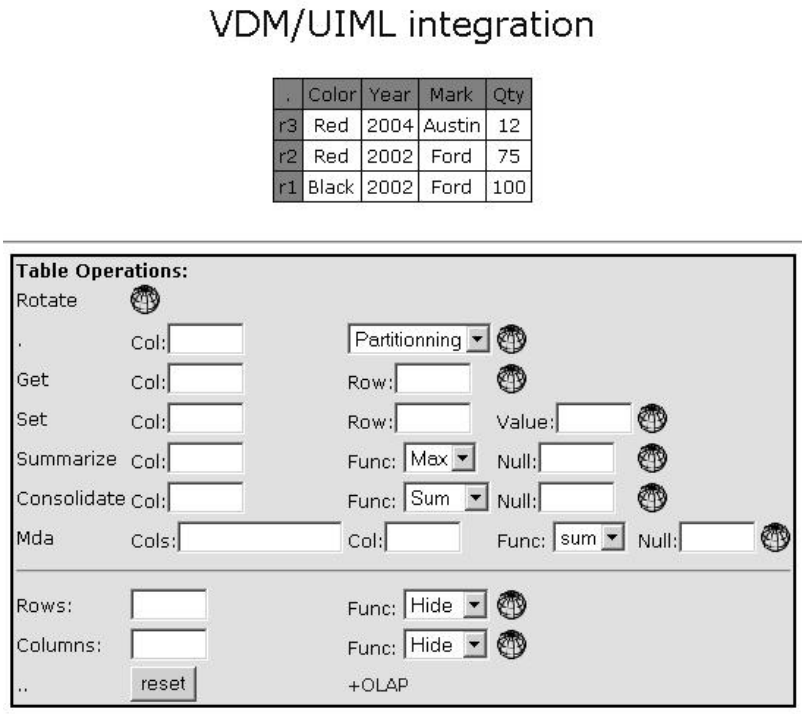

Figure 2: VDM/UIML integration prototype

- Summarize
- Consolidate
- Multidimensional analysis
- Hide, Show, Add, Delete columns and rows
- Roll-Up and Drill-Down

4.1 Source code

Listing 1: PHP CGI for HTML prototype

```
2 <?php
  // -------------------------------------------------------
4 //by lufer
  // -------------------------------------------------------
```

```
if (!session_is_registered('Me')) {

    $filearray=array();
    session_register('Me');
    $vdm_exp="";
}
else {
    session_start();
}
?>

<HTML>
<HEAD>
  <TITLE>Table IO</TITLE>
  <link rel="stylesheet" type="text/css" href="style.css">
</HEAD>
<BODY>
<center>
<TABLE border="0" width="620">
<TR><TD>
<H2>VDM/UIML integration</H2>
</TD></TR>
<TR><TD align="center">
<TABLE class="vdm" cellpadding="3" cellspacing="0" border="1"
bordercolor="#000080" style="border-collapse: collapse">

<?PHP

import_request_variables("gP","_");


//--------------- main --------------------

if (empty($_FirstPass)){
    global $filearray;

    if (file_exists("t.arg")) {
        $filearray=file("t.arg");
        file2str($filearray);
    }
    ShowForm();
    $fp = fopen("t1.arg","w");
    fwrite($fp,$vdm_exp);
    fclose($fp);
 }
else{
     if (file_exists("t1.arg")){
        $filearray=file("t1.arg");
        file2str($filearray);
     }
     if (!empty($_B)){
        foreach ( $_B as $key => $value ) {
            output($key,$value,$filearray);
        }
     }
}

exit;


//--------------- functions ----------------

function output ($key,$value,$filearray){
    global $prefix;
    global $sufix;
```

```
72      global $vdm_exp;
        global $_row;
74      global $_col;
        global $_rowvalue, $_colvalue,$_newvalue, $_tipo, $_colTable;
76      global $_mdaType,$_mdaCols,$_mdaCol, $_mdaNull;
        global $_colfunc,$_func,$_nullvalue;
78      global $_getCol,$_getRow;
        global $_setCol,$_setRow,$_setNew;
80      global $_consCol, $_consFunc, $_consNull;
        global $_sumCol, $_sumFunc, $_sumNull;
82      global $_colOper,$_rowOper;

84      //clean strings
        $_mdaCols=clear($_mdaCols);
86      $_mdaCol=clear($_mdaCol);
        $_consCol=clear($_consCol);
88      $_sumCol=clear($_sumCol);
        $_getCol = clear($_getCol);
90      $_getRow = clear($_getRow);
        $_setCol = clear($_setCol);
92      $_setRow = clear($_setRow);
        $_setNew = clear($_setNew);
94      $_colTable = clear($_colTable);
        $_row = clear($_row);
96      $_col = clear($_col);

98      switch($key){
            case "ROTATE":
100             case "rotate":
                    $fp = fopen("t1.arg","w");
102                 $fp1 = fopen("tout.arg","w");
                    $pref_oper="UIMLSpecAbs`rotate(";
104                 $suf_oper=")";
                    $vdm_exp = $pref_oper .$vdm_exp .$suf_oper;

                    $prefix = "UIMLSpecAbs`outHtml(";
108                 $sufix=",\"res.html\",1)";

110                 fwrite($fp1,$prefix);
                    fwrite($fp1,$vdm_exp);
112                 fwrite($fp1,$sufix);
                        fclose($fp1);
114                 fwrite($fp,$vdm_exp);
                    fclose($fp);
116                 exec('vdmtest tout.arg');
                    ShowForm();
118                 break;
            case "table":
120             switch($_tipo){
                    case "Part":
122                     $fp = fopen("t1.arg","w");
                        $fp1 = fopen("tout.arg","w");
124                     $pref_oper="UIMLSpecAbs`partition(";
                        $suf_oper=",{" .$_colTable."})";
126                     $vdm_exp = $pref_oper .$vdm_exp .$suf_oper;

128                     $prefix = "UIMLSpecAbs`outHtml(";
                        $sufix=",\"res.html\",1)";

                        fwrite($fp1,$prefix);
132                     fwrite($fp1,$vdm_exp);
                        fwrite($fp1,$sufix);
134                     fclose($fp1);
                        fwrite($fp,$vdm_exp);
136                     fclose($fp);
                        exec('vdmtest tout.arg');
```

```
138                     ShowForm();
                        break;
140                 case "Proj":
                        $fp = fopen("t1.arg","w");
142                     $fp1 = fopen("tout.arg","w");
                        $pref_oper="UIMLSpecAbs`project(";
144                     $suf_oper=",{" . $_colTable."})";
                        $vdm_exp = $pref_oper .$vdm_exp .$suf_oper;

                        $prefix = "UIMLSpecAbs`outHtml(";
148                     $sufix=",\"res.html\",1)";

150                     fwrite($fp1,$prefix);
                        fwrite($fp1,$vdm_exp);
152                     fwrite($fp1,$sufix);
                        fclose($fp1);
154                     fwrite($fp,$vdm_exp);
                        fclose($fp);
156                     exec('vdmtest tout.arg');
                        ShowForm();
158                     break;
                }
160             break;


            case "MDA":
164             $fp = fopen("t1.arg","w");
                $fp1 = fopen("tout.arg","w");
166             $pref_oper="UIMLSpecAbs`mda[Value](";
                $suf_oper=",{".$_mdaCols."},".$_mdaCol.",UIMLSpecAbs`".
168             $_mdaType.",".$_mdaNull.")";
                $vdm_exp = $pref_oper .$vdm_exp .$suf_oper;
170             fwrite($fp1,"UIMLSpecAbs`mda2html(");
                fwrite($fp1,$vdm_exp);
172             fwrite($fp1,",\"res.html\")");
                fclose($fp1);
174             fwrite($fp,$vdm_exp);
                fclose($fp);
176             exec('vdmtest tout.arg');
                ShowForm();
178             break;

180         case "Sumariz":
                $fp = fopen("t1.arg","w");
182             $fp1 = fopen("tout.arg","w");
                $pref_oper="UIMLSpecAbs`summarize[int,int](";
184             $suf_oper=",".$_sumCol.",UIMLSpecAbs`".$_sumFunc.",".$_sumNull.")";
                $vdm_exp = $pref_oper .$vdm_exp .$suf_oper;
186             fwrite($fp1,"UIMLSpecAbs`outHtmlValue(");
                fwrite($fp1,$vdm_exp);
188             fwrite($fp1,",\"resOper.html\")");
                fclose($fp1);
190             fwrite($fp,$vdm_exp);
                fclose($fp);
192             exec('vdmtest tout.arg');
                ShowForm();
194             break;

196         case "Consolid":
                $fp = fopen("t1.arg","w");
198             $fp1 = fopen("tout.arg","w");
                $pref_oper="UIMLSpecAbs`consolidate[int,int](";
200             $suf_oper=",".$_consCol.",UIMLSpecAbs`".$_consFunc.",".$_consNull.")";
                $vdm_exp = $pref_oper .$vdm_exp .$suf_oper;
202             fwrite($fp1,"UIMLSpecAbs`outHtmlValue(");
                fwrite($fp1,$vdm_exp);
```

```
204             fwrite($fp1,",\"resOper.html\")");
                fclose($fp1);
206             fwrite($fp,$vdm_exp);
                fclose($fp);
208             exec('vdmtest tout.arg');
                ShowForm();
210             break;

212         case "SET":
                $fp = fopen("t1.arg","w");
214             $fp1 = fopen("tout.arg","w");
                $pref_oper="UIMLSpecAbs`setCell(";
216             $suf_oper="," . $_setRow."," .$_setCol."," .$_setNew.")";
                $vdm_exp = $pref_oper .$vdm_exp .$suf_oper;

                $prefix = "UIMLSpecAbs`outHtml(";
220             $sufix=",\"res.html\",1)";

222             fwrite($fp1,$prefix);
                fwrite($fp1,$vdm_exp);
224             fwrite($fp1,$sufix);
                fclose($fp1);
226             fwrite($fp,$vdm_exp);
                fclose($fp);
228             exec('vdmtest tout.arg');
                ShowForm();
230             break;

232         case "GET":
                $fp = fopen("t1.arg","w");
234             $fp1 = fopen("tout.arg","w");
                $pref_oper="UIMLSpecAbs`getCellValue(";
236             $suf_oper="," . $_getRow."," .$_getCol.")";
                $vdm_exp = $pref_oper .$vdm_exp .$suf_oper;

                $prefix = "UIMLSpecAbs`outHtmlValue(";
240             $sufix=",\"resOper.html\")";

242             fwrite($fp1,$prefix);
                fwrite($fp1,$vdm_exp);
244             fwrite($fp1,$sufix);
                fclose($fp1);
246             fwrite($fp,$vdm_exp);
                fclose($fp);
248             exec('vdmtest tout.arg');
                ShowForm();
250             break;

252         case "ROWOPER":
                switch($_rowOper){
254                 case "hide":
                        $fp = fopen("t1.arg","w");
256                     $fp1 = fopen("tout.arg","w");
                        $pref_oper="UIMLSpecAbs`hideRow(";
258                     $suf_oper="," . $_row.")";
                        $vdm_exp = $pref_oper .$vdm_exp .$suf_oper;

                        $prefix = "UIMLSpecAbs`outHtml(";
262                     $sufix=",\"res.html\",1)";

264                     fwrite($fp1,$prefix);
                        fwrite($fp1,$vdm_exp);
266                     fwrite($fp1,$sufix);

268                     fclose($fp1);
                        fwrite($fp,$vdm_exp);
```

```
270                     fclose($fp);
                        exec('vdmtest tout.arg');
272                     ShowForm();
                        break;
274                 case "show":
                        $fp = fopen("t1.arg","w");
276                     $fp1 = fopen("tout.arg","w");
                        $pref_oper="UIMLSpecAbs`showRows(";
278                     $suf_oper=")";
                        $vdm_exp = $pref_oper .$vdm_exp .$suf_oper;

                        $prefix = "UIMLSpecAbs`outHtml(";
282                     $sufix=",\"res.html\",1)";

284                     fwrite($fp1,$prefix);
                        fwrite($fp1,$vdm_exp);
286                     fwrite($fp1,$sufix);

288                     fclose($fp1);
                        fwrite($fp,$vdm_exp);
290                     fclose($fp);
                        exec('vdmtest tout.arg');
292                     ShowForm();
                        break;
294                 case "add":
                        $fp = fopen("t1.arg","w");
296                     $fp1 = fopen("tout.arg","w");
                        $pref_oper="UIMLSpecAbs`addRow(";
298                     $suf_oper="," . $_row.",{})";
                        $vdm_exp = $pref_oper .$vdm_exp .$suf_oper;

                        $prefix = "UIMLSpecAbs`outHtml(";
302                     $sufix=",\"res.html\",1)";

304                     fwrite($fp1,$prefix);
                        fwrite($fp1,$vdm_exp);
306                     fwrite($fp1,$sufix);

308                     fclose($fp1);
                        fwrite($fp,$vdm_exp);
310                     fclose($fp);
                        exec('vdmtest tout.arg');
312                     ShowForm();
                        break;
314                 case "del":
                        $fp = fopen("t1.arg","w");
316                     $fp1 = fopen("tout.arg","w");
                        $pref_oper="UIMLSpecAbs`dellRow(";
318                     $suf_oper="," . $_row.")";
                        $vdm_exp = $pref_oper .$vdm_exp .$suf_oper;

                        $prefix = "UIMLSpecAbs`outHtml(";
322                     $sufix=",\"res.html\",1)";

324                     fwrite($fp1,$prefix);
                        fwrite($fp1,$vdm_exp);
326                     fwrite($fp1,$sufix);
                        fclose($fp1);
328                     fwrite($fp,$vdm_exp);
                        fclose($fp);
330                     exec('vdmtest tout.arg');
                        ShowForm();
332                     break;
                }};break;
334         case "COLOPER": switch ($_colOper){
                    case "hide":
```

```
336                     $fp = fopen("t1.arg","w");
                        $fp1 = fopen("tout.arg","w");
338                     $pref_oper="UIMLSpecAbs`hideCol(";
                        $suf_oper="," . $_col.")";
340                     $vdm_exp = $pref_oper .$vdm_exp .$suf_oper;

342                     $prefix = "UIMLSpecAbs`outHtml(";
                        $sufix=",\"res.html\",1)";

                        fwrite($fp1,$prefix);
346                     fwrite($fp1,$vdm_exp);
                        fwrite($fp1,$sufix);

                        fclose($fp1);
350                     fwrite($fp,$vdm_exp);
                        fclose($fp);
352                     exec('vdmtest tout.arg');
                        ShowForm();
354                     break;
                    case "show":
356                     $fp = fopen("t1.arg","w");
                        $fp1 = fopen("tout.arg","w");
358                     $pref_oper="UIMLSpecAbs`showCol(";
                        $suf_oper=",".$_col.")";
360                     $vdm_exp = $pref_oper .$vdm_exp .$suf_oper;

362                     $prefix = "UIMLSpecAbs`outHtml(";
                        $sufix=",\"res.html\",1)";

                        fwrite($fp1,$prefix);
366                     fwrite($fp1,$vdm_exp);
                        fwrite($fp1,$sufix);

                        fclose($fp1);
370                     fwrite($fp,$vdm_exp);
                        fclose($fp);
372                     exec('vdmtest tout.arg');
                        ShowForm();
374                     break;
                        break;
376                 case "add":
                        $fp = fopen("t1.arg","w");
378                     $fp1 = fopen("tout.arg","w");
                        $pref_oper="UIMLSpecAbs`addCol(";
380                     $suf_oper="," . $_col.",{})";
                        $vdm_exp = $pref_oper .$vdm_exp .$suf_oper;

                        $prefix = "UIMLSpecAbs`outHtml(";
384                     $sufix=",\"res.html\",1)";

386                     fwrite($fp1,$prefix);
                        fwrite($fp1,$vdm_exp);
388                     fwrite($fp1,$sufix);

390                     fclose($fp1);
                        fwrite($fp,$vdm_exp);
392                     fclose($fp);
                        exec('vdmtest tout.arg');
394                     ShowForm();
                        break;
396                 case "addN":
                        $fp = fopen("t1.arg","w");
398                     $fp1 = fopen("tout.arg","w");
                        $pref_oper="UIMLSpecAbs`addColls(";
400                     $suf_oper=",". $_col.")";
                        $vdm_exp = $pref_oper .$vdm_exp .$suf_oper;
```

```
                    $prefix = "UIMLSpecAbs`outHtml(";
404                 $sufix=",\"res.html\",1)";

406                 fwrite($fp1,$prefix);
                    fwrite($fp1,$vdm_exp);
408                 fwrite($fp1,$sufix);

410                 fclose($fp1);
                    fwrite($fp,$vdm_exp);
412                 fclose($fp);
                    exec('vdmtest tout.arg');
414                 ShowForm();
                    break;
416             };
                break;

        case "reset":
420             $fp = fopen("resOper.html","w");
                fclose($fp);
422             exec('vdmtest t0.arg');
                ShowForm();
424             $vdm_exp="";
                $filearray=file("t.arg");
426             file2str($filearray);
                $fp = fopen("t1.arg","w");
428             fwrite($fp,$vdm_exp);
                fclose($fp);
430             break;
    }

}


436 //-------------------------------------------------------

438 function body($fp,$filearray){
        while(list(,$oneline)=each($filearray)){
440         fwrite($fp,$oneline);
    }
442 }

444 //-------------------------------------------------------

446 function file2str($filearray){
    global $vdm_exp;
448 while(list(,$oneline)=each($filearray)){
        $vdm_exp .=$oneline;
450 }
}

//-------------------------------------------------------

//remove character \
456 function clear($x)
{
458 $x = ereg_replace("[\\]","",$x);
    return(trim($x));
460 }

462 //-------------------------------------------------------

464 function ShowForm() {
  global $PHP_SELF;

?>
```

```
<?
if (file_exists("res.html")){
    include("res.html");
}
else{
    include("res_i.html");
}

if (file_exists("resOper.htnl")) { include("resOper.html");}

$HTML=<<<HTML
</tr></table>
<center>
<br/>
<p>
<FORM ACTION="$PHP_SELF">
  <INPUT TYPE="HIDDEN" NAME="FirstPass" VALUE="No">
   <p>
   <hr>
   <table class="oper" cellSpacing="1" cellPadding="1" border="0"
    style="border: 2px solid #000000">

   <tr>
   <td colspan=3><b>Table Operations:</b></td>
    </tr>
   <tr><td>Rotate</td>
   <td><input type="image" src="web.gif" height="20" width="20"
   border="0" Alt="Submit" name="B[ROTATE]"></td>
    </tr>
   <tr>
   <tr><td>.</td>
   <td>Col:<input type="text" name="colTable" size="5"></td>
   <td>
   <select name="tipo">
    <option value="Part">Partitionning</option>
    <option value="Proj">Projection</option>
   </select>
   </td>
   <td><input type="image" src="web.gif" height="20" width="20"
   border="0" Alt="Submit" name="B[table]"></td>

   <tr>
    <td>Get</td>
        <td>Col:<input type="text" name="getCol" size="5"></td>
        <td>Row:<input type="text" name="getRow" size="5"></td>
    <td><input type="image" src="web.gif" height="20" width="20"
    border="0" Alt="Submit" name="B[GET]"></td>
   </tr>

     <tr>
    <td>Set</td>
        <td>Col:<input type="text" name="setCol" size="5"></td>
        <td>Row:<input type="text" name="setRow" size="5"></td>
        <td>Value:<input type="text" name="setNew" size="5"></td>
    <td><input type="image" src="web.gif" height="20" width="20"
    border="0" Alt="Submit" name="B[SET]"></td>
   </tr>

   <tr>
    <td>Summarize</td>
    <td>Col:<input type="text" name="sumCol" size="5"></td>
        <td>Func:
        <select name="sumFunc">
            <option value="max">Max</option>
            <option value="min">Min</option>
        </select>
```

```
534      </td>
             <td>Null:<input type="text" name="sumNull" size="5"></td>
536      <td><input type="image" src="web.gif" height="20" width="20"
         border="0" Alt="Submit" name="B[Sumariz]"></td>
538     </tr>
        <tr>
540      <td>Consolidate</td>
         <td>Col:<input type="text" name="consCol" size="5"></td>
542      <td>Func:
             <select name="consFunc">
544              <option value="sum">Sum</option>
                 <option value="avg">Avg</option>
546              <option value="count">Count</option>
             </select>
548      </td>
             <td>Null:<input type="text" name="consNull" size="5"></td>
550      <td><input type="image" src="web.gif" height="20" width="20"
         border="0" Alt="Submit" name="B[Consolid]"></td>
552     </tr>

554     <!--MDA -->
        <tr>
556      <td>Mda</td>
         <td>Cols:<input type="text" name="mdaCols" size="15"></td>
558          <td>Col:<input type="text" name="mdaCol" size="5"></td>
             <td>Func:
560          <select name="mdaType">
             <option value="sum">sum</option>
562          <option value="avg">avg</option>
         </select>
564      </td>
             <td>Null:<input type="text" name="mdaNull" size="5"></td>
566      <td><input type="image" src="web.gif" height="20" width="20"
         border="0" Alt="Submit" name="B[MDA]"></td>
568     </tr>

570     <tr><td colspan=6><hr></td></td>
        <tr>
572     <td>Rows:</td>
        <td><input type="text" name="row" size="5"></td>
574     <td>Func:
        <select name="rowOper">
576      <option value="hide">Hide</option>
         <option value="show">Show</option>
578      <option value="add">Add</option>
         <option value="del">Del</option>
580     </select>
        </td>
582     <td><input type="image" src="web.gif" height="20" width="20"
        border="0" Alt="Submit" name="B[ROWOPER]"></td>
584     </tr>

586     <tr>
        <td>Columns:</td>
588     <td><input type="text" name="col" size="5"></td>
        <td>Func:
590     <select name="colOper">
         <option value="hide">Hide</option>
592      <option value="show">Show</option>
         <option value="add">Add</option>
594     </select>
        </td>
596     <td><input type="image" src="web.gif" height="20" width="20"
        border="0" Alt="Submit" name="B[COLOPER]"></td>
598     </tr>
```

```
600     <tr>
          <td>..</td>
602       <td><input type="submit" value="reset" name="B[reset]"></td>
          <td><a href="entrada1.php">+OLAP</a></td>
604       </tr>
        </table>
606     </p>
    HTML;
608 echo $HTML;
    } # End of function ShowForm

    ?>
612 </TR></TR></TABLE>
    </center>
614 </BODY>
    </HTML>
```

4.2 Prototype settings and requirements

Our prototype works on *Windows* platform with any *httpd* server (we used IIS) which allow PHP processing. There are some particular files to system configuration and particular scripts (batch files) to support the process.

Is is necessary to have VDM-SL installed as well as Harmonia UIML rendering engines (http://www.harmonia.com) mainly *u2ji* (java rendering), *u2h* (HTML rendering) and *u2w* (WML rendering).

Once installed UIML renderers, one must set windows system variables according to *Harmonia* documentation (http://www.harmonia.com/products/index.htm). *Harmonia LiquidUI* browser (trial release) is also important to be installed. Once expired, one can contact directly *Harmonia* (*support@harmonia.com*) to get an academic license.

Listing 2: *vdmtest.bat*: UIML VDM-SL testing batch file

```
    @echo off
2   rem Tests the UIML specification for one test argument

4   rem -- Output the argument to stdout (for redirect) and
    rem -- "con" (for user feedback)
6   echo VDM Test: '%1' > con
    echo VDM Test: '%1'

    rem Assumes specification in Word RTF Format. Change below if otherwise.

    set SPEC=UIMLSpec30.vdm
12  set SPEC1=VDM2UIML.vdm
    set IO=io.vdm
14  set TC= UIML.tc

16  PATH = %SYSTEMROOT%\SYSTEM32;%SYSTEMROOT%;%SYSTEMROOT%\SYSTEM32\WBEM;
    PATH = %PATH%;D:\TEXMF\MIKTEX\BIN;H:\PROGRAM FILES\THE IFAD VDM-SL TOOLBOX V3.7.2\BIN

    vdmde -i -P -R %TC% -O %1.res %1 %SPEC1% %SPEC% %IO%

    rem -- Check for difference between result of execution and
22  rem -- expected result.

24  if EXIST %1.exp fc /w %1.res %1.exp

26  :end
```

Listing 3: Batch file to renderer Java from UIML

```
u2ji %1
```

Listing 4: Java Applet viewer from UIML

```
appletviewer -J-Djava.security.policy=.java.policy resJava.html
```

Listing 5: *final.prj*: VDM-TOOLS project

```
b6,k11,ProjectFilef3,f3,e2,m4,filem42,io.vdme2,m4,filem50,
UIMLSpec30.vdme2,m4,filem56,TableIOCaseStudy.vdm
```

Listing 6: VDM-TOOLS utilization

```
1 - start VDM-SL ToolBox

2 - Load project final.prj

3 - Open Interpreter window:

        a) init

        b) push UIMLSpecTab

        c) now any command can be test. Examples:

                - p t
                - p outHtml(t,"file.html",1)
```

4.3 Some frontend results

Next figures depict the results of applying the main OLAP features over initial data set.

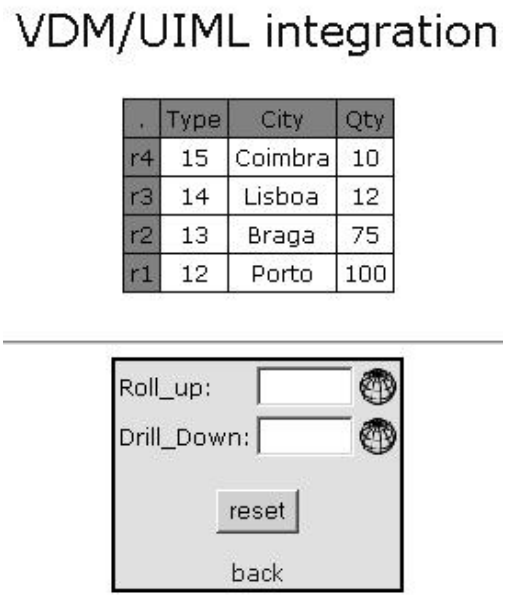

Figure 3: VDM/UIML integration test case - OLAP

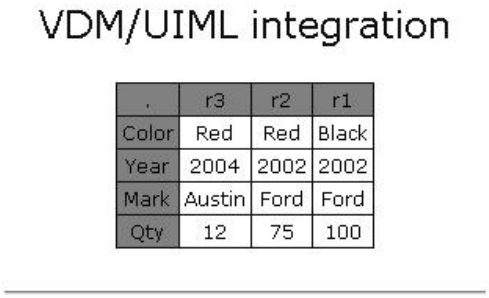

Figure 4: Rotation operation

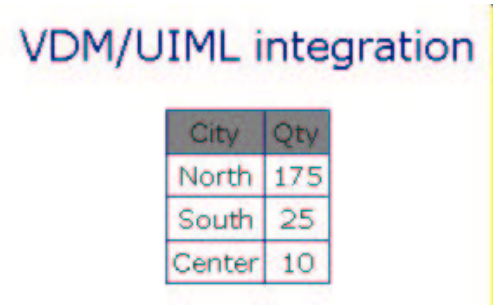

Figure 5: Roll-Up operation

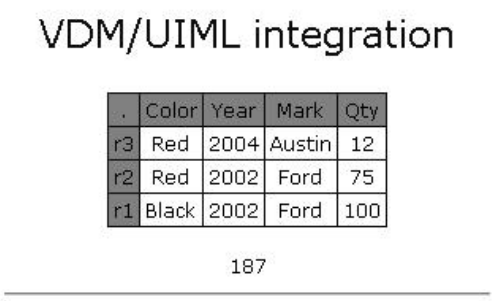

Figure 6: Consolidation operation (*sum of Qty*)

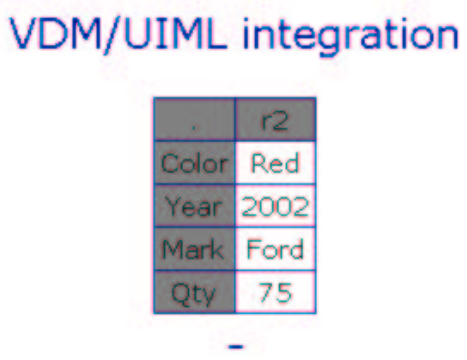

Figure 7: Projection (”r3”) after Rotation operation

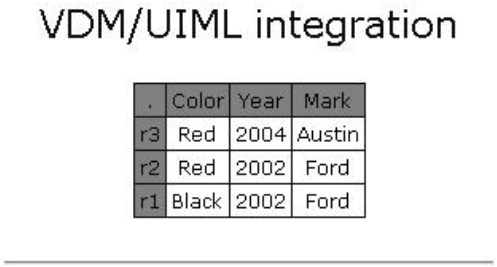

Figure 8: Hiding column operation

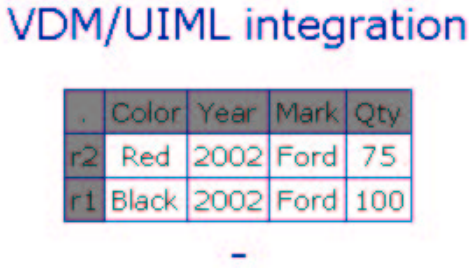

Figure 9: Hiding row operation

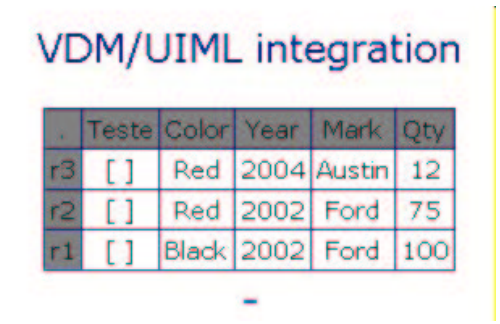

Figure 10: Add column operation (”Teste”)

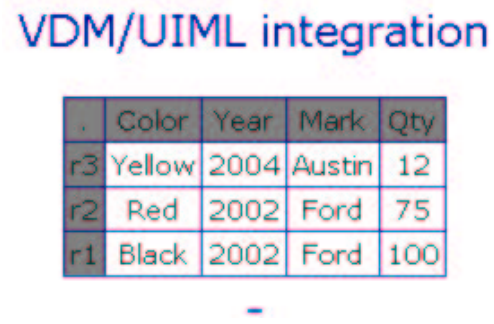

Figure 11: Set cell value (r3,Color)=”yellow”

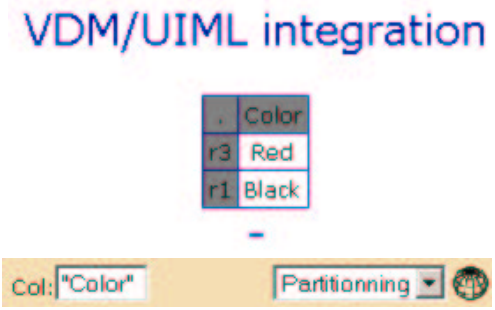

Figure 12: Partition operation on Color column

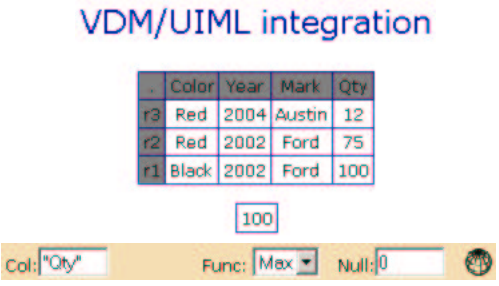

Figure 13: Summarize operation on Qty column

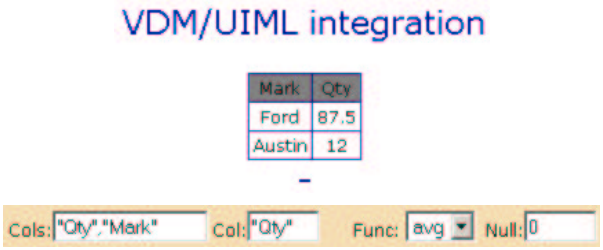

Figure 14: Multidimensional Analysis (average) over Qty