Not your Daddy's Language
A Language Workbench – a tool for defining, composing and using ecosystems of languages.
Open Source
Apache 2.0
http://jetbrains.com/mps
V 3.3.4 is current.
V 3.4 to be released in Q3 2016
Comprehensive Support for many aspects of Language Definition.

- Refactorings, Find Usages, Syntax Coloring, Debugging, ...

Diagram:

- Language
- Structure: Concepts, Properties, Inheritance, Relationships
- Editor: Projection Rules, Side Transformations, Intentions
- Type System: Typing Rules, Type Checks, Other Validations
- Transformations: Reduction Rules, Weaving Rules, Transformation Priorities
- Constraints: Scopes, Usage Restrictions, Property Value Limitations

Phrases:
- generates to
- extends 0..*

Semantic relationships:
- Structure provides editors for Editor
- Editor defines execution semantics for Type System
- Type System specifies priority 0..*
- Transformations defines static semantics for Type System
MPS uses a Projectional Editor

A Projectional Editor modifies the AST directly. No grammars or parsers are involved.
Projectional Editing

Advantage: Syntactic Flexibility

Regular Code/Text

Mathematical

Tables

Graphical
[Projectional Editing]

Advantage: Syntactic Flexibility / MPS

Regular Code/Text

```c
// [ A documentation comment with references ]
void aSummingFunction(int8[] data, int8 dataLen) {
    int16 sum;
    for (int8 i = 0; i < dataLen; i++) {
        sum += data[i];
    }
} aSummingFunction (function)
```

Mathematical

```c
double midnight2(int32 a, int32 b, int32 c) {
    return \(-b + \sqrt{b^2 - \sum_{i=1}^{4} a * c}\) / 2 * a;
} midnight2 (function)
```

Tables

```c
int16 decide(int8 spd, int8 alt) {
    return spd > 0 || spd > 100 || otherwise 0;
    spd > 0  spd > 100  otherwise 0;
    alt < 0  1        1
    alt == 0 10       20
    alt > 0  30       40
    alt > 100 50       60
} decide (function)
```

Graphical

```
Cst.Customer

Contract
starts: date
ends: date

Tariff attributes
```
[Projectional Editing]
Advantage: Language Composition

Separate Files
Type System
Transformation
Constraints

In One File
Type System
Transformation
Constraints
Syntax
IDE

5+ base languages
50+ extensions to C
10+ extensions to requirements lang.

mbeddrr
No change to definition of $L_1$ or $L_2$ in order to use them together.

**Embedding**

$L_{Host} + L_{Adapt} + L_{Emb} = \text{(result)}$

**Extension**

$L_{Base} + L_{Ext} = \text{(result)}$

**Extension Composition**

$L_{Base} + L_{Ext1} + L_{Ext2} = \text{(result)}$
**[Projectional Editing]**  
Study Results on Editor Usability

People prefer MPS over conventional IDEs

MPS more is more efficient than normal IDEs

MPS more is more productive than normal IDEs

MPS makes it easier to create correct programs

MPS enforces a structurally correct AST

People benefit from language modularity

People benefit from the flexible notations

People benefit from advanced navigation support

The experience with learning MPS is mixed.

It takes some time to get used to MPS
Projecting a Modular Future
Three different case studies of using MPS
http://voelter.de/data/pub/projectingModuleFuture.pdf

mbeddr -- Instantiating a Language Workbench in the Embedded Software Domain
Detailed Discussion of mbeddr (using MPS for embedded s/w engineering)
http://voelter.de/data/pub/voelteretal-mbeddr-AUSE.pdf

Towards User-Friendly Projectional Editors
Study about the usability of projectional editors and MPS‘ „tricks“ for improving it

Supporting Diverse Notations with MPS' Projectional Editor
Paper explaining most of the notations MPS can use.
Demos
Thank you!

voelter@acm.org
www.voelter.de
@markusvoelter