The Bmod DSL in MPS by JetBrains

Mathias Ooms

University of Antwerp

9 January 2020

Overview

The main overview:

- Introduction to MPS
- Bmod in MPS
- Demo
- Conclusion and comparisons

Introduction to MPS

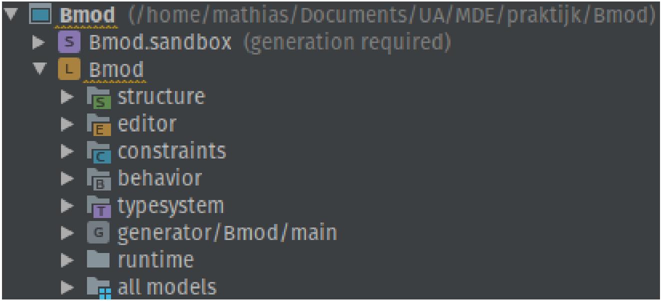
MPS (MetaProgrammingSystem):

- tool developed by JetBrains
- to design DSL
- suitable workbench and IDE

Introduction to MPS

The structure of a DSL in MPS:

- Solutions and languages
- Six main components (models)

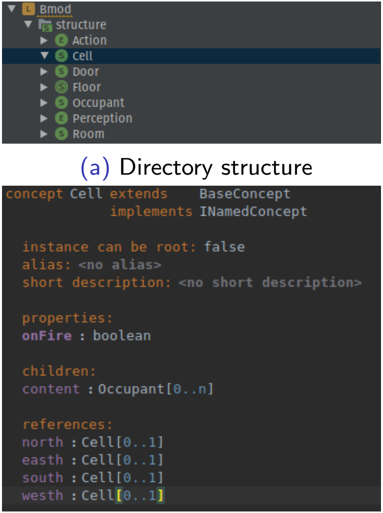


```
▼ Bmod (/home/mathias/Documents/UA/MDE/praktijk/Bmod)
    ▶ S Bmod.sandbox (generation required)
    ▼ L Bmod
        ▶ S structure
        ▶ E editor
        ▶ C constraints
        ▶ B behavior
        ▶ T typesystem
        ▶ G generator/Bmod/main
        ▶ runtime
        ▶ all models
```

Figure 1: The directory structure of a DSL

Bmod in MPS

The structure:



```
▼ Bmod
  ▼ structure
    ▶ Action
    ▼ Cell
    ▶ Door
    ▶ Floor
    ▶ Occupant
    ▶ Perception
    ▶ Room
```

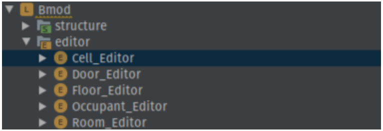
(a) Directory structure

```
concept Cell extends     BaseConcept
             implements INamedConcept

  instance can be root: false
  alias: <no alias>
  short description: <no short description>

  properties:
  onFire : boolean

  children:
  content : Occupant[0..n]

  references:
  north : Cell[0..1]
  easth : Cell[0..1]
  south : Cell[0..1]
  westh : Cell[0..1]
```

(b) The concept Cell

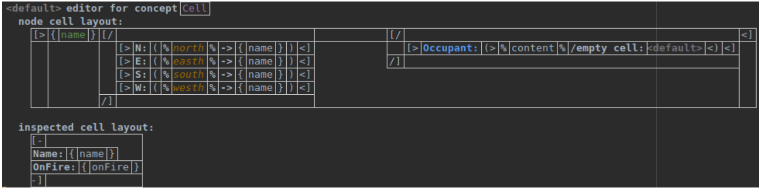
Bmod in MPS

The editor:



```
▼ Bmod
  ▶ structure
  ▼ editor
    ▶ Cell_Editor
    ▶ Door_Editor
    ▶ Floor_Editor
    ▶ Occupant_Editor
    ▶ Room_Editor
```

(c) Directory structure

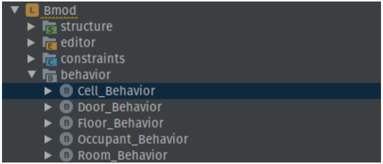


```
<default> editor for concept Cell
  node cell layout:
    [> { name } [/                                        [/                                                                   <]
                  [> N: ( % north % -> { name } ) <]           [> Occupant: (> % content % /empty cell: <default> <) <]
                  [> E: ( % easth % -> { name } ) <]      /]
                  [> S: ( % south % -> { name } ) <]
                  [> W: ( % westh % -> { name } ) <]
                /]

  inspected cell layout:
    [-
    Name: { name }
    OnFire: { onFire }
    -]
```

(d) The concept Cell

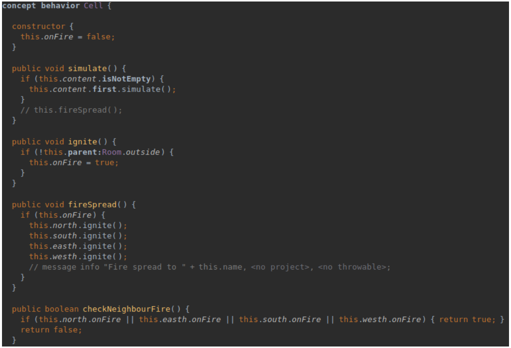
Bmod in MPS

Behavior:



```
Bmod
  structure
  editor
  constraints
  behavior
    Cell_Behavior
    Door_Behavior
    Floor_Behavior
    Occupant_Behavior
    Room_Behavior
```

(e) Directory structure



```
concept behavior Cell {

  constructor {
    this.onFire = false;
  }

  public void simulate() {
    if (this.content.isNotEmpty) {
      this.content.first.simulate();
    }
    // this.fireSpread();
  }

  public void ignite() {
    if (!this.parent:Room.outside) {
      this.onFire = true;
    }
  }

  public void fireSpread() {
    if (this.onFire) {
      this.north.ignite();
      this.south.ignite();
      this.easth.ignite();
      this.westh.ignite();
      // message info "Fire spread to " + this.name, <no project>, <no throwable>;
    }
  }

  public boolean checkNeighbourFire() {
    if (this.north.onFire || this.easth.onFire || this.south.onFire || this.westh.onFire) { return true; }
    return false;
  }
```

(f) The concept Cell

Demo

(click to play)

Conclusions and comparisons

MPS:

- No actual limitations
- Can handle big projects ↔ MetaDepth
- More appropriate for textual languages
- Learning curve ↔ AToMPM
- Handy helping tools