

A new Test Tool in SAP

Circuit Diagram for Code

SAP ABAP Test Seams

Moose2Model

CubeServ®

Referent: Rainer Winkler
Date: 11 September 2018

Unit Tests are fine

**If external dependencies
can be managed**

SAP ABAP Test Seams

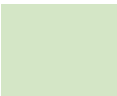
For Unit Tests
with external dependencies


Coding

```
CLASS cls IMPLEMENTATION.  
METHOD get_input.
```

```
cl_demo_input=>request( CHANGING field = input ).
```

```
ENDMETHOD.  
ENDCLASS.
```

 Syntax Check based on Coding

 Syntax Check based on Testcoding

Testcoding

```
CLASS tst IMPLEMENTATION.  
METHOD test_input.
```

```
TEST-INJECTION fake_input.
```

```
input = 'xxx'.
```

```
END-TEST-INJECTION .
```

```
DATA(input) = NEW cls( )->get_input( ).  
cl_abap_unit_assert=>assert_equals(  
EXPORTING  
exp = 'xxx'  
act = input ).
```

```
ENDMETHOD.  
ENDCLASS.
```

Test Insertions can be done multiple times

Typical application:

1. Insert a fail statement in the setup method of the test
2. Insert the test coding in the test

Works fine

When a few simple rules are followed in the insertion

1 – Implement inserted test coding correctly

2 – Monitor whether all Test Seams have a Test Injection

See

<https://blogs.sap.com/2018/06/08/abap-test-seam-for-unit-test-with-external-dependencies-personal-guideline/>
for complete list

No need to implement the Test Seam Pattern manually

Test and productive coding are well separated

Can be added to Legacy Code with minimal risk

Other techniques to handle dependencies can be better

If they are used

If they are worth the additional effort

Before Test Seams where available
I wrote Tests only sometimes

With Test Seams, I do it much more
often

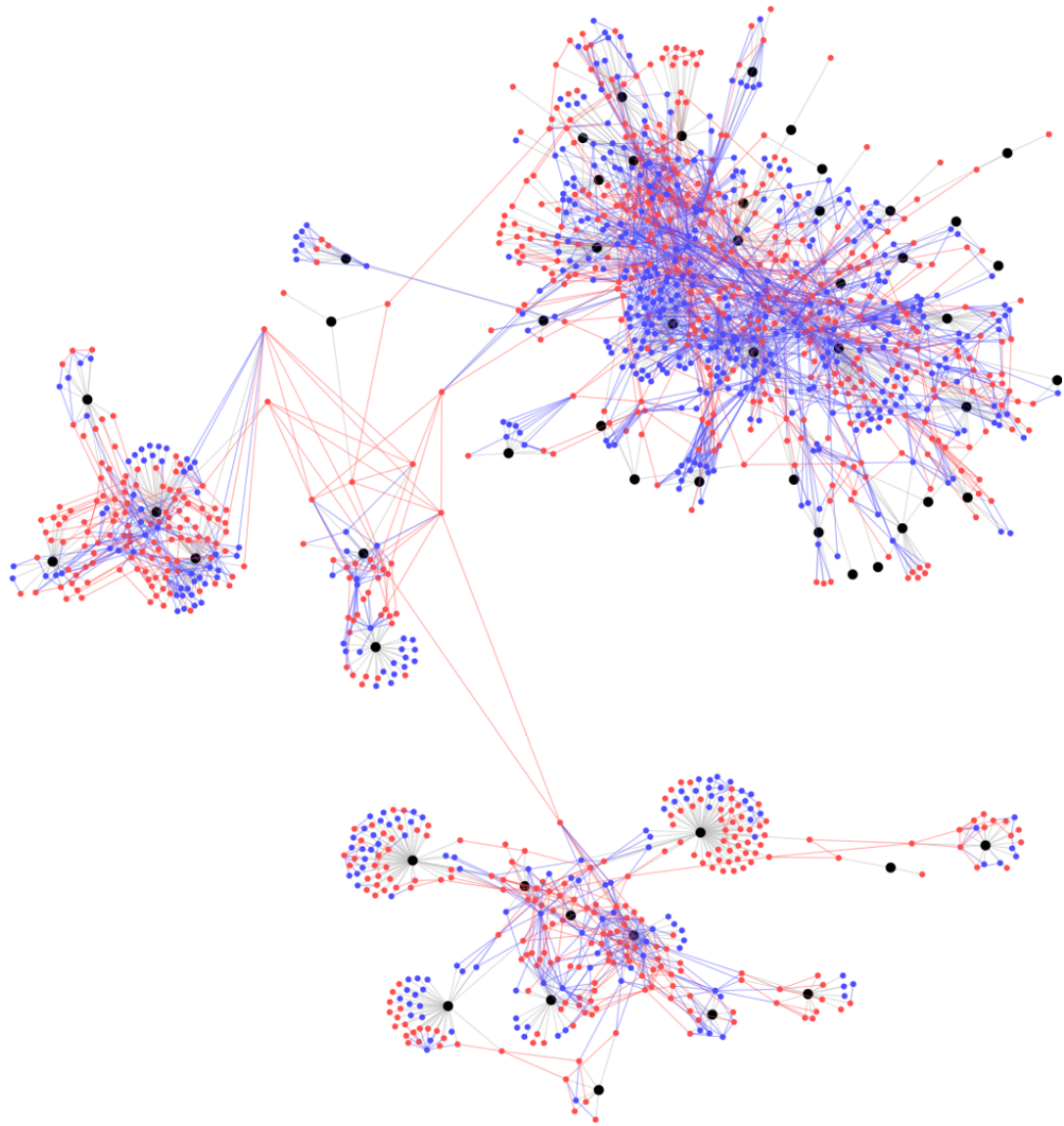
What other computer languages provide Test Seams as part of the language?

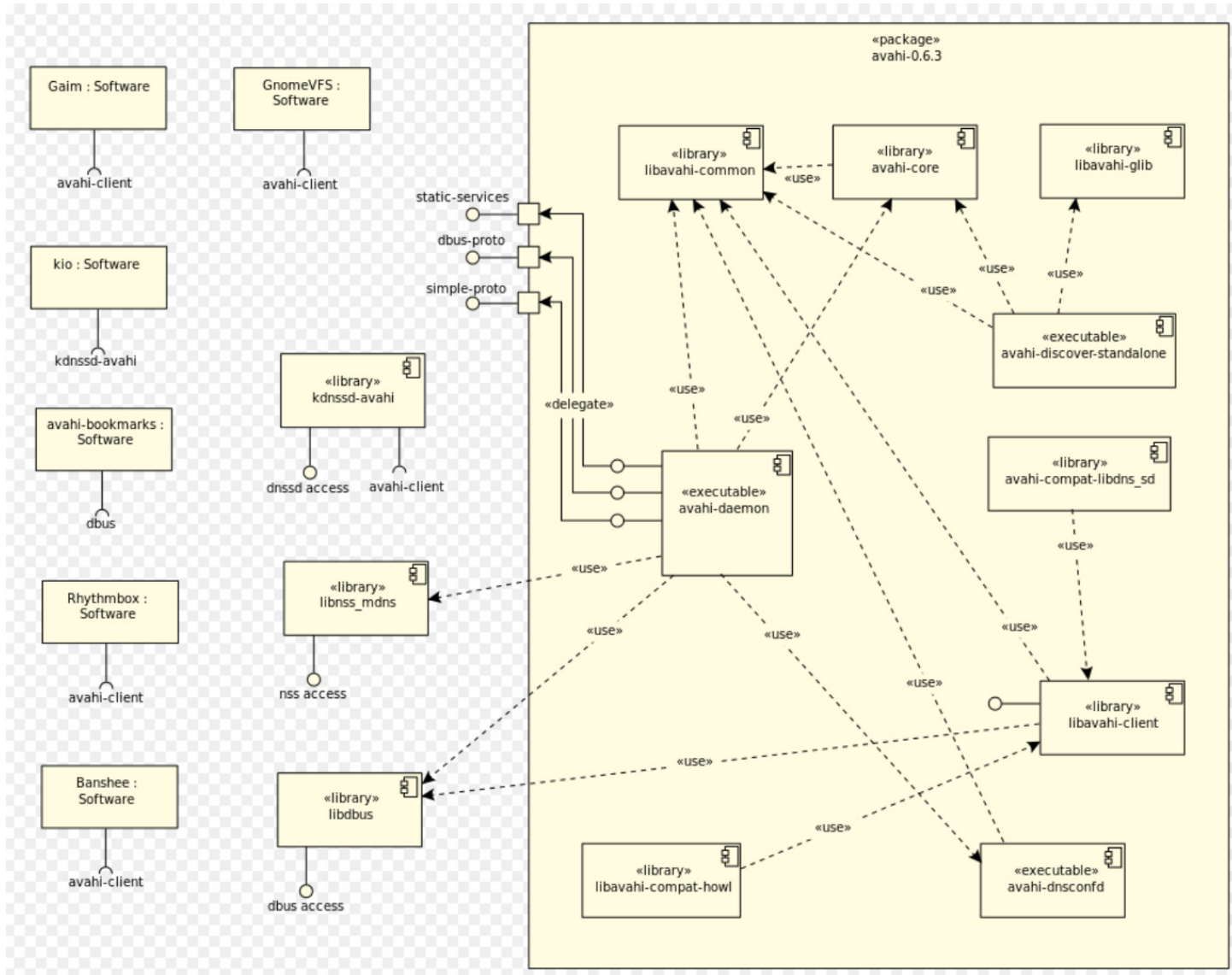
Do we need this in Smalltalk?

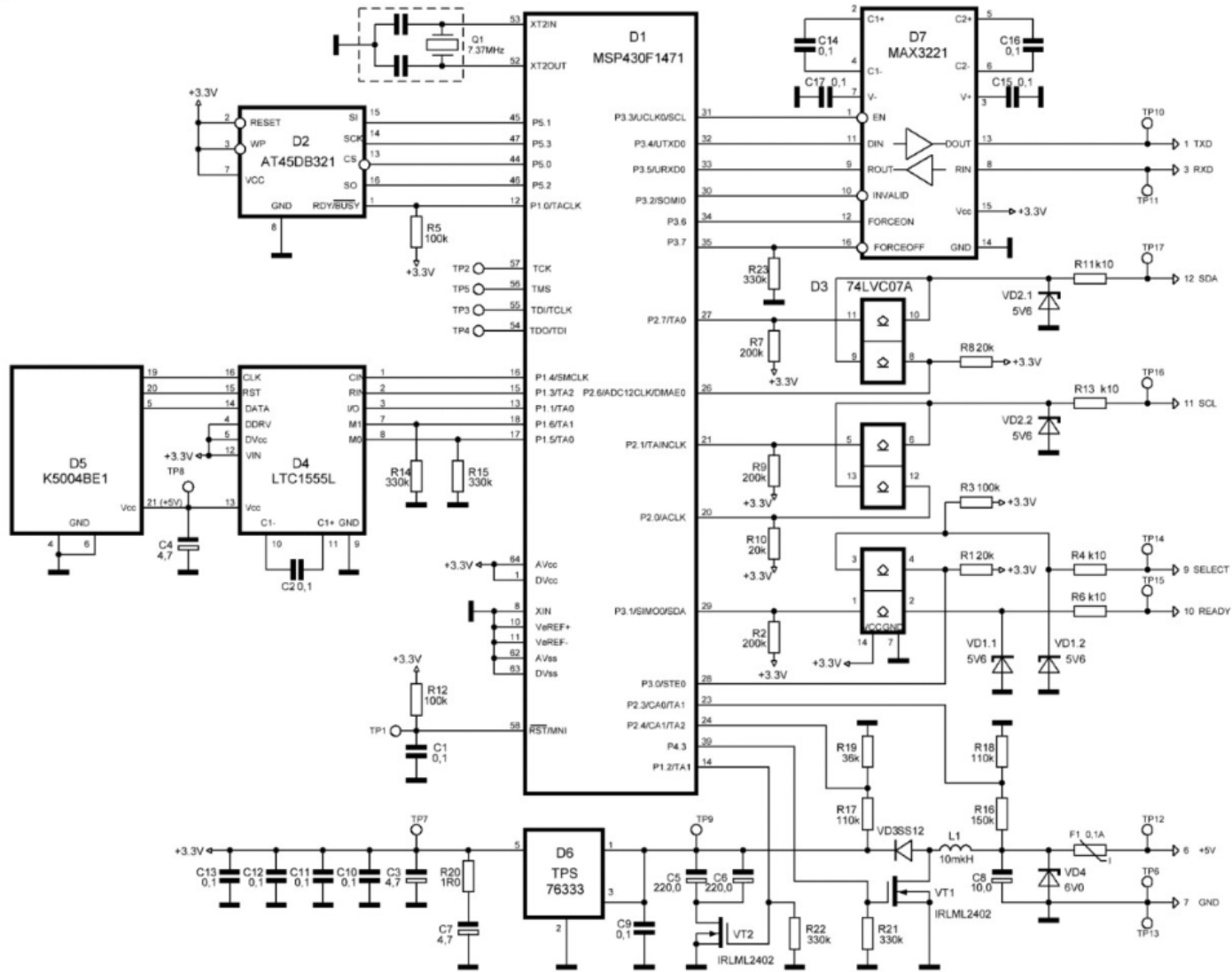
Search for Test Seam and SAP or
ABAP on the internet for more
informations

2 Circuit diagrams for software

Moose2Model





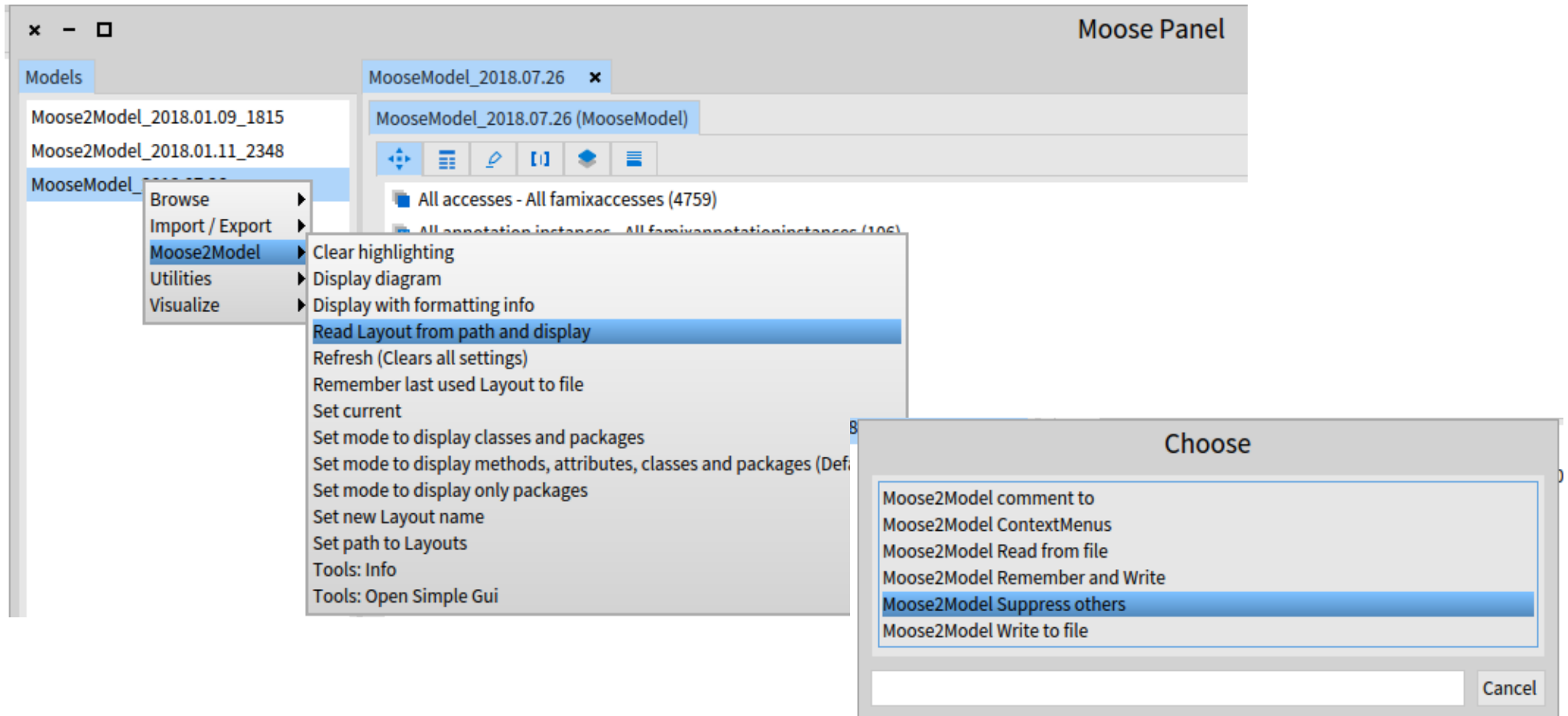


Can this help developers?

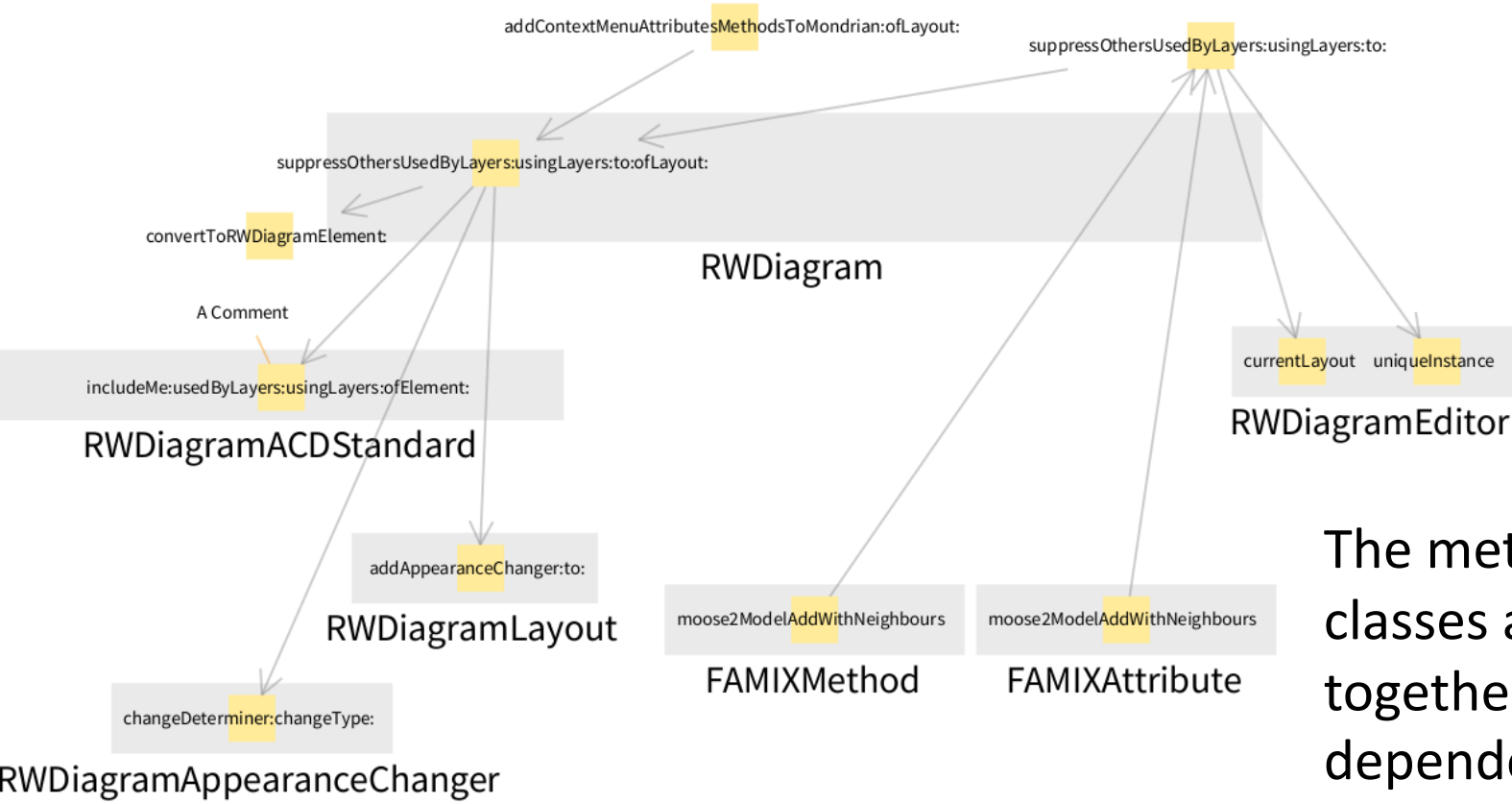
Yes - with tools
that automatize
creation and updating

Moose2Model - Display existing diagram

Context Menu of Moose Model -> Read Layout from Path



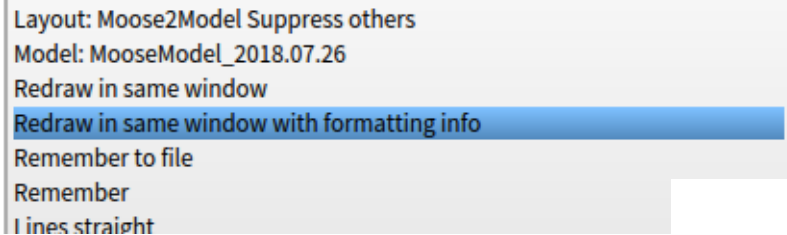
I worked on the "Suppress Other Logic" yesterday, which elements where to be regarded?



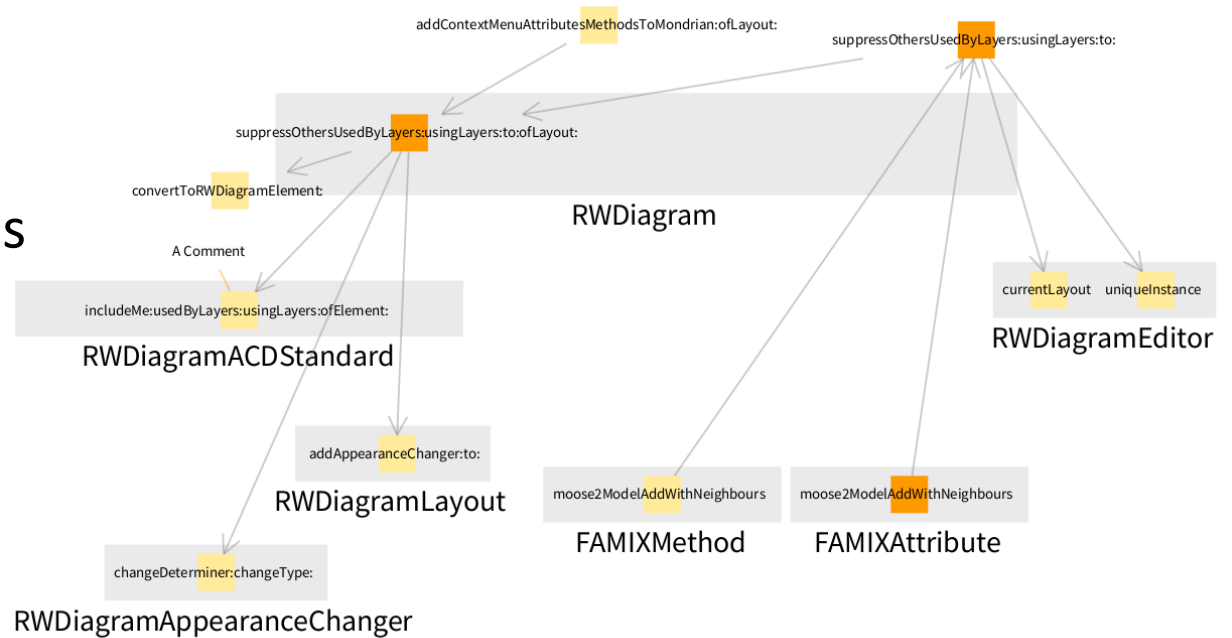
The methods and classes are displayed together with there dependencies

What is displayed?

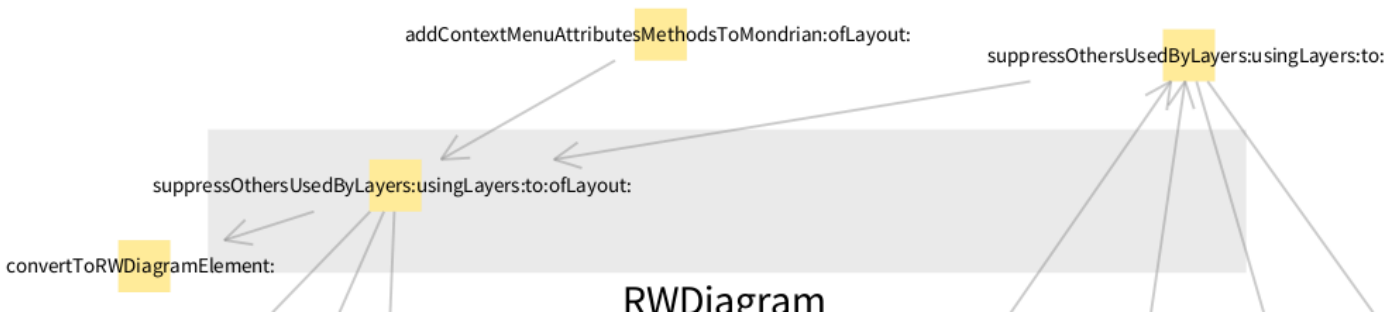
Right click on desktop -> Redraw ... with formatting info



Highlighted elements are always displayed (as long as they are in the extraction)
Other elements only when they exist



Customize



Rearrange elements - will be stored to file

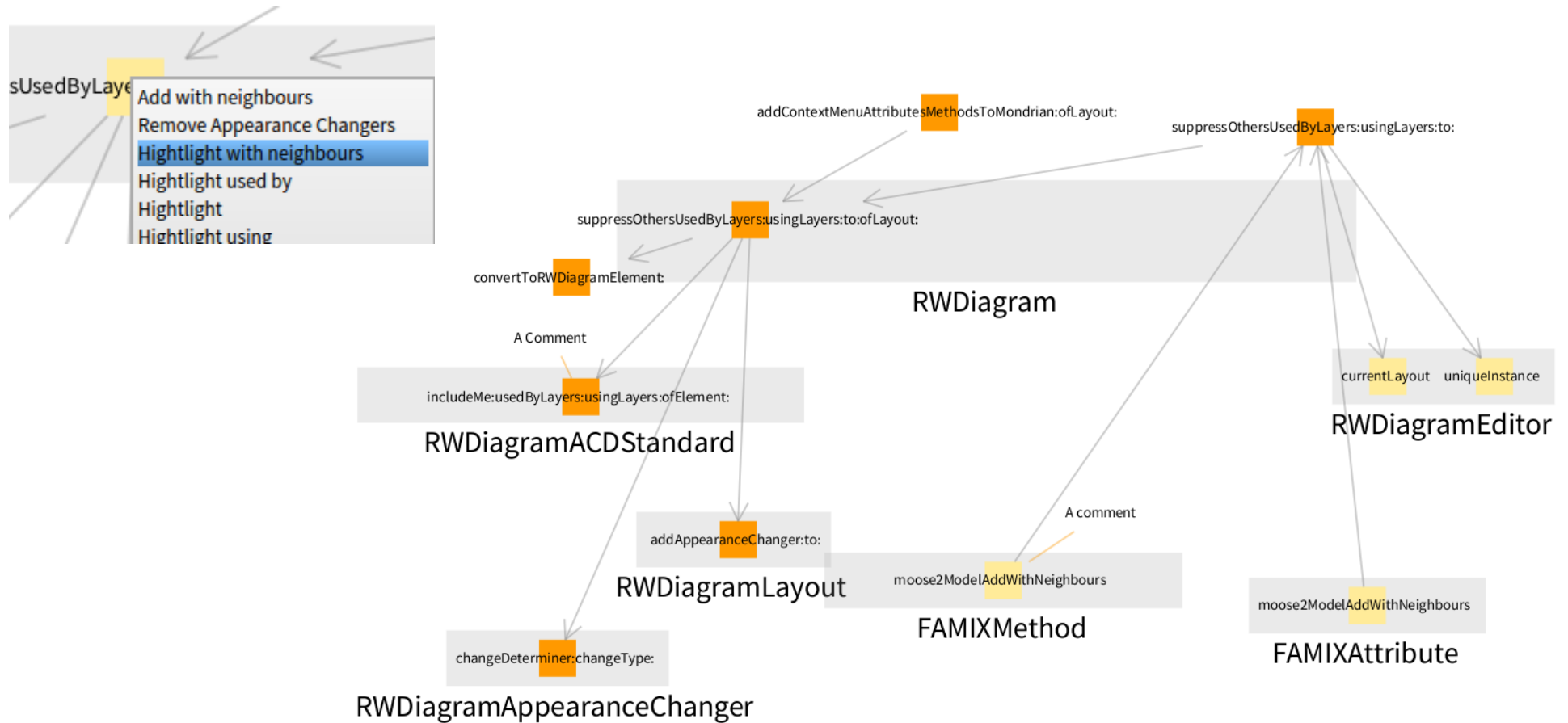


Add comments

The image illustrates the process of adding a comment to a FAMIXMethod element in a software model through four sequential screenshots:

- Top Left:** A context menu is open over a FAMIXMethod element labeled "moose2ModelAddWithNeighbours". The "Comment" option is highlighted in blue.
- Top Right:** A dialog box titled "Specify a new comment (use
 to specify a line break)" is displayed. The input field contains the text "A comment".
- Bottom Left:** A context menu is open over a FAMIXMethod element labeled "moose2ModelAddWithNeighbours". The "Remember to file" option is highlighted in blue. A comment "A comment" is visible next to the element.
- Bottom Right:** The final state shows the FAMIXMethod element labeled "moose2ModelAddWithNeighbours" with the comment "A comment" successfully added next to it.

Highlight neighbours (Helps in big diagrams)



Explore

addAppearanceChanger

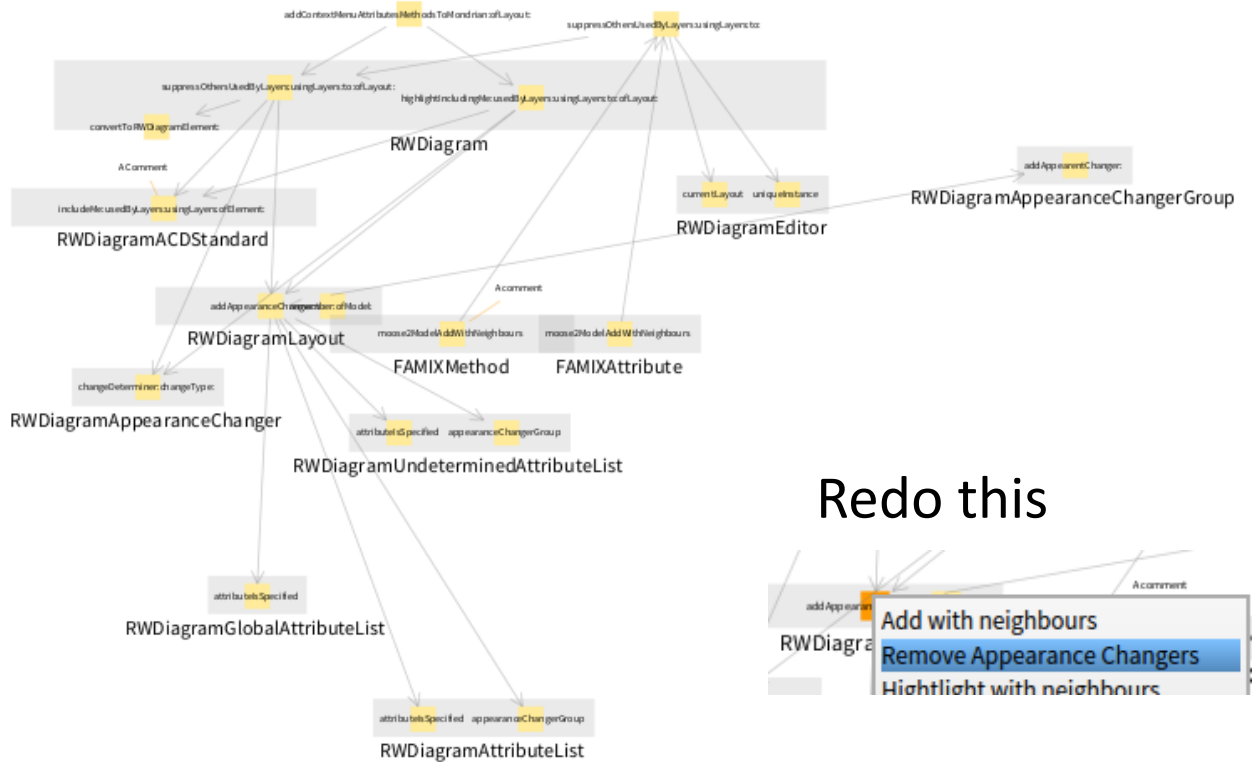
- Add with neighbours
- Remove Appearance Changers
- Highlight with neighbours
- Highlight used by

RWDiagram

Layout: Moose2Model Suppress others
Model: MooseModel_2018.07.26

- Redraw in same window
- Redraw in same window with formatting info
- Remember to file

More elements are shown



Redo this

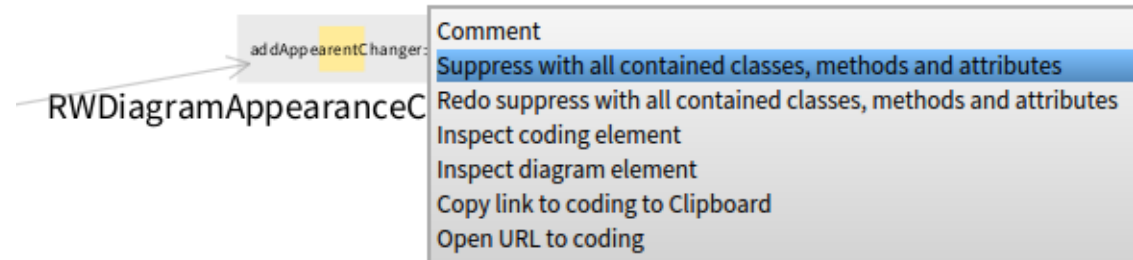
addAppearanceChanger

- Add with neighbours
- Remove Appearance Changers
- Highlight with neighbours

RWDiagram

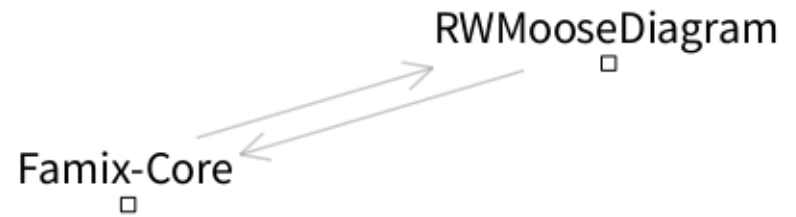
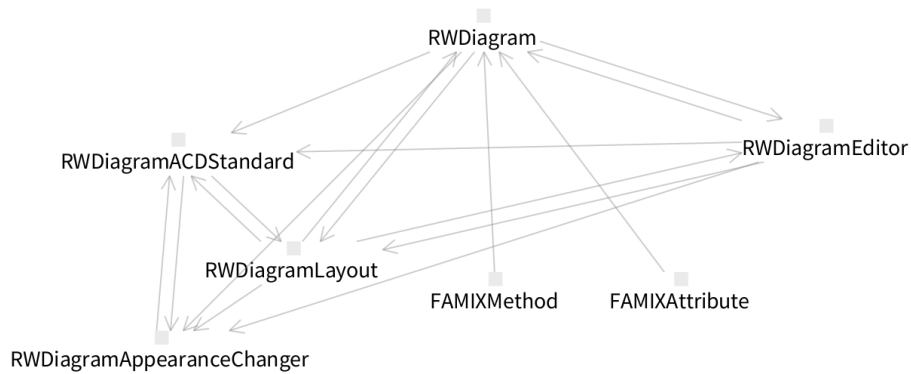
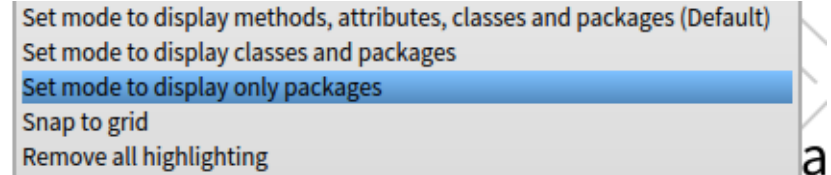
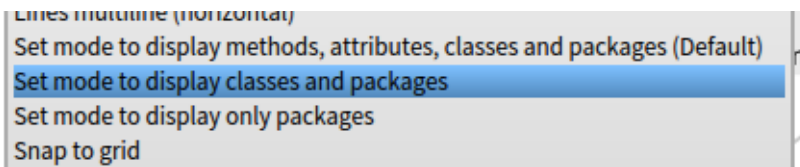
Simplify

(Exclude some elements, for instance for logging...)



Architecture Diagrams

Show only groupings (class/packages)



Diagrams can be made fast

Diagram generation is relaxing (automatization)

Keeping diagrams correct is relaxing (automatized)

Diagrams are (always) correct

Reduces cognitive load during coding

www.moose2model.org

MIT License

<https://youtu.be/k8RkDwlXKmg>

Works currently for **SAP** and **Smalltalk**,
could work for all languages where an **extractor to Moose** exists