



MetaCase

# Applying TDD for Creating DSM solutions: demo

Juha-Pekka Tolvanen

30 October, 2016

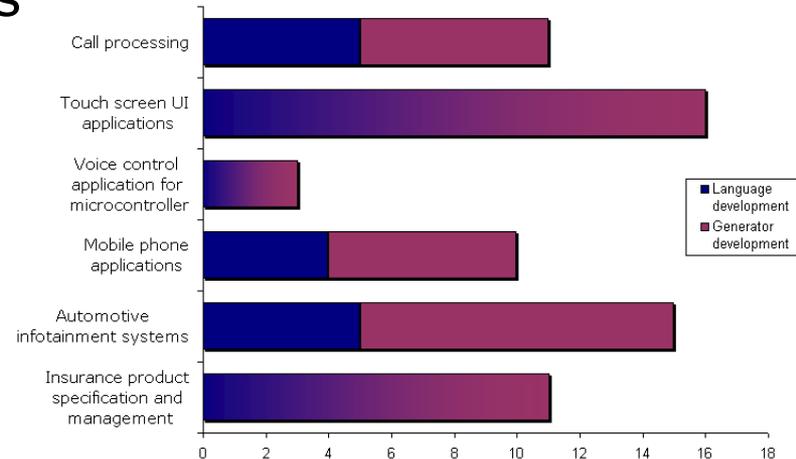
DSM Workshop, Amsterdam

# Content

- Motivation
- Idea on applying TDD for DSM
- Demonstration
- Discussion

# Motivation

- Better ensure that DSM solution (= language and generator) works as expected
  - When created
  - When modified while the domain evolves
- Our special focus is on generators as their development takes more time than languages
- Various approaches have been presented to test languages and generators
  - We present one considered cost-effective



# Idea

- Apply TDD for DSM creation and maintenance as follows:
  1. Create a library of small test models along with related output (in a format preferred)
  2. Build the DSM solution in small steps starting from the simplest test model of the library and then extend to others  
**Step = language element(s) + its test model + its generator + expected output**
  3. Test DSM solution after each step of adding/removing/changing metamodel or generator
  4. Run all previous tests for regression testing
  5. Automate the process

# Domain for demo: Apps for IoT device

## ■ Sensors

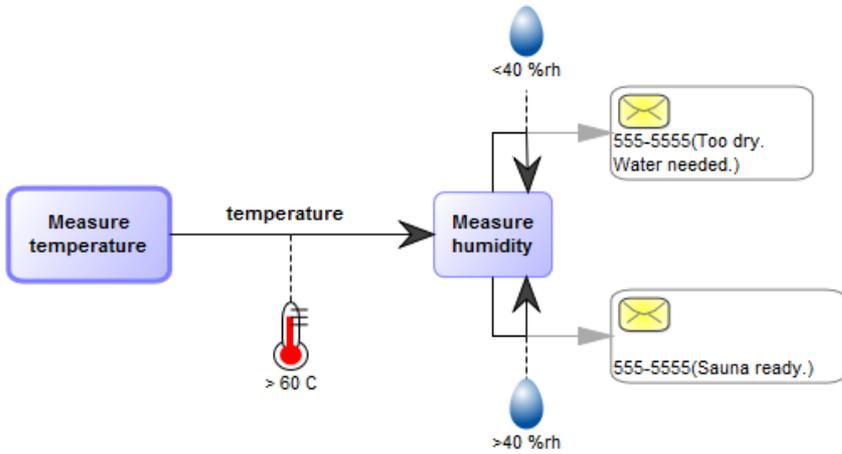
- altitude, movement (in 3 directions), humidity, location, luminance, pressure, speed, temperature, time, geofence, battery status...)

## ■ Actions

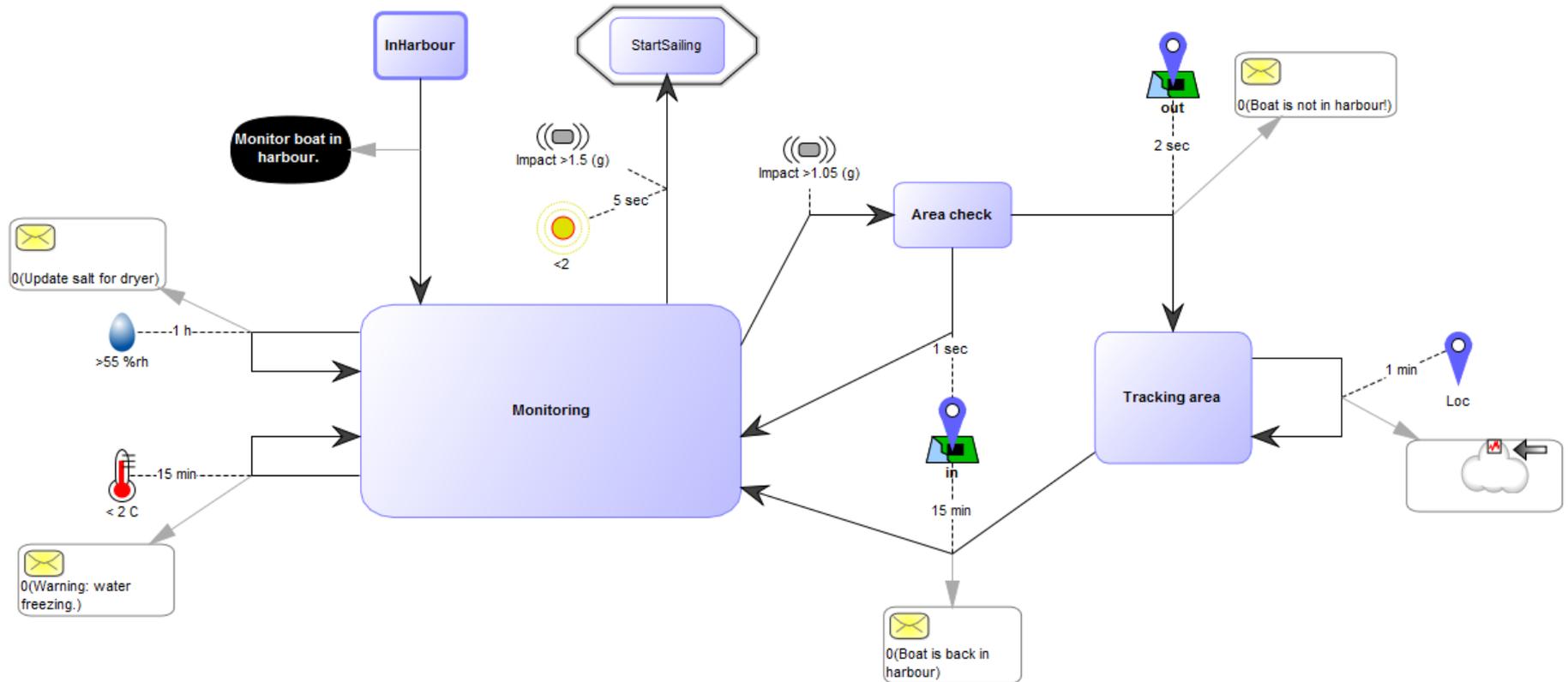
- sending SMS, send to cloud, push notification, saving logs



# Sample: Sauna App

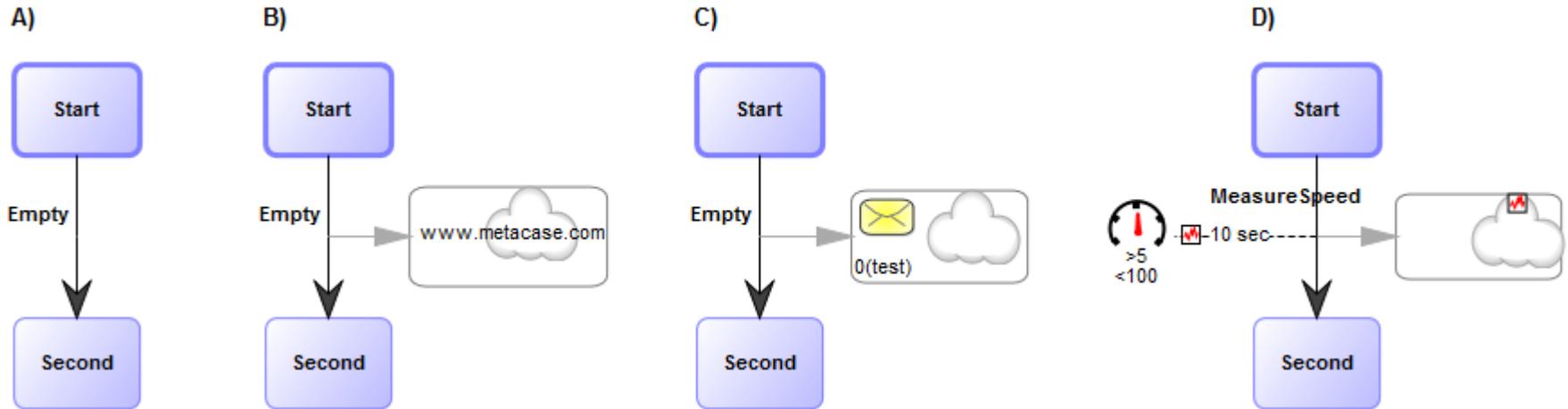


# Sample 2: Boat monitor



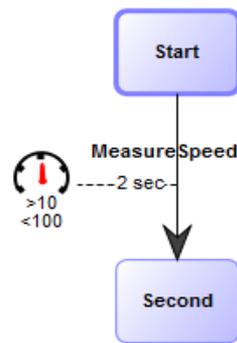
# Some test models to start with

- 4 out of 37 in total



# Test model and expected code

- Model with 3 elements and 1 connection => 50+ LOC
- Automated test checking that generated = expected
- Test individual tests or all tests from the library



```
{
  "pid": "1",
  "apiVersion": "00.18",
  "initPuId": 1,
  "purposes": [
    {
      "puId": 1,
      "name": "02 OneTrigger",
      "initStId": 0,
      "states": [
        {
          "stId": 0,
          "name": "Start",
          "events": [
            {
              "evId": 0,
              "name": "MeasureSpeed",
              "actions": {
                "engine": {
                  "gotoStId": 1
                }
              },
              "cloud": {
                "sendEvent": false,
                "sendPush": false
              }
            }
          ],
          "causes": [
            {
              "sId": "0x00020100",
              "threshold": {
                "count": 1
              },
              "measurement": {
                "log": false,
                "interval": 2000
              },
              "thresholds": {
                "isGt": 2.77778,
                "isLt": 29.7778
              }
            }
          ]
        }
      ]
    }
  ],
  {
    "stId": 1,
    "name": "Second",
    "events": []
  }
}
}
```

# Test library at the end

The screenshot displays the Thingssee Profile application interface. The main area is a grid of test models, each represented by an octagonal icon with a unique ID and name. The grid is organized into rows and columns, with some cells containing multiple icons. The icons include various symbols such as clocks, location pins, and battery indicators, representing different test scenarios.

**Test Library Grid:**

ID	Name
0	One state
00	BasicTransition
01	OneAction
01a	OneEmptyAction
01b	OneActionLogsToCloud
01c	OneActionSendsURL
01d	OneActionSendsPush
01e	OneActionSendsSMS
01f	TwoActions
02	OneTrigger
03	OneActionAndTrigger
04	OneActionAnd2Triggers
05	TimerTrigger
06	CalendarTrigger
07	Impact
08a	Longitudinal acceleration
08b	Lateral acceleration
08c	Vertical acceleration
08d	Longitudinal and Lateral acceleration
08e	Lateral and Vertical acceleration
09	Temperature
10	Humidity
11	Pressure
12	Luminance
13	Battery
13b	Battery any value
14a	Charger disconnected
14b	Charger any value
14c	Charger connected
14d	Charger and Charger
15	Battery and Charger
16	Location
16a	Acc Location
16b	Alt Location
16c	Geo Location
17	Display
18	Display with action
19	Display with trigger

**Left Panel (Tree View):**

- Purpose
  - 0 One state
    - 00 BasicTransition
    - 01 OneAction
      - 01a OneEmptyAction
      - 01b OneActionLogsToCloud
      - 01c OneActionSendsURL
      - 01d OneActionSendsPush
      - 01e OneActionSendsSMS
      - 01f TwoActions
    - 02 OneTrigger
      - 03 OneActionAndTrigger
      - 04 OneActionAnd2Triggers
    - 05 TimerTrigger
    - 06 CalendarTrigger
    - 07 Impact
    - 08a Longitudinal acceleration
    - 08b Lateral acceleration
    - 08c Vertical acceleration
    - 08d Longitudinal and Lateral acceleration
    - 08e Lateral and Vertical acceleration
    - 09 Temperature
    - 10 Humidity
    - 11 Pressure
    - 12 Luminance
    - 13 Battery
      - 13b Battery any value
    - 14a Charger disconnected
    - 14b Charger any value
    - 14c Charger connected
    - 14d Charger and Charger
    - 15 Battery and Charger
  - 16 Location

**Bottom Panel (Property Table):**

Property	Value
Graph type	Thingssee Profile
Name	Test-Driven Apps
Description	Test models: part

**Status Bar:** Active: None | Grid: 40 @ 40 | Snap  Show  100%

# Demo

# Experiences on applying TDD for DSM

- Makes DSL development "systematic"/phased
  - Supports planning and estimating on the DSM development effort
- Provides regression testing giving conformance that nothing breaks when language & generator changed
  - Is not the only testing approach as also existing models (and generated code) can be applied as test data
- Can be extended to cover results generated with the platform/components/etc.
  - i.e. integrate resulting code with the rest and test together
- Considered applicable in practice (by MetaEdit+ users)

# Remarks

- This approach is been applied in industrial projects when the customer has known:
  1. what needs to be generated early on
  2. enough language concepts to create test models
- Extensions applied include running the tests from other tools, including the framework code (that is not generated from models) with the tests
- Is not applied to cover all parts of the DSM solution (e.g. translators from model entry to output entry, naming rules with regexp)



MetaCase

**Thank you!**

**Questions?**

Contacts: Juha-Pekka Tolvanen

[jpt@metacase.com](mailto:jpt@metacase.com)

[www.metacase.com](http://www.metacase.com)