Lively Walk-Through: A Lightweight Formal Method in UI/UX Design

Tomohiro Oda

Software Research Associates, Inc.
Key Technology Laboratory

This work was supported by Kaken Japan Society for the Promotion of Science, Grants-in-Aid for Scientific Research, Grant Number 24220001.

Lively Walk-Through

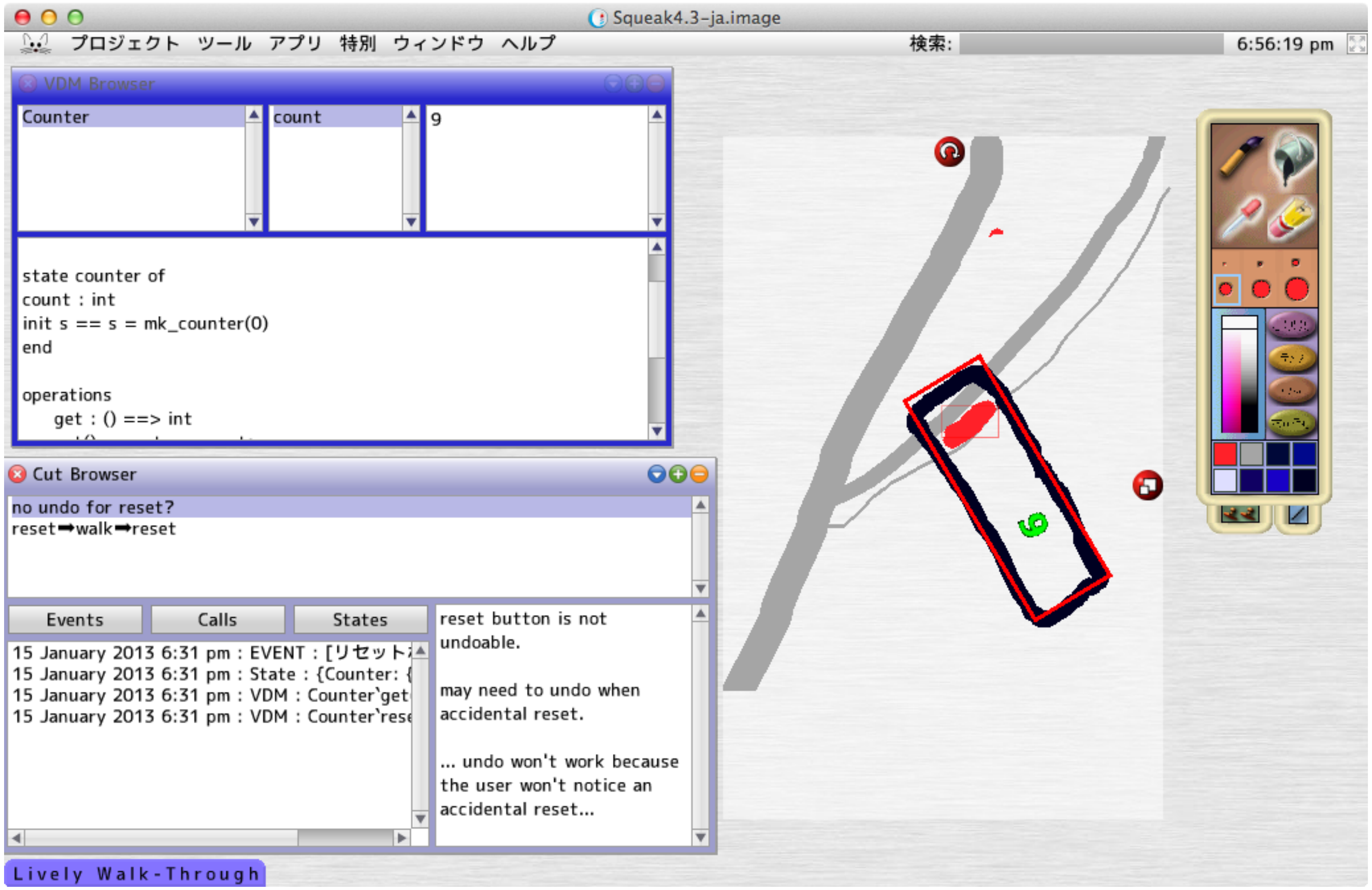
What is Lively Walk-Through?

UI Prototyping environment built on
VDM-SL interpreter and Smalltalk system

Users:
    VDM specifiers and UI/UX designers

Objective:
    To better undersntand the system
    To discuss and make agreements

Lively Walk-Through in Action



```
state counter of
count : int
init s == s = mk_counter(0)
end

operations
    get : () ==> int
```

no undo for reset?
reset➡walk➡reset

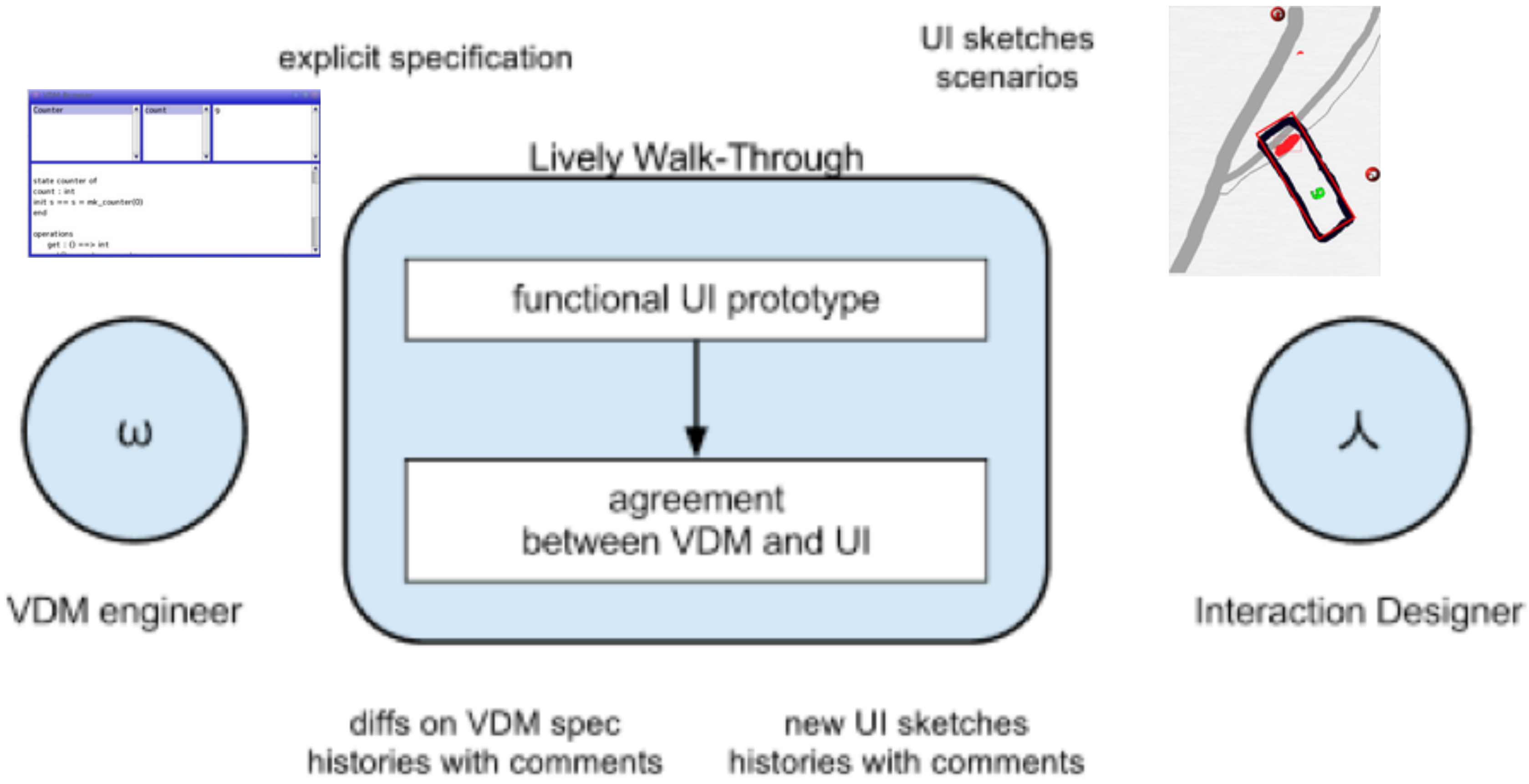
```
15 January 2013 6:31 pm : EVENT : [リセットi
15 January 2013 6:31 pm : State : {Counter: {
15 January 2013 6:31 pm : VDM : Counter`get
15 January 2013 6:31 pm : VDM : Counter`rese
```

reset button is not undoable.

may need to undo when accidental reset.

... undo won't work because the user won't notice an accidental reset...

UI Prototyping with Lively Walk-Through



```
state counter of
count : int
init s == s = mk_counter(0)
end

operations
    get : () ==> int
```

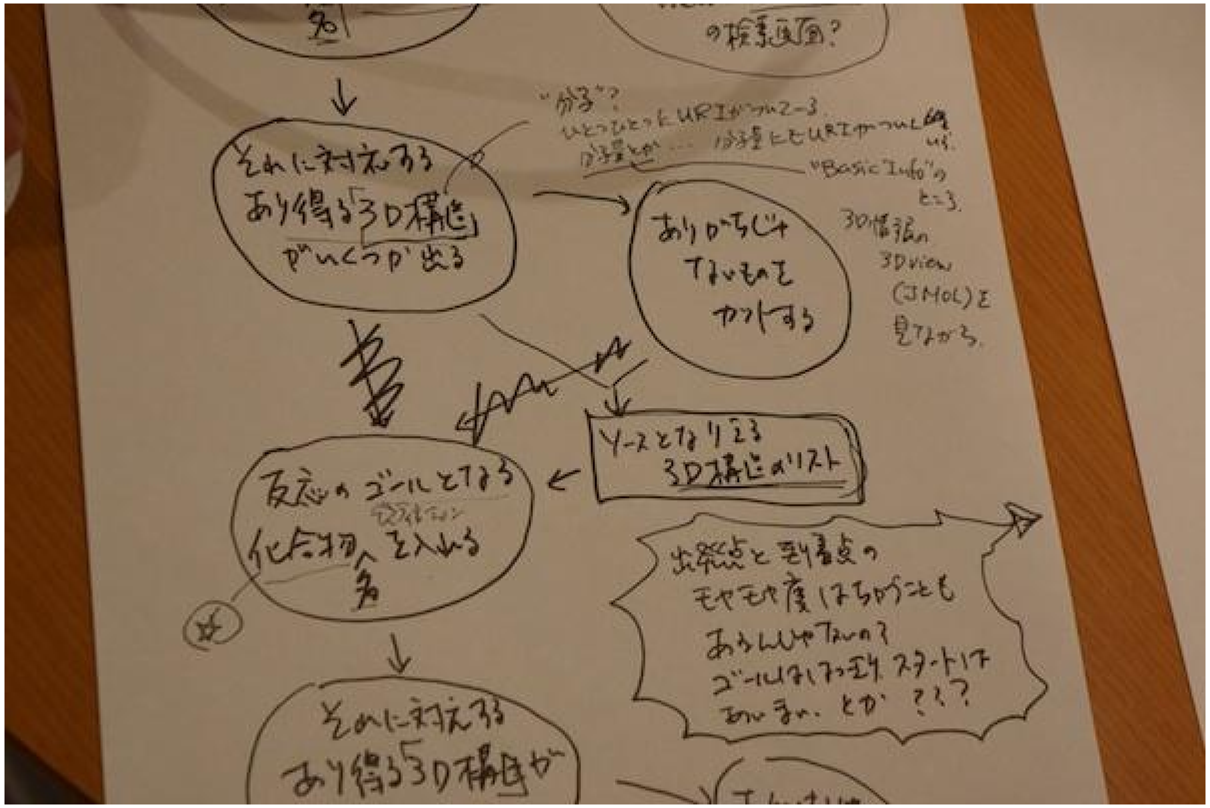
Story

Case Story :
Chemical Reaction Database

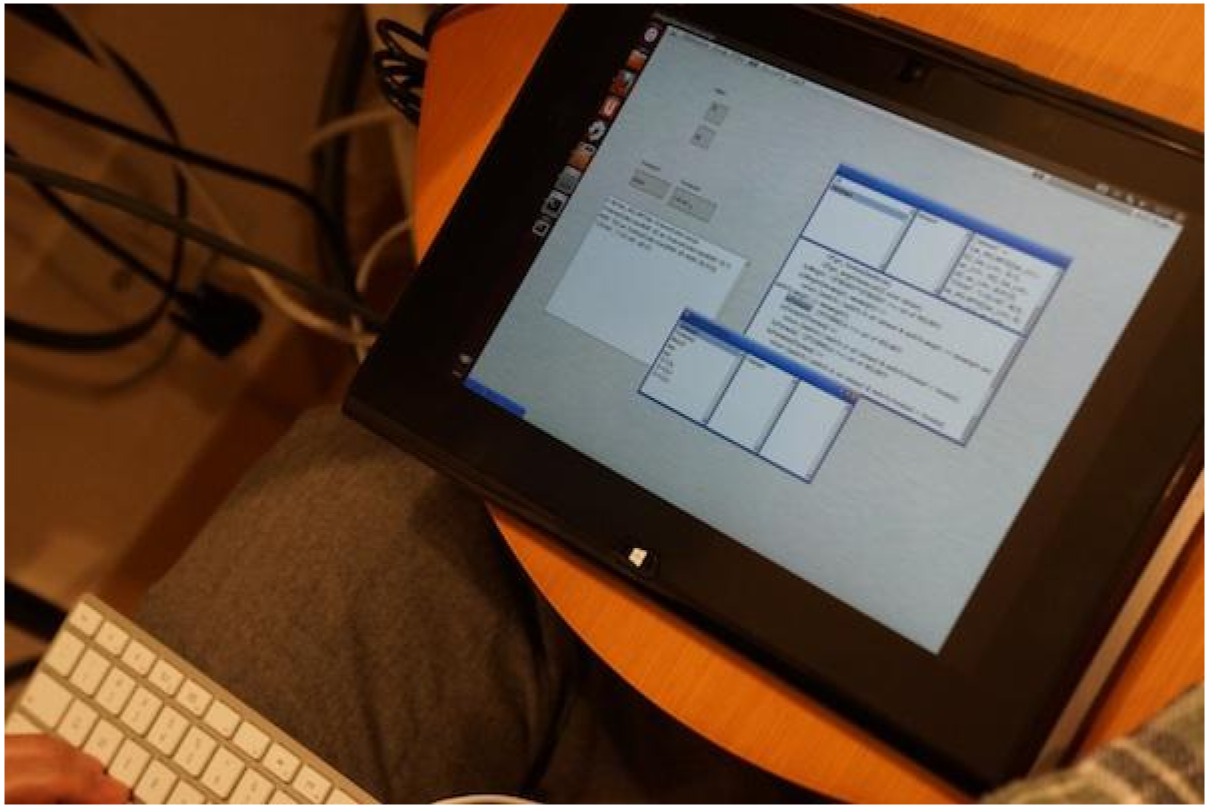
A VDM engineer and two UI designers
is working together on the Chem DB project

Feb 2013, the first meeting of the VDM
engineer and the UI designers

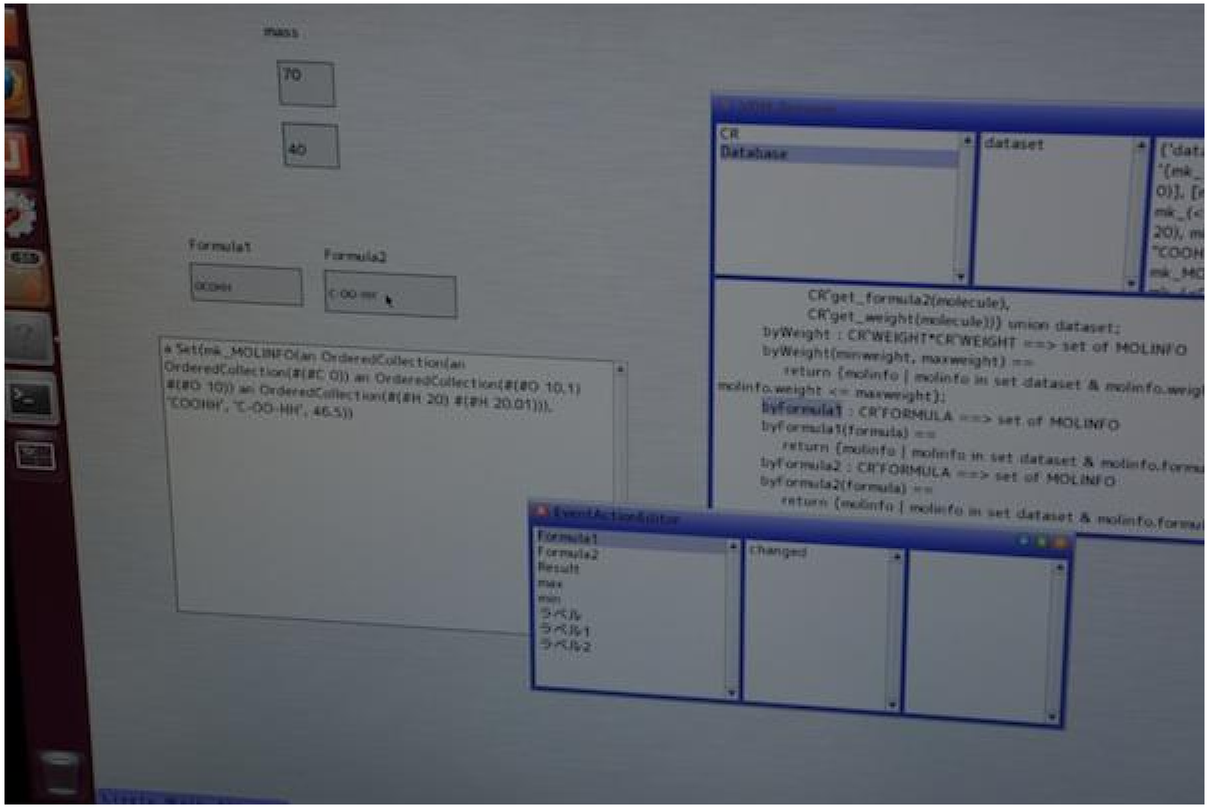
Overview of Customer's Requirements


の検索画面?
それに対応する ありうる3D構造がいくつか出る
"分子"? ひとつひとつにURIがついてる 分子量とか… 分子量にもURIがついてる "Basic Info"のところ
ありがちじゃないものをカットする
3D情報の 3D view (JMOL)を見ながら
ソースとなりうる 3D構造のリスト
反応のゴールとなる 化合物を入れる
出発点と到着点の モヤモヤ度はちがうこともあるんじゃないの? ゴールはピッタリ スタートはあいまい とか ???
それに対応する ありうる3D構造が

VDMer explaining the spec of DB



Simple GUI Prototype for Query

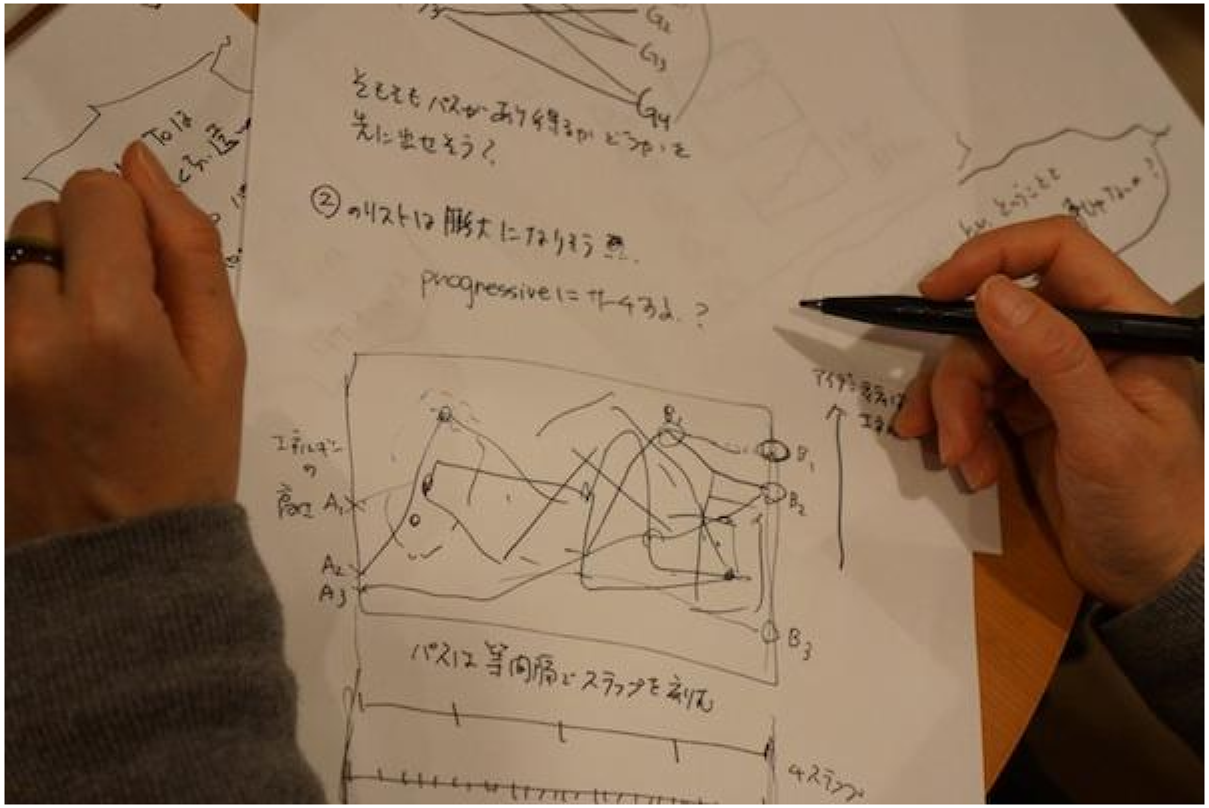


```
a Set(mk_MOLINFO(an OrderedCollection(an
OrderedCollection(#(#C 0)) an OrderedCollection(#(#O 10.1)
#(#O 10)) an OrderedCollection(#(#H 20) #(#H 20.01))),
'COOHH', 'C-OO-HH', 46.5))
```

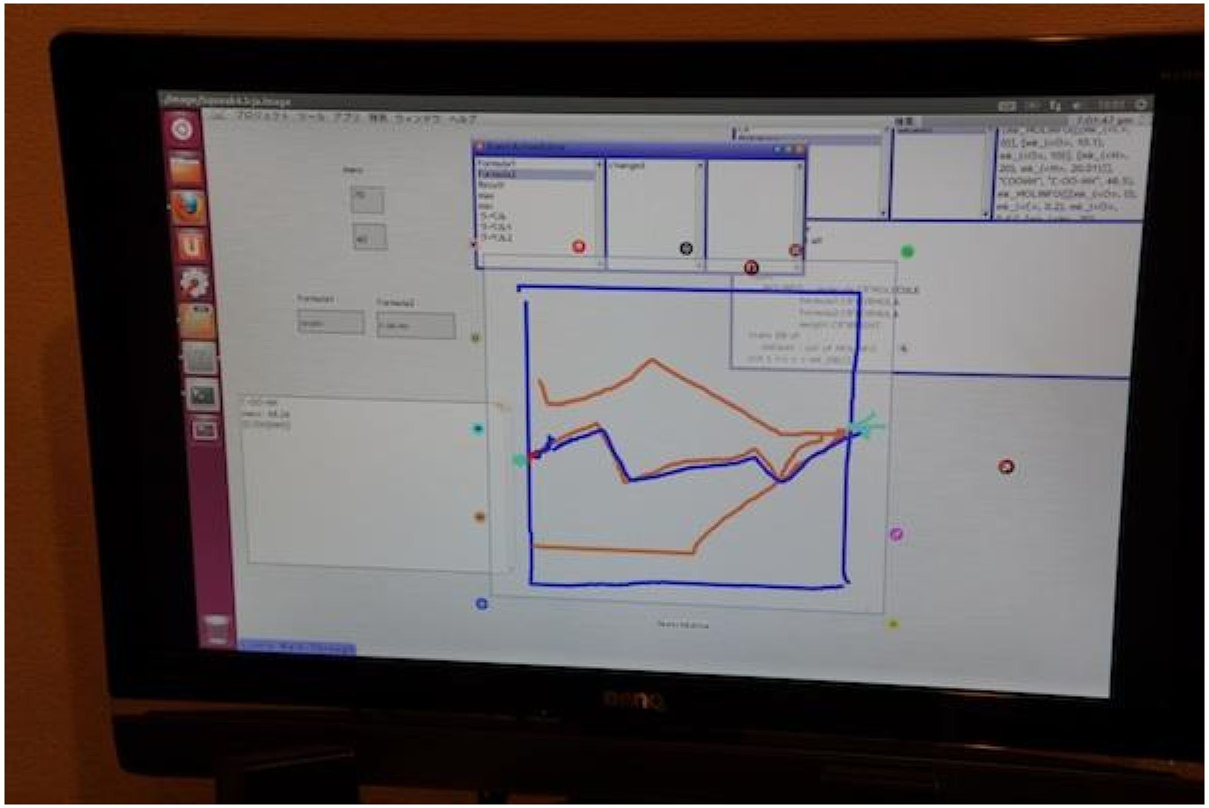
```
            CR`get_formula2(molecule),
            CR`get_weight(molecule))} union dataset;
    byWeight : CR`WEIGHT*CR`WEIGHT ==> set of MOLINFO
    byWeight(minweight, maxweight) ==
        return {molinfo | molinfo in set dataset & molinfo.weig
molinfo.weight <= maxweight};
    byFormula1 : CR`FORMULA ==> set of MOLINFO
    byFormula1(formula) ==
        return {molinfo | molinfo in set dataset & molinfo.formu
    byFormula2 : CR`FORMULA ==> set of MOLINFO
    byFormula2(formula) ==
        return {molinfo | molinfo in set dataset & molinfo.formu
```

Sketching UI Design for Query

Sketching UI Design for Search Result


そもそもパスがあり得るかどうかを先に出せそう?
② のリストは膨大になりそう
progressive にサーチする ?
パスは等間隔でスラップを刻む

Putting the Sketch into the Prototype



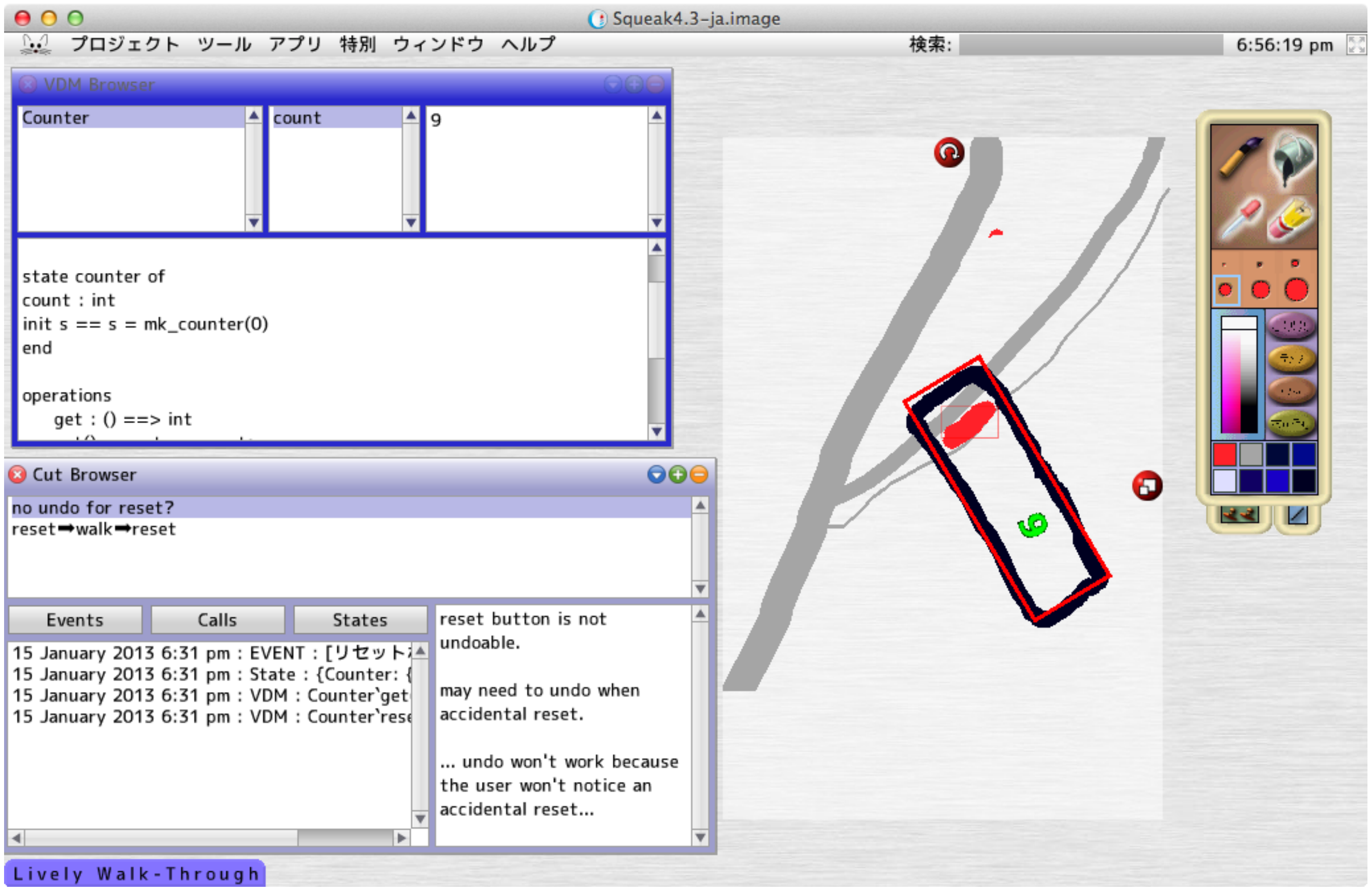
Assignment for the next session

VDM: writes a VDM spec for "reaction path"
estimates computational complexity of reachability test

UI: designs interactivity of energy-level graphs

Lively Walk-Through: System Design

Lively Walk-Through Prototyping Environment



```
state counter of
count : int
init s == s = mk_counter(0)
end

operations
    get : () ==> int
```

no undo for reset?
reset➡walk➡reset

```
15 January 2013 6:31 pm : EVENT : [リセットﾎ
15 January 2013 6:31 pm : State : {Counter: {
15 January 2013 6:31 pm : VDM : Counter`get
15 January 2013 6:31 pm : VDM : Counter`rese
```

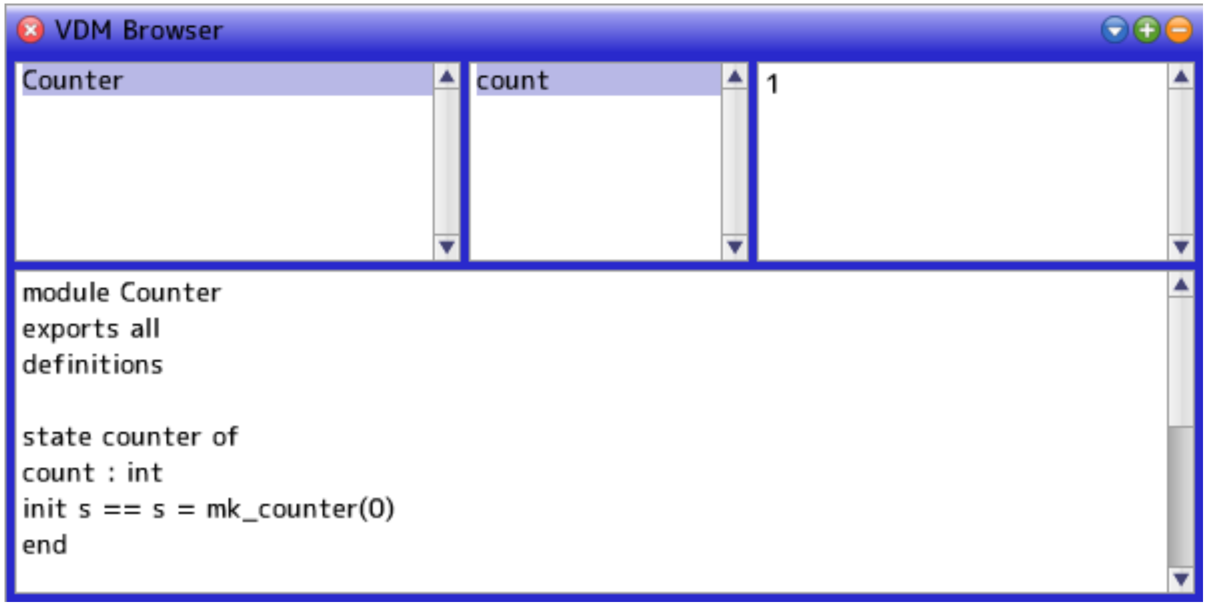
reset button is not undoable.

may need to undo when accidental reset.

... undo won't work because the user won't notice an accidental reset...

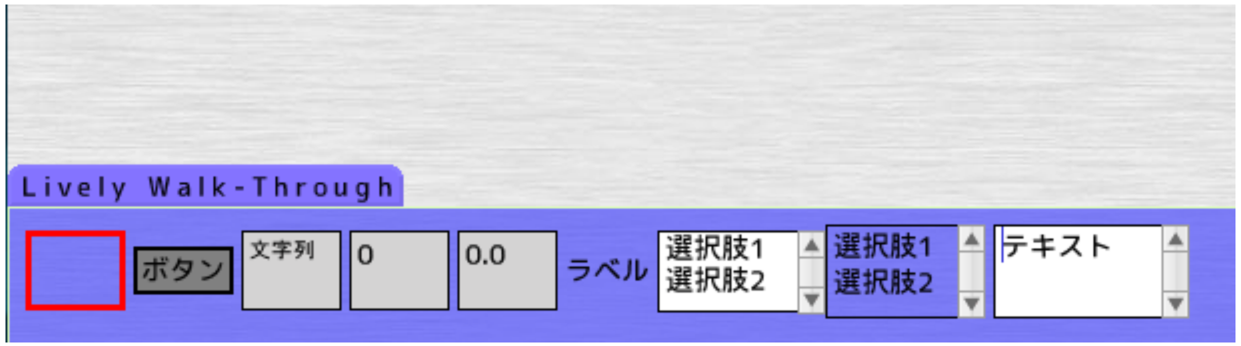
3 Layers for Animation

Top Layer: VDM Browser

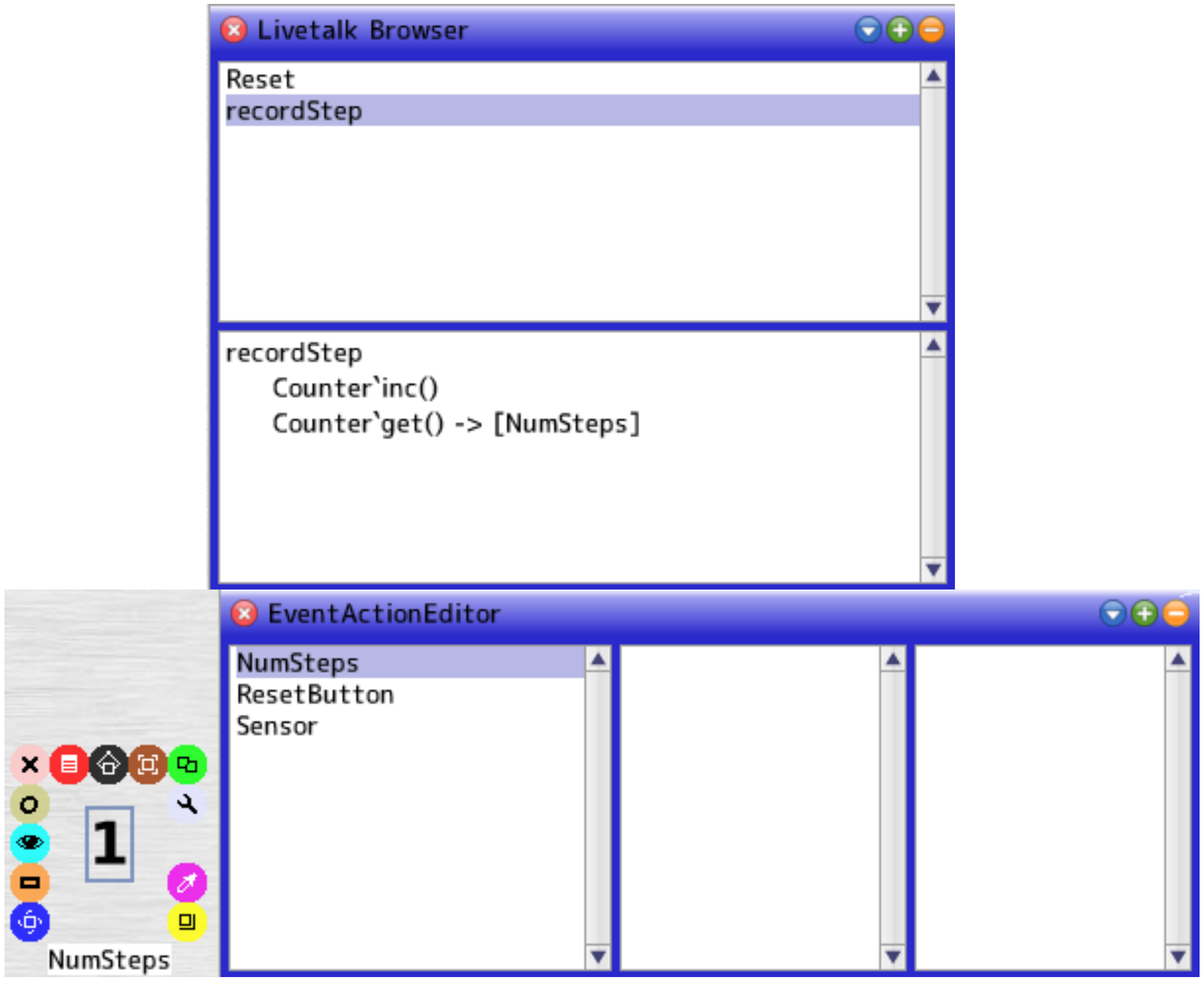


```
module Counter
exports all
definitions

state counter of
count : int
init s == s = mk_counter(0)
end
```

Bottom Layer: UI Parts



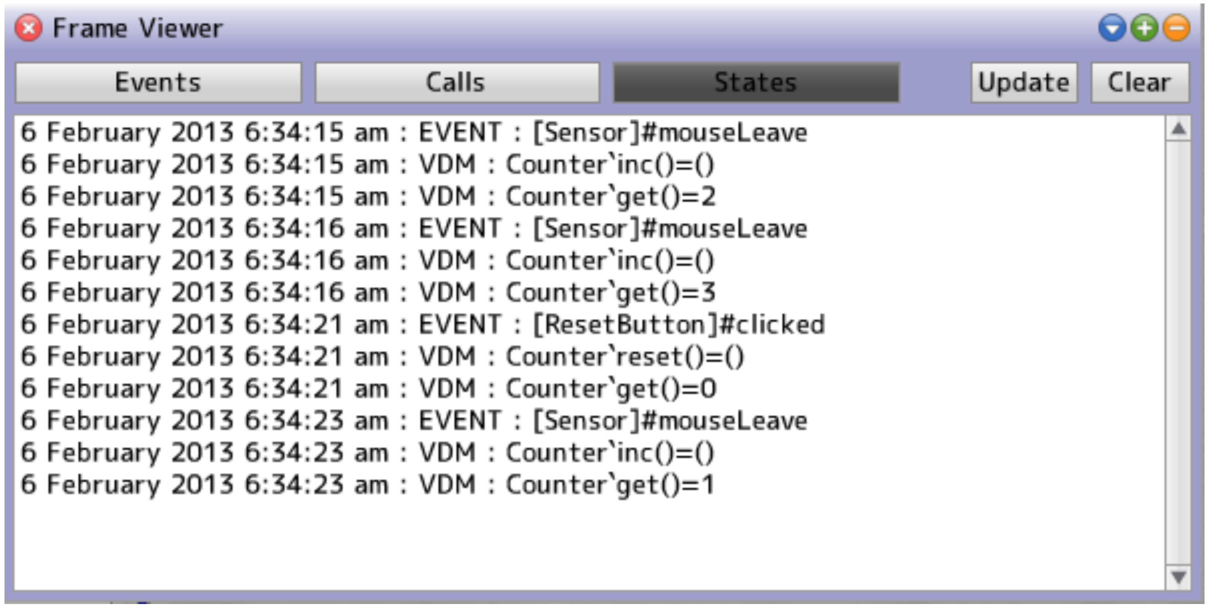
Middle Layer: Livetalk Browser and Event-Action Editor



```
recordStep
    Counter`inc()
    Counter`get() -> [NumSteps]
```

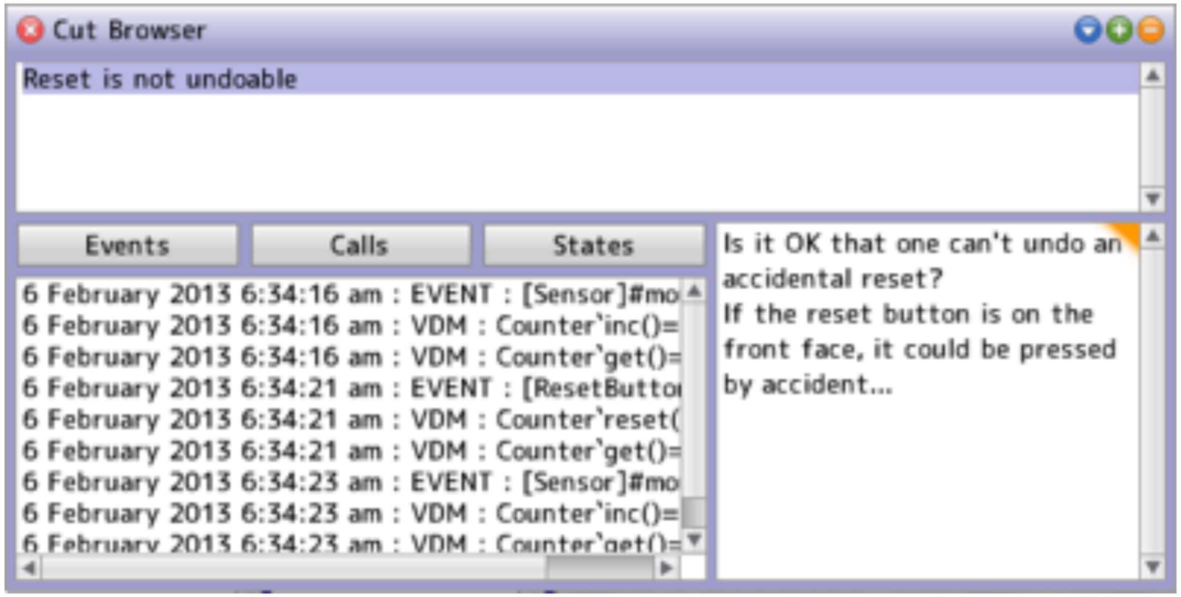
3 Tools for Discussion

Frame Viewer shows what's going on



```
6 February 2013 6:34:15 am : EVENT : [Sensor]#mouseLeave
6 February 2013 6:34:15 am : VDM : Counter`inc()=()
6 February 2013 6:34:15 am : VDM : Counter`get()=2
6 February 2013 6:34:16 am : EVENT : [Sensor]#mouseLeave
6 February 2013 6:34:16 am : VDM : Counter`inc()=()
6 February 2013 6:34:16 am : VDM : Counter`get()=3
6 February 2013 6:34:21 am : EVENT : [ResetButton]#clicked
6 February 2013 6:34:21 am : VDM : Counter`reset()=()
6 February 2013 6:34:21 am : VDM : Counter`get()=0
6 February 2013 6:34:23 am : EVENT : [Sensor]#mouseLeave
6 February 2013 6:34:23 am : VDM : Counter`inc()=()
6 February 2013 6:34:23 am : VDM : Counter`get()=1
```

Cut Viewer
to note and review discussion

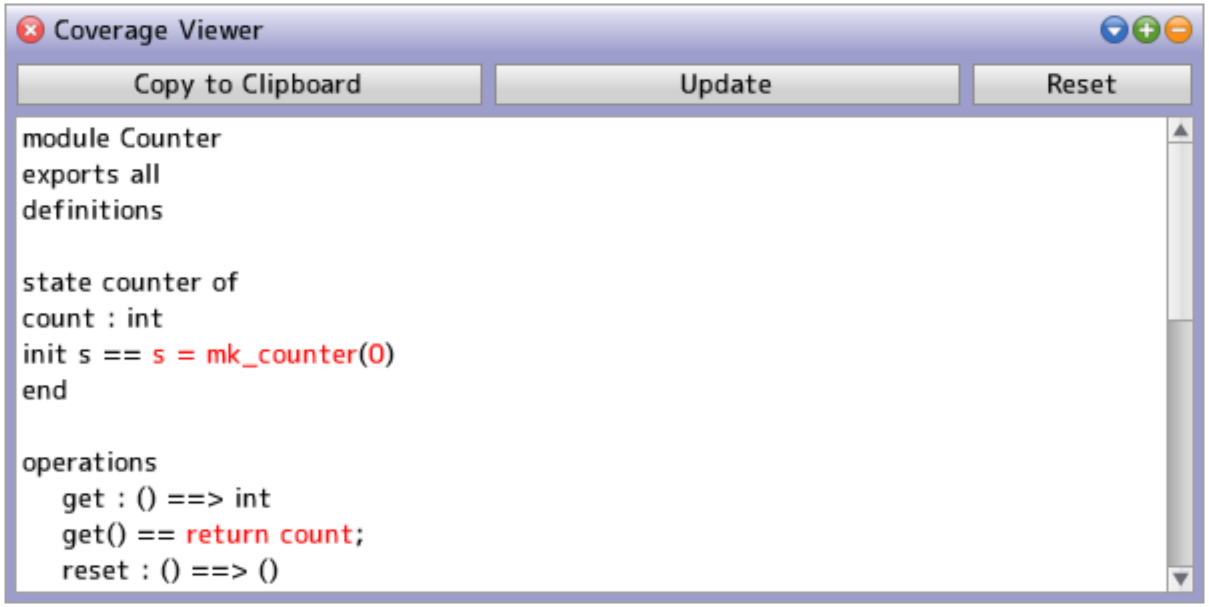


Reset is not undoable

```
6 February 2013 6:34:16 am : EVENT : [Sensor]#mo
6 February 2013 6:34:16 am : VDM : Counter`inc()=
6 February 2013 6:34:16 am : VDM : Counter`get()=
6 February 2013 6:34:21 am : EVENT : [ResetButto
6 February 2013 6:34:21 am : VDM : Counter`reset(
6 February 2013 6:34:21 am : VDM : Counter`get()=
6 February 2013 6:34:23 am : EVENT : [Sensor]#mo
6 February 2013 6:34:23 am : VDM : Counter`inc()=
6 February 2013 6:34:23 am : VDM : Counter`get()=
```

Is it OK that one can't undo an accidental reset?
If the reset button is on the front face, it could be pressed by accident...

Coverage Viewer shows what are unseen



```
module Counter
exports all
definitions

state counter of
count : int
init s == s = mk_counter(0)
end

operations
    get : () ==> int
    get() == return count;
    reset : () ==> ()
```

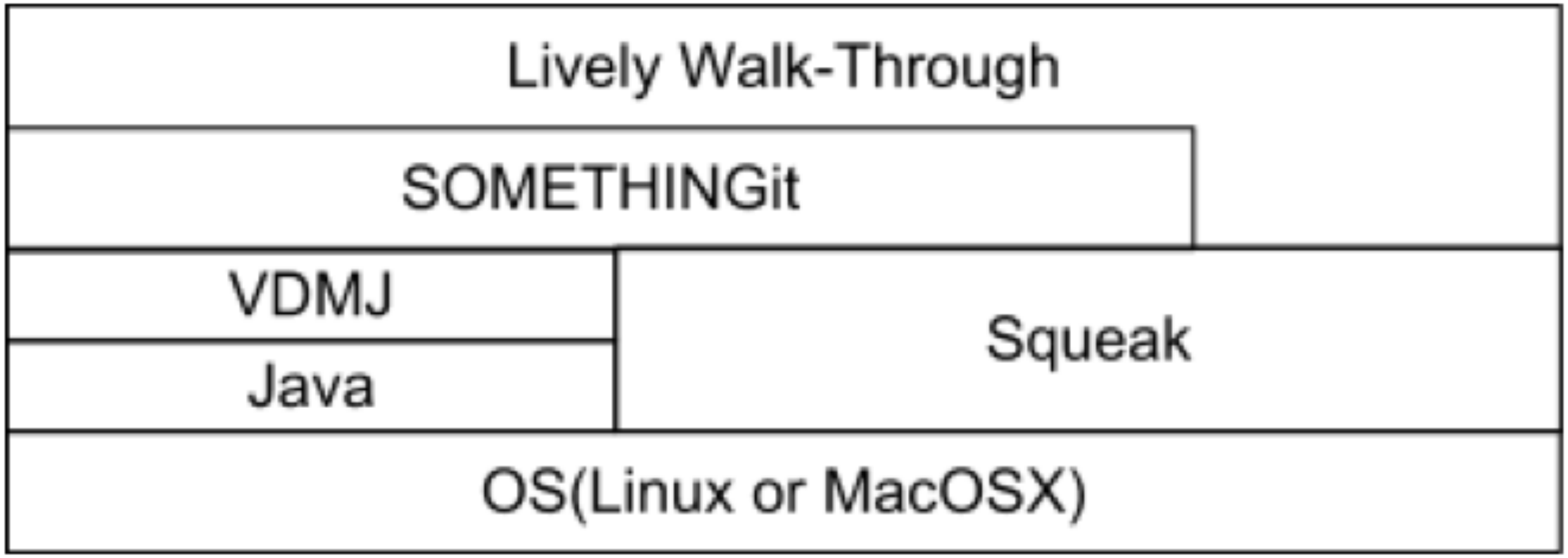
System Requirements

OS: Linux or MacOSX

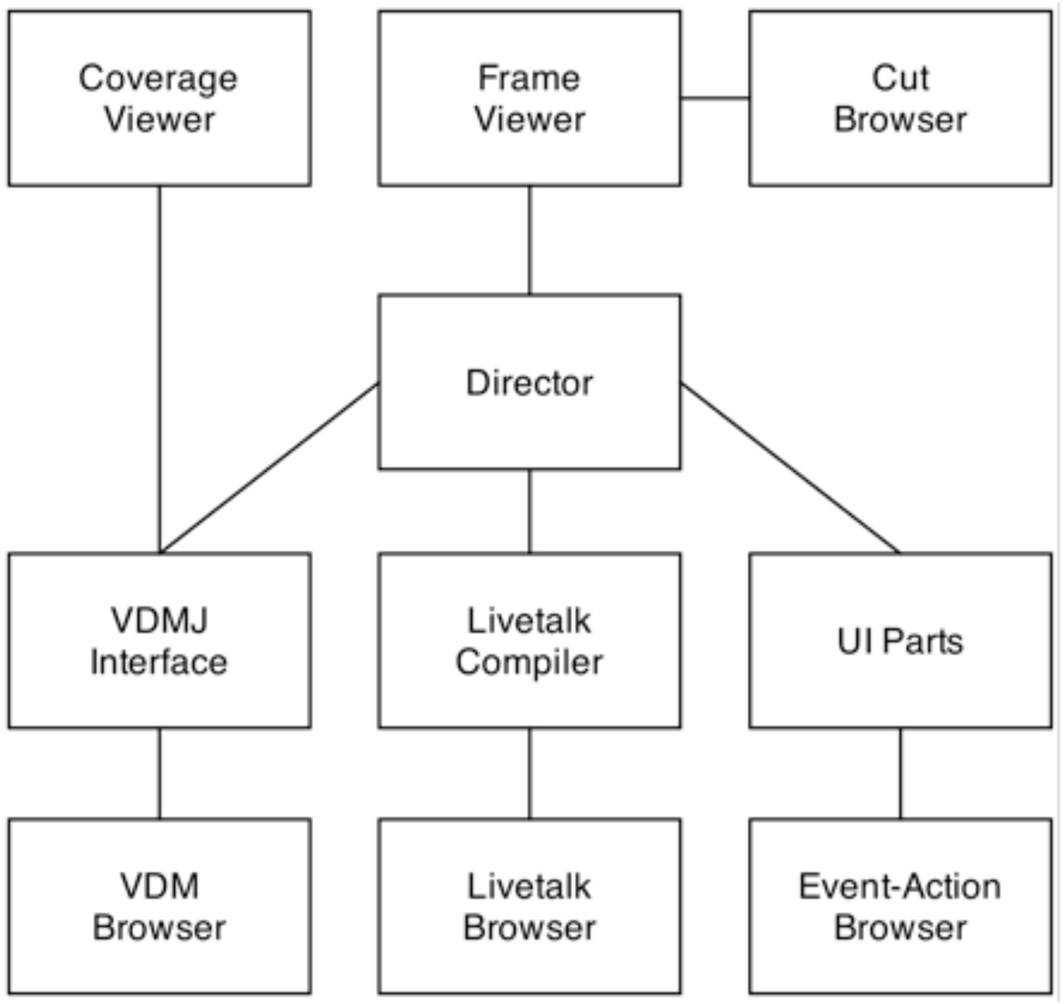
Smalltalk System: Squeak 4.3 or higher
VDM interpreter: VDMJ-2.0.1

Libraries: SOMETHINGit, OSProcess

Architecture



Major Components



Lightweight Formal Methods

Why Lightweight?

Formal specs in ~~other~~ MORE than specification phase...

- requirement analysis
  - type checking, animation
- design
  - reference, assertion
- test
  - test oracles, test cases

Why Lightweight?

Formal specs in ~~other~~ MORE than specification phase...

- requirement analysis
    - type checking, animation
- design
    - reference, assertion, unit test
- test
    - test oracles, test cases
- UI/UX design
    - why not?

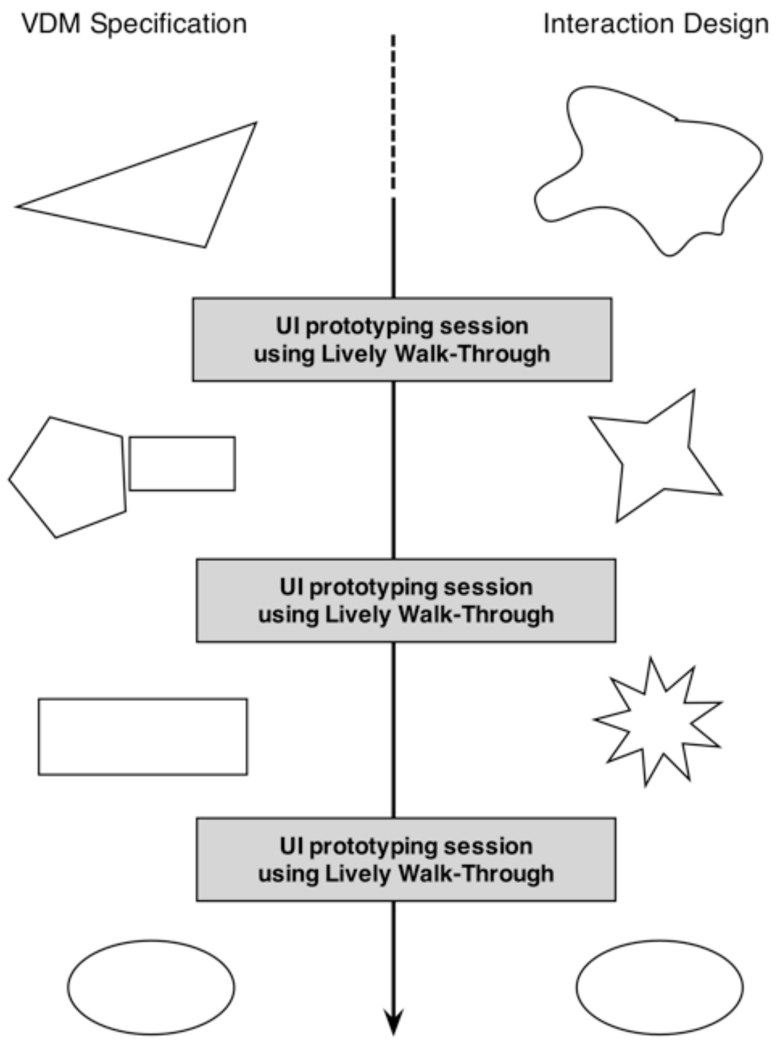
Two Worlds

Why collaborate?

UI design without computer science may "create" an unfeasible UI.

Functional modeling without interaction design may "construct" a stressful system.

UI Prototyping Cycles


VDM Specification
Interaction Design
UI prototyping session using Lively Walk-Through
UI prototyping session using Lively Walk-Through
UI prototyping session using Lively Walk-Through

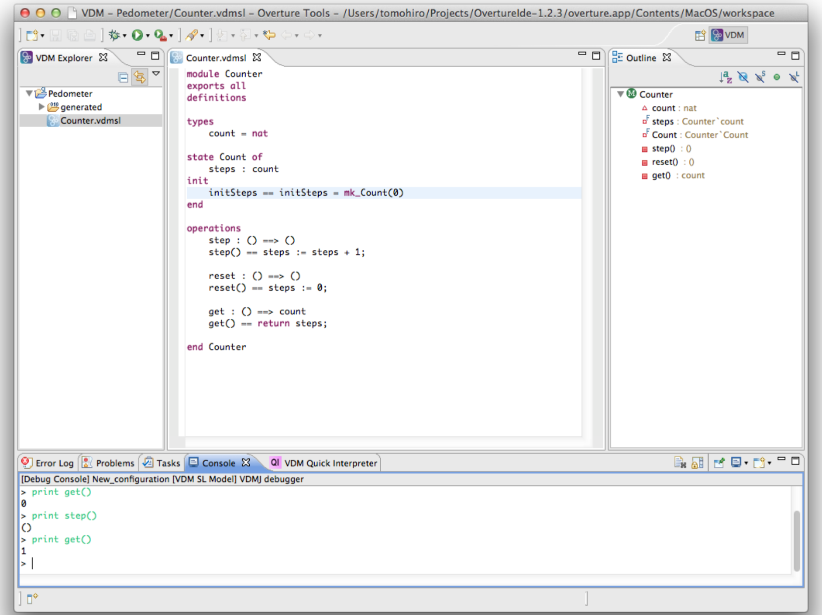
How to make this happen?

How to make this happen?

or

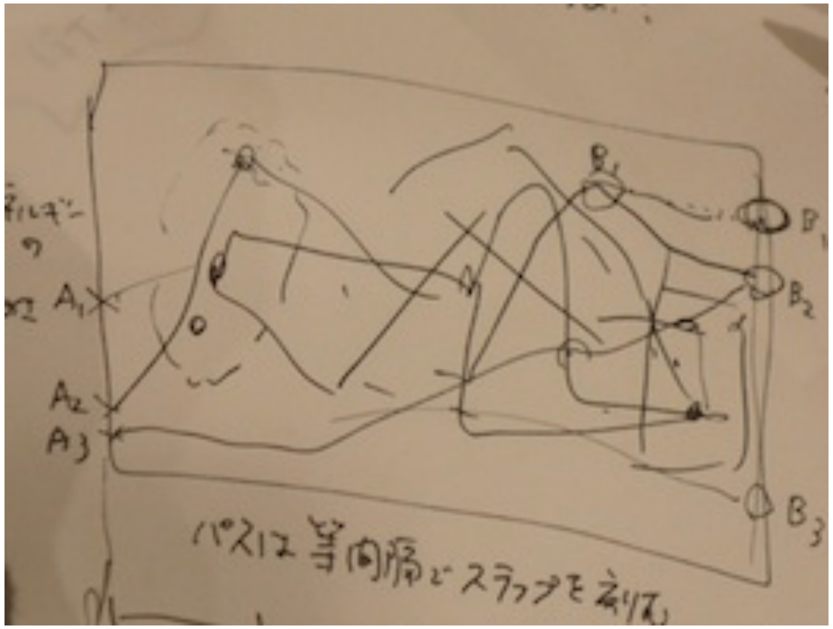
Why this does not happen often?

They are Different Animals

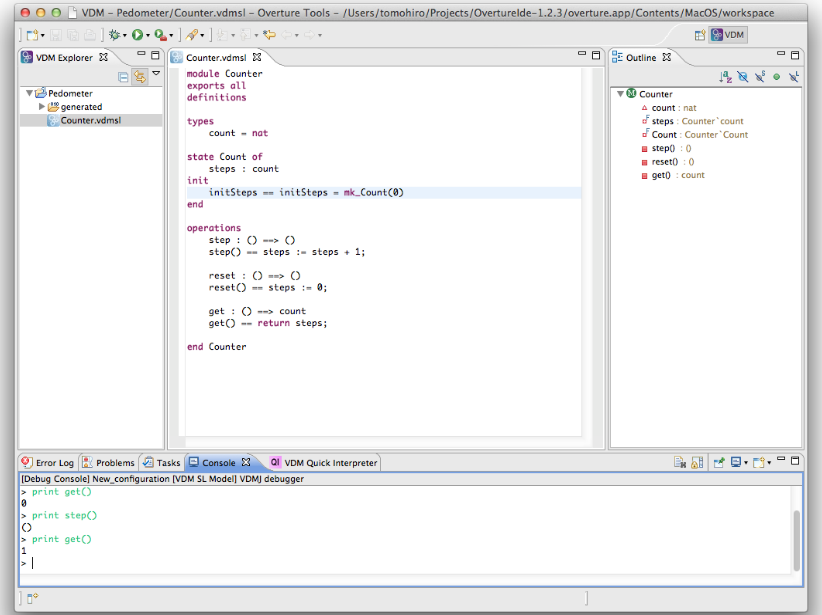


```
module Counter
exports all
definitions

types
    count = nat

state Count of
    steps : count
init
    initSteps == initSteps = mk_Count(0)
end

operations
    step : () ==> ()
    step() == steps := steps + 1;

    reset : () ==> ()
    reset() == steps := 0;

    get : () ==> count
    get() == return steps;

end Counter
```

```
[Debug Console] New_configuration [VDM SL Model] VDMJ debugger
> print get()
0
> print step()
()
> print get()
1
> |
```

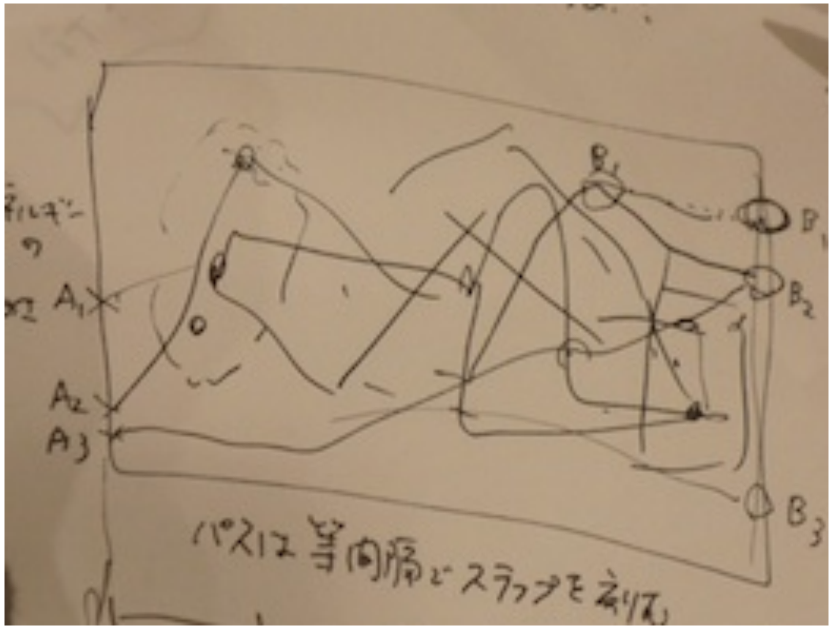


They are Different Animals



```
module Counter
exports all
definitions

types
    count = nat

state Count of
    steps : count
init
    initSteps == initSteps = mk_Count(0)
end

operations
    step : () ==> ()
    step() == steps := steps + 1;

    reset : () ==> ()
    reset() == steps := 0;

    get : () ==> count
    get() == return steps;

end Counter
```

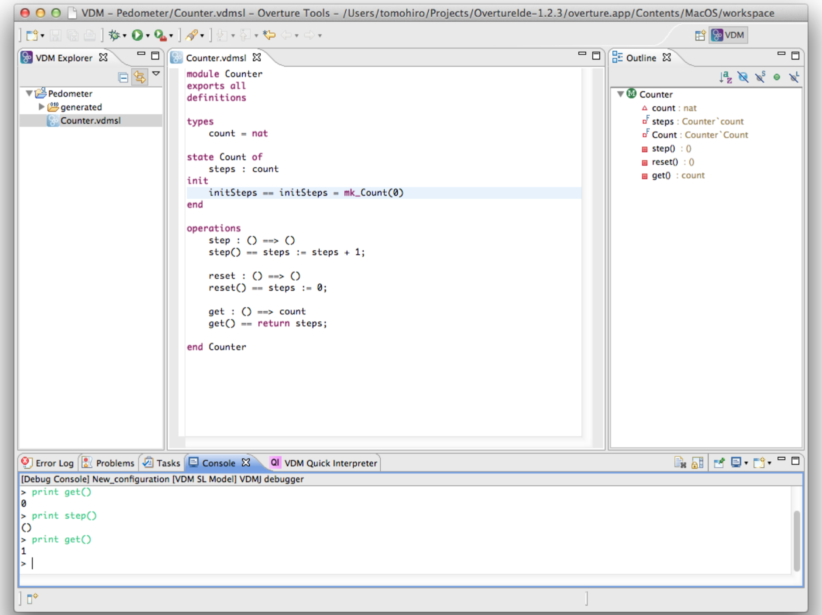
```
[Debug Console] New_configuration [VDM SL Model] VDMJ debugger
> print get()
0
> print step()
()
> print get()
1
> |
```



Formal specification　　UI/UX design

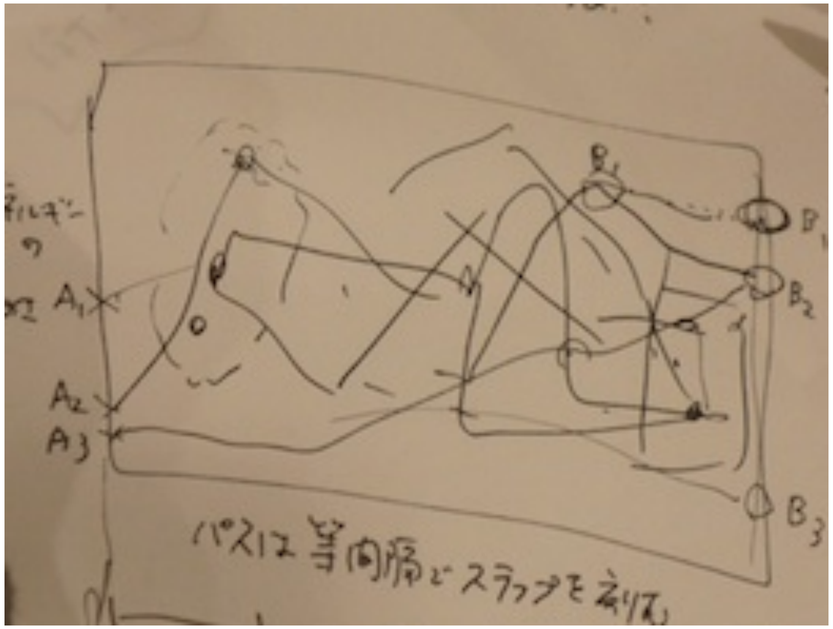
the world of MAKING　　the world of USING

They are Different Animals



```
module Counter
exports all
definitions

types
    count = nat

state Count of
    steps : count
init
    initSteps == initSteps = mk_Count(0)
end

operations
    step : () ==> ()
    step() == steps := steps + 1;

    reset : () ==> ()
    reset() == steps := 0;

    get : () ==> count
    get() == return steps;

end Counter
```

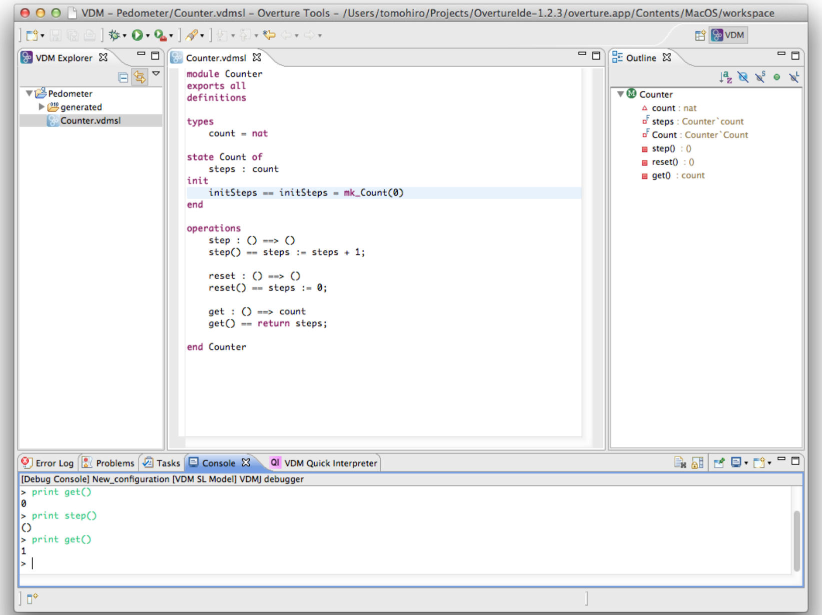
```
[Debug Console] New_configuration [VDM SL Model] VDMJ debugger
> print get()
0
> print step()
()
> print get()
1
> |
```



What is the system?

What the user interact with?

They are Different Animals



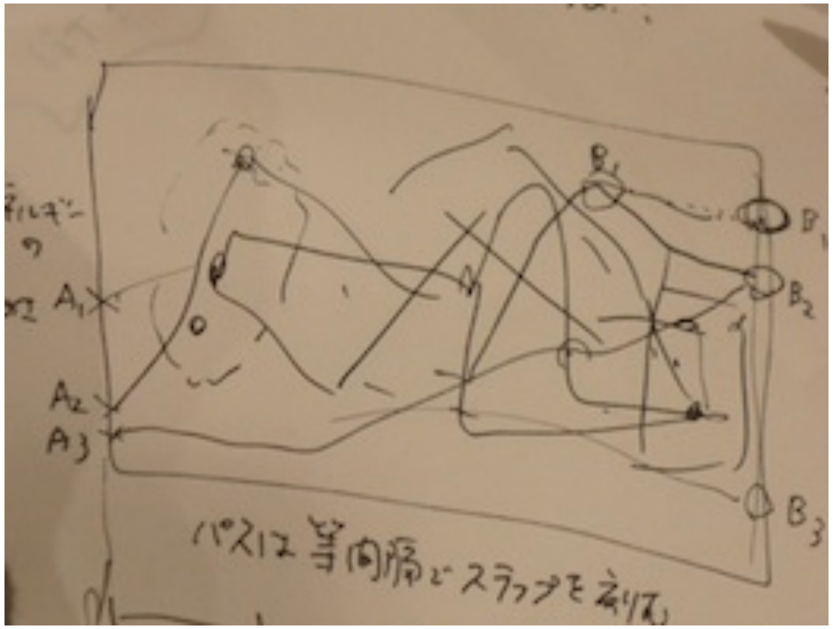
VDM – Pedometer/Counter.vdmsl – Overture Tools – /Users/tomohiro/Projects/OvertureIde-1.2.3/overture.app/Contents/MacOS/workspace

```
module Counter
exports all
definitions

types
    count = nat

state Count of
    steps : count
init
    initSteps == initSteps = mk_Count(0)
end

operations
    step : () ==> ()
    step() == steps := steps + 1;

    reset : () ==> ()
    reset() == steps := 0;

    get : () ==> count
    get() == return steps;

end Counter
```

```
Counter
  count : nat
  steps : Counter`count
  Count : Counter`Count
  step() : ()
  reset() : ()
  get() : count
```

[Debug Console] New_configuration [VDM SL Model] VDMJ debugger

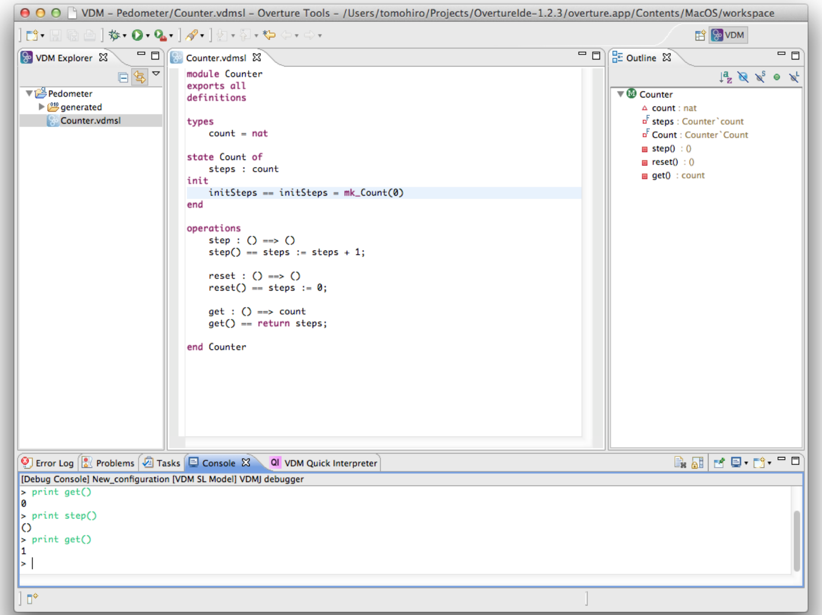
```
> print get()
0
> print step()
()
> print get()
1
> 
```



Logical soundness

Cognitive soundness

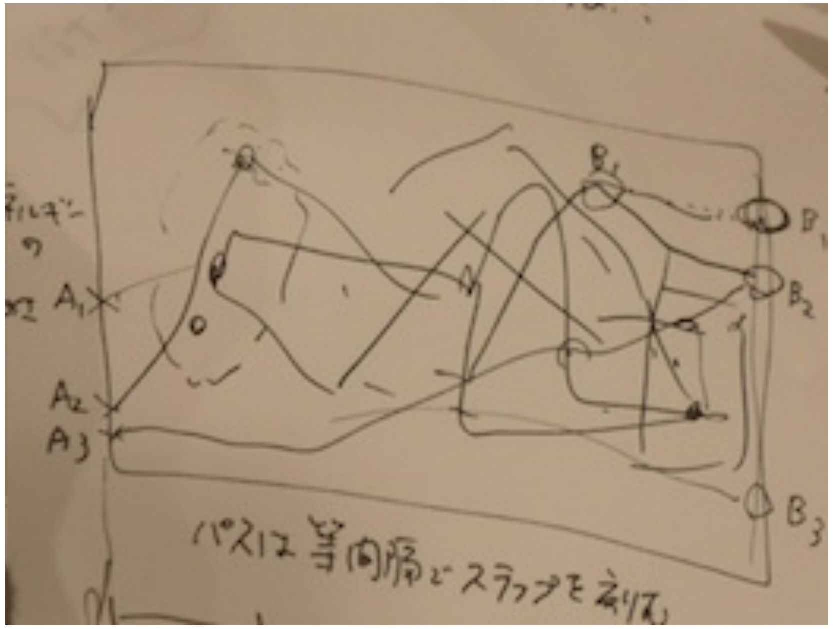
They are ~~Different~~ Similar Animals



VDM – Pedometer/Counter.vdmsl – Overture Tools – /Users/tomohiro/Projects/OvertureIde-1.2.3/overture.app/Contents/MacOS/workspace

```
module Counter
exports all
definitions

types
    count = nat

state Count of
    steps : count
init
    initSteps == initSteps = mk_Count(0)
end

operations
    step : () ==> ()
    step() == steps := steps + 1;

    reset : () ==> ()
    reset() == steps := 0;

    get : () ==> count
    get() == return steps;

end Counter
```

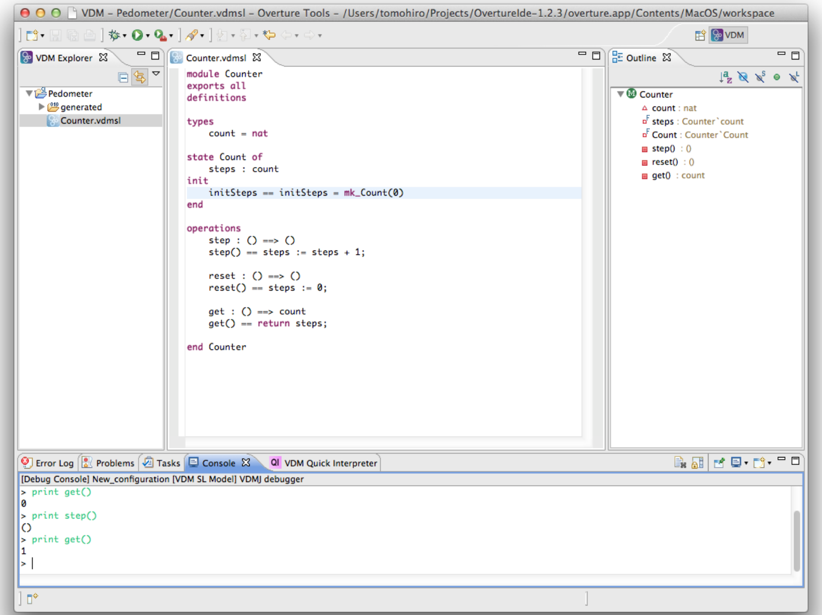
```
[Debug Console] New_configuration [VDM SL Model] VDMJ debugger
> print get()
0
> print step()
()
> print get()
1
> |
```



Understand by writing

Understand by sketching

They are ~~Different Animals~~ Friends



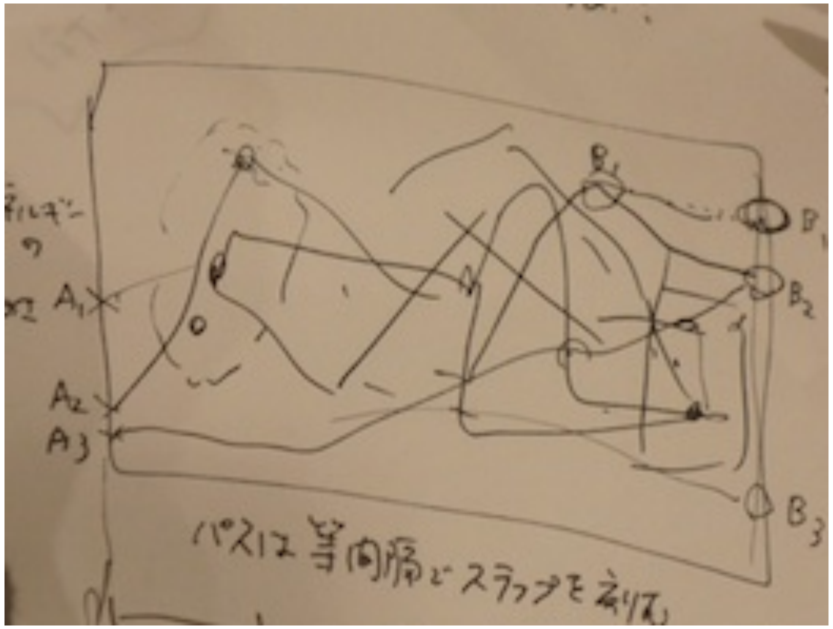
VDM – Pedometer/Counter.vdmsl – Overture Tools – /Users/tomohiro/Projects/OvertureIde-1.2.3/overture.app/Contents/MacOS/workspace

```
module Counter
exports all
definitions

types
    count = nat

state Count of
    steps : count
init
    initSteps == initSteps = mk_Count(0)
end

operations
    step : () ==> ()
    step() == steps := steps + 1;

    reset : () ==> ()
    reset() == steps := 0;

    get : () ==> count
    get() == return steps;

end Counter
```

```
Counter
  count : nat
  steps : Counter`count
  Count : Counter`Count
  step() : ()
  reset() : ()
  get() : count
```

[Debug Console] New_configuration [VDM SL Model] VDMJ debugger

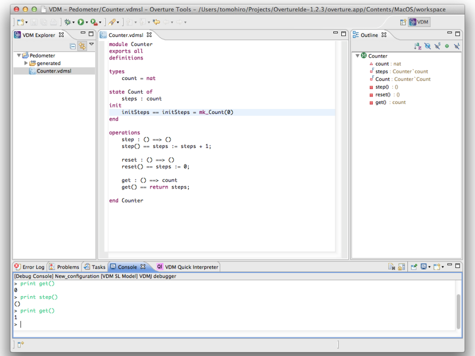
```
> print get()
0
> print step()
()
> print get()
1
> |
```



Animating the system makes
formal engineers and UI/UX designers
understand their design artifacts

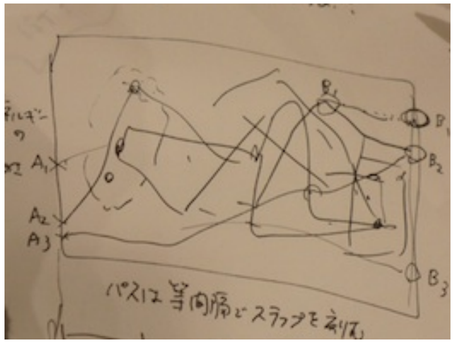
They are Good Friends

VDM spec gives a functional basis
VDM animation gives motion to sketches


VDM - Pedometer/Counter.vdmsl - Overture Tools - /Users/tomohiro/Projects/OvertureIde-1.2.3/overture.app/Contents/MacOS/workspace

```
module Counter
exports all
definitions

types
    count = nat

state Count of
    steps : count
init
    initSteps == initSteps = mk_Count(0)
end

operations
    step : () ==> ()
    step() == steps := steps + 1;

    reset : () ==> ()
    reset() == steps := 0;

    get : () ==> count
    get() == return steps;

end Counter
```

```
[Debug Console] New_configuration [VDM SL Model] VDMJ debugger
> print get()
0
> print step()
()
> print get()
1
> |
```



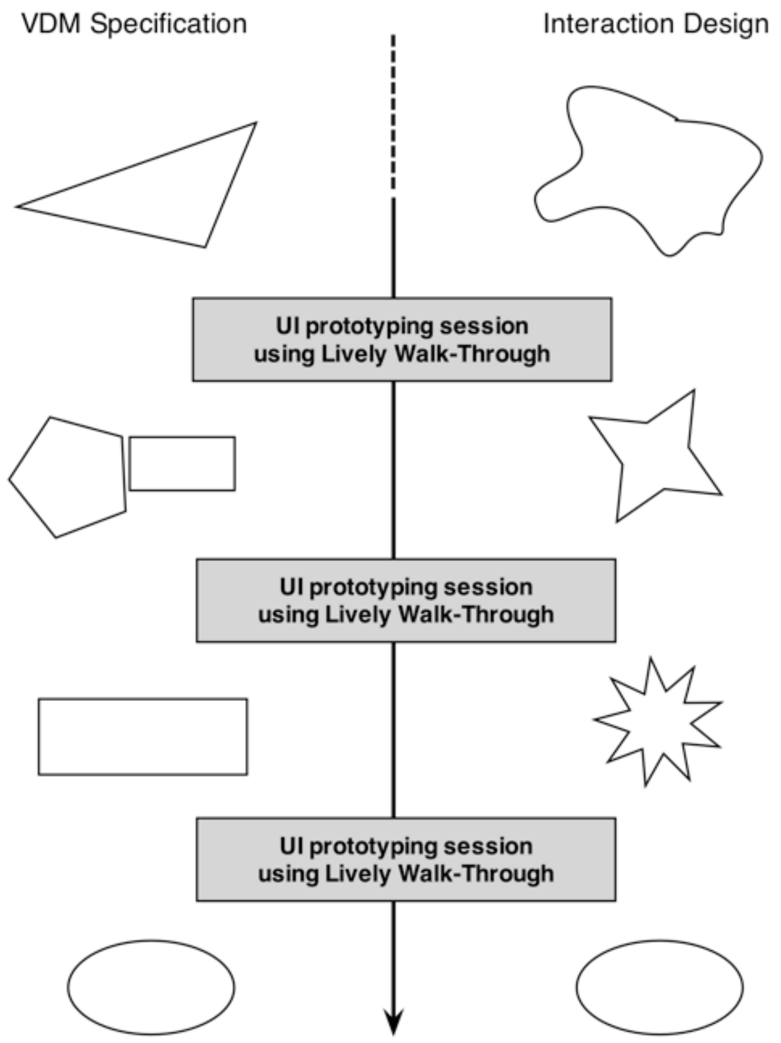
UI sketch gives a context of functions
UI animation gives user's perception

How to make this happen?

How to make this happen?

Animation

Animation And Discussion Drive UI Prototyping Cycles


VDM Specification
Interaction Design
UI prototyping session using Lively Walk-Through
UI prototyping session using Lively Walk-Through
UI prototyping session using Lively Walk-Through

Live Demo

Conclusion

- Lively Walk-Through bridges between functional modeling and UI/UX design
  - VDM animation gives motion to a UI sketch.
  - UI animation gives user's perception.

Future Work

- Image processing (animating a sketch)
- Support for post-session tasks
  - for VDM engineers
  - for UI designers