
VDM Animation for a Wider Range of Stakeholders

**Tomohiro Oda
Yasuhiro Yamamoto
Kumiyo Nakakoji
Keijiro Araki
Peter Gorm Larsen**

**Software Research Associates, Inc.
University of Tokyo
Kyoto University
Kyushu University
Aarhus University**

This work is supported by Grant-in-Aid for Scientific Research (S) 24220001

expressiveness

Formal specification

A formal specification of a system explains the system's functionality:

- what concepts are involved
- what should be achieved

- **It's hard to express things like "user's feelings"** in formal specs, which may change value of the system, i.e.
 - crispy transition of modes
 - instant and responsive update of info

Expressiveness of Formal Specs

- Animation can let people to experience the specified system
 - implications to UI
 - implications to client modules
 - implications to programs that is out of the scope of the formal spec.
 - implications to the value of the system

Animation can be understood by people who have no formal methods background.

Topics in this talk

to extend users of formal specs
from readers to experiential group

- **formal engineers**
 - overture tool, VDMTools, **VDMPad**
- **programmers and testers**
 - **Webly Walk-Through, pyVDMC**
- **UI designers**
 - **Lively Walk-Through**
- **non-engineering stakeholders**
 - **Cloudly Walk-Through**

basis

VDMPad

Animation to explore spec space

The screenshot shows the VDMPad web application running in a browser. The interface is divided into several sections, each with a label and a bracket on the right side:

- Specification Editor:** Contains a code editor with the following code:

```
1 types
2 Item = <BEER> | <WINE>;
3 Bag = map Item to nat1;
4 Order = seq of (Item * nat1);
5
6 values
7   I-> : Item -> nat1;
8   y of
9
10 stock : Bag
11 init s == s = mk_InVENTORY(empty)
12 end
13
14 operations
15 Buy : Order ==> ()
16 Buy(order) == for mk_(item, num) in order do
17   let
18     current_num =
19       if item in set_dom stock
20         then stock(item)
21         else 0
22   in
23     stock := stock ++ { item I-> current_num + num };
```

An arrow labeled "Menu Handle" points to a small square icon in the top-left corner of the code editor.
- State Area:** Shows the current state of the system. It includes a "DEFAULT" section with a "stock" field containing the value "{<BEER> I-> 10, <WINE> I-> 3}". Below this, there are two buttons labeled "BEER" and "WINE", and the values "10" and "3" are displayed. An "Initialize" button is located at the bottom right of this section.
- Workspace:** Contains a text area with the following code:

```
Buy([mk_(<BEER>, 10), mk_(<WINE>, 3)])
Sell([mk_(<BEER>, 2)])
```

An "evaluate" button is located at the bottom right of this section.
- Return Value:** Shows the result of the evaluation, which is "0".
- Message Area:** Currently empty.

outreach

Webly Walk-Through

Web API prototype for web programmers

- specify Web API in VDM-SL
- serve the Web API
 - `http://localhost:8087/<module>/<operation>?arg...`
 - VDM-SL \Leftrightarrow JSON conversion
- build a prototype of web client
 - html/css/javascript
- evaluate and discuss the Web API
 - history of Web API calls

Weby Walk-Through demo

Weby Walk-Through

powered by Squeak Smalltalk with VDMJ
about Weby Walk-Through

VDM

HTML

History

```
1 types
2 channel = nat
3 inv c == c in set channels;
4 station ::
5   ch : channel
6   name :- seq of char
7   url :- seq of char;
8
9 values
10 channels = {1,...,12};
11
12 state memory of
13 stations : set of station
14 current : channel
15 zapList : seq of channel
16 zapIndex : [nat1]
17
18 init
19 s == s = mk_memory(
20   {mk_station(1, "chem channel",
21     "https://www.youtube.com/embed/ZCiWfy615rw?t=9"),
22     mk_station(2, "teabreak TV",
23     "https://www.youtube.com/embed/q8HbihRt1eE?t=7"),
24     mk_station(3, "Delft!",
25     "https://www.youtube.com/embed/EoUcCVHH1Zk?t=12"),
26     mk_station(4, "Food TV",
27     "https://www.youtube.com/embed/R1deCm7vTWU?t=9"),
28     mk_station(5, "CGTV",
29     "https://www.youtube.com/embed/jl6OSWUXh14?t=10")},
30   1, [], nil)
31
32 end
33 operations
34 num : channel ==> station
35 num(x) (current : VDM return
```

save

update

DEFAULT

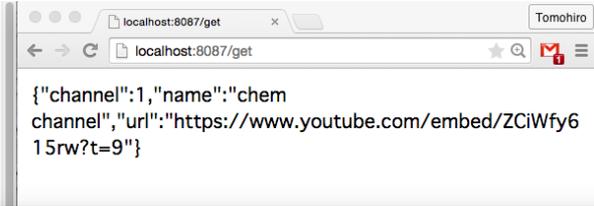
```
current 1
{mk_station(1, "chem channel",
"https://www.youtube.com/embed/ZCiWfy615rw?t=9"), mk_station(2, "teabreak TV",
"https://www.youtube.com/embed/q8HbihRt1eE?t=7"), mk_station(3, "Delft!",
"https://www.youtube.com/embed/EoUcCVHH1Zk?t=12"), mk_station(4, "Food TV",
"https://www.youtube.com/embed/R1deCm7vTWU?t=9"), mk_station(5, "CGTV",
"https://www.youtube.com/embed/jl6OSWUXh14?t=10")},
1, [], nil)
```

print

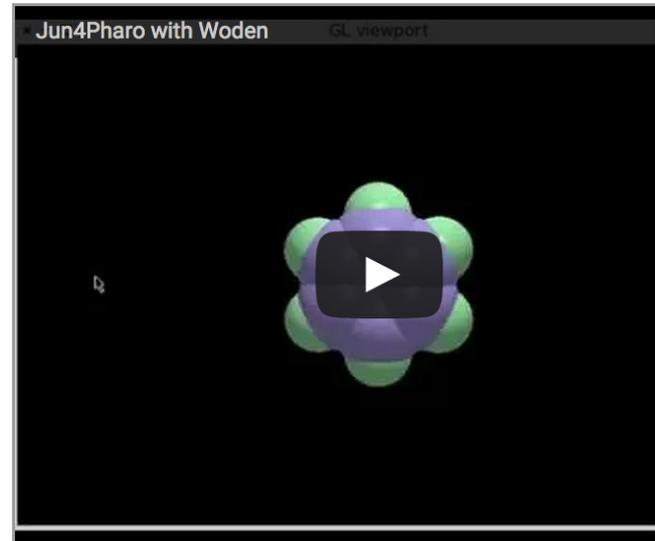
```
1 mk_station($ch, $name, $url) <=>
  {"channel": $ch, "name": $name, "url": $url}
```

save

update



1 - chem channel



<- ->

ZAP

+

pyVDMC

example fibonacci numbers

```
from pyVDMC import *
@vdm_module('n1', 'n2')
class fibonacci:
    """
    state State of
        n1 : nat
        n2 : nat
        init s == s = mk_State(0, 1)
    end
    operations
        next : () ==> nat
        next() == (dc1 n : nat := n1 + n2; n1 := n2;
n2 := n; return n)
        post RESULT = n1~ + n2~ and n1 = n2~ and n2 =
RESULT;
        prev : () ==> nat
        prev() == (dc1 n : nat := n2 - n1; n2 := n1;
n1 := n; return n2)
        post n1 + n2 = n2~ and n2 = n1~ and n2 =
RESULT;
    """
```

the class comment
as a VDM-SL spec

```
def __init__(self):
    self.n1 = 0
    self.n2 = 1
    @vdm_method
    def next(self):
        pass
    @vdm_test
    def prev(self):
        n = self.n2 - self.n1
        self.n2 = self.n1
        self.n1 = n
        return self.n2
```

the spec is used as
an implementation.

the spec is used as
a test oracle.

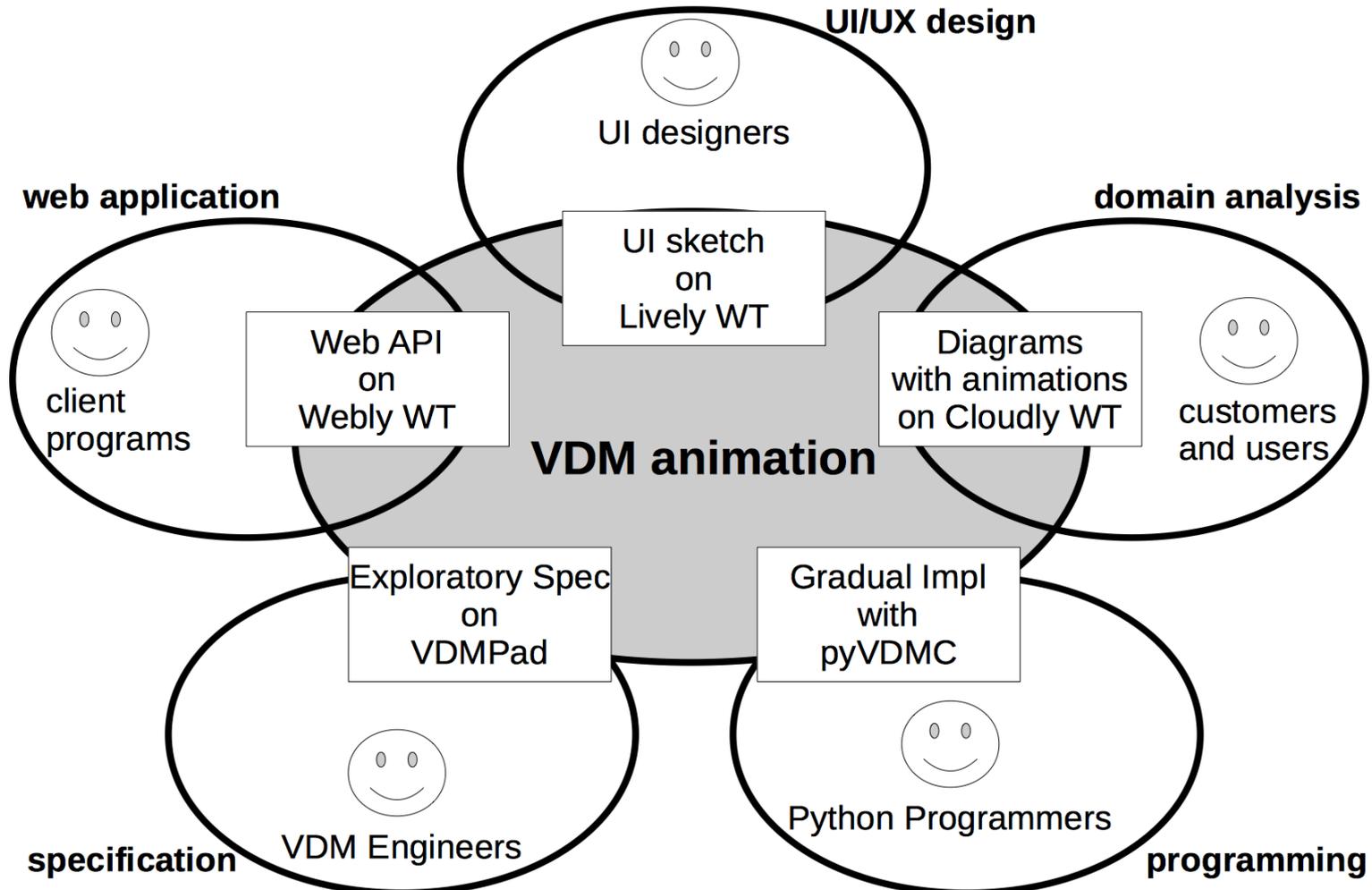
Lively Walk-Through prototyping with UI designers

The image displays a prototyping tool interface. On the left is a hand-drawn UI design for a radio tuner. It features a numeric keypad (1-12) with red boxes around each digit, a display showing '12', and several control buttons: 'PART', '+ZAP', 'MEM', 'DEL', 'BTN_MEM', 'CH', 'VOL', and 'V'. On the right, three software windows are open:

- VDM Browser**: Shows a table of variables and their values, and a block of code. The table lists 'current' (value: {'current' -> '12'}, 'zapIndex' (value: 'zapIndex' -> 'nil'), and 'zapList' (value: 'zapList' -> '11 5 12'). The code includes functions for navigating between zap channels and managing a list of zap channels.
- Livetalk Browser**: Shows a list of objects and their associated actions. The selected object is 'NEXTZAP', with the action 'nextZap() -> [DISPLAY]'.
- EventActionEditor**: Shows a table of events and their actions. The selected event is 'BTN9', with the action 'clicked' and the target 'MEM'.

conclusion

Summary



Overture in 1, 5, 10 years

in 1 year

- supports gradual implementation
 - all ops in VDM -> some ops in PL -> ...
-> most ops in PL -> all ops in PL

in 5 years

- supports more impl languages

in 10 years

- Dynabook of VDM
 - live spec
 - VDM spec and code on one dynamic media
 - VDM spec for everyone including end users