mbeddr (MPS)

Lucas Heer | MDE project presentation | 03.02.2017

Recap

- **mbeddr:**
  - **Extensions to C for embedded system software**
  - **Build with JetBrains MPS (meta programming system) language workbench**
- **Project: Implement custom extension**

But ...

- mbeddr:
  - Consists of 81 “sub-languages“
  - 88000 LOC
  - 1600 concepts in MPS
  - estimated 10 person years

- Instead: Implement directly in MPS!

JetBrains MPS

- "most complete" language workbench
- Developed since the early 2000s
- Core concept: Projectional editing (edit the AST directly and perceive concrete syntax)

Goal

```
Pipeline MyPipeline
    load image.jpg
    adjust brightness value 1.2
    grayscale
    blur method gauss strength 2
    save image.jpg
```

Goal

```
7  import javax.imageio.ImageIO;
8  import java.io.File;
9  import com.jhlabs.image.AbstractBufferedImageOp;
10 import com.jhlabs.image.ContrastFilter;
11 import com.jhlabs.image.GrayscaleFilter;
12 import com.jhlabs.image.GaussianFilter;
13
14 public class MyPipeline {
15
16   private static void saveImage(BufferedImage image, String filename) throws IOException {
17     String fileExt = "";
18     int i = filename.lastIndexOf('.');
19     if (i > 0) {
20       fileExt = filename.substring(i + 1);
21     }
22     ImageIO.write(image, fileExt, new File(filename));
23   }
24
25   public static void main(String[] args) throws IOException {
26     BufferedImage image = null;
27     AbstractBufferedImageOp filter = null;
28
29     try {
30       image = ImageIO.read(new File("image.png"));
31     } catch (IOException e) {
32       e.printStackTrace();
33     }
34
35     filter = new ContrastFilter();
36     filter.setBrightness(1.2f);
37     image = filter.filter(image, null);
38     filter = new GrayscaleFilter();
39     image = filter.filter(image, null);
40     filter = new GaussianFilter(2);
41     image = filter.filter(image, null);
42
43     try {
44       saveImage(image, "image.jpg");
45     } catch (IOException e) {
46       e.printStackTrace();
47     }
```

MPS workflow

1) Abstract syntax (= structure)
2) Concrete syntax (= editor)
3) Generator
4) Optional: Constraints, data flow, type system

Abstract syntax

```
concept Blur extends    Command
             implements <none>

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

  properties:
  method   : string
  strength : integer

  children:
  << ... >>

  references:
  << ... >>
```

Concrete syntax

```
<default> editor for concept Blur
  node cell layout:
    [- blur method ^{ method } strength { strength } -]

  inspected cell layout:
    <choose cell model>
```

Generator

```
root mapping rules:
   ⎡concept           Pipeline⎤ --> map_Pipeline
   ⎢inheritors         false  ⎥
   ⎢condition         <always> ⎥
   ⎣keep input root   default  ⎦
```

Generator

```
public static void main(string[] args) throws IOException {
  BufferedImage image = null;
  $IF$[AbstractBufferedImageOp filter = null;]

  $COPY_SRC$[]

  $LOOP$[$COPY_SRC$[System.out.println("cmd");]]

  $COPY_SRC$[]
}
```

Generator

```
$LOOP$[$COPY_SRC$[System.out.println("cmd"); ]]

$COPY_SRC$[]
  }
}
```

+ Behavior | Typesystem | Actions | Refactorings | Intentions | Find Usages

Inspector

jetbrains.mps.lang.generator.structure.LoopMacro

```
iterate over sequence of nodes

comment            : <none>
mapping label      : <no label>
iteration sequence : (genContext, node, operationContext)->sequence<node<>> {
                       node.commands;
                     }
counter variable   : <no variable>
```

Generator

```
reduction rules:
  [concept    SaveImage] --> reduce_SaveImage
  [inheritors false     ]
  [condition  <always>  ]

  [concept    Grayscale] --> reduce_Grayscale
  [inheritors false     ]
  [condition  <always>  ]

  [concept    LoadImage] --> reduce_LoadImage
  [inheritors false     ]
  [condition  <always>  ]

  [concept    Blur                                                    ] --> reduce_BlurGauss
  [inheritors false                                                   ]
  [condition  (genContext, node, operationContext)->boolean {         ]
                node.method.equals("gauss");
              }
```

Generator

```
template reduce_BlurGauss
input    Blur

parameters
<< ... >>

content node:
{
  BufferedImage image = null;
  AbstractBufferedImageOp filter = null;
  <TF {
      filter = new GaussianFilter($[1]);
      image = filter.filter(image, null);
    } TF>
}
```

Constraints

```
property {strength}
  get:<default>
  set:<default>
  is valid:(propertyValue, node)->boolean {
    propertyValue > 0;
  }

property {method}
  get:<default>
  set:<default>
  is valid:(propertyValue, node)->boolean {
    propertyValue.equals("gauss") || propertyValue.equals("box");
  }
```

Experiences

and

Experiences

Tool complexity

- Look and feel similar to other JetBrains products
- Easy in the beginning
- Documentation, tutorials, screencasts
- Official forum

Experiences

Tool scope

- Language workbench for DSLs
- Tries to leverage projectional editing
- Different representations of the same concept
- Closely tied to Java (main target language)
- Also functions as IDE for language

Experiences

Usability of DSL

- Text-based – Domain expert may prefer graphical representation
- Constrained: Pipeline starts with load and ends with save, command parameters constrained to meaningful values

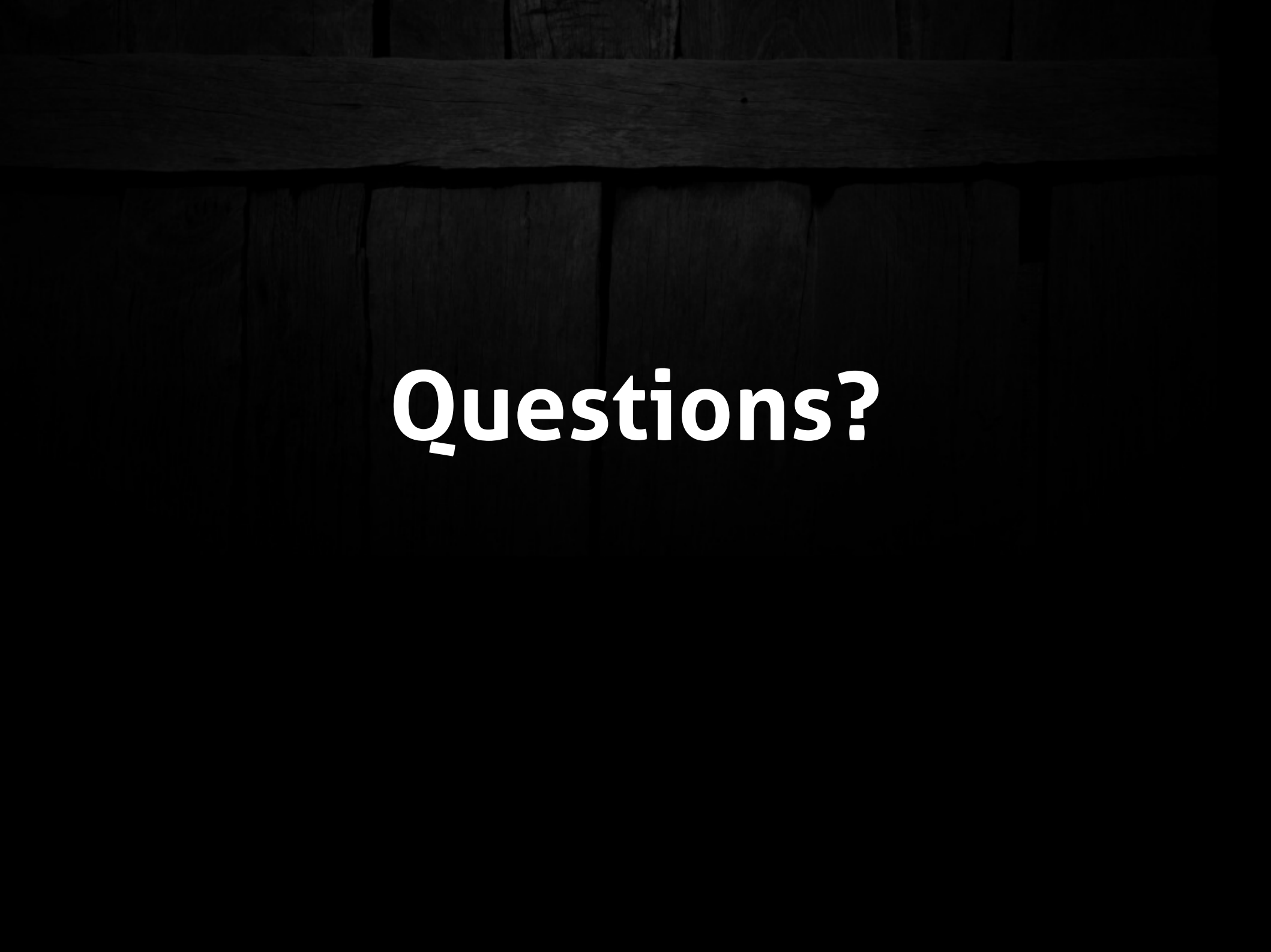
Experiences

Extensibility

- Add new abstract and concrete syntax
- (Optionally) define constraints
- Add generator template

Summary

- Mature language workbench
- Implementing languages on a large scale possible (→ mbeddr)
- Strong Java roots
- Proof-of-concept for projectional editing


Questions?

Resources

- MPS learning: https://confluence.jetbrains.com/display/MPSD34/Fast+Track+to+MPS
- mbeddr development experiences: http://voelter.de/data/pub/voelterEtAl2017-buildingMbeddr.pdf
- MPS overview: http://voelter.de/data/pub/PPPJ.pdf