“Houston, we have a Problem!”

Programming Languages

C#  Erlang
C++  Python
Ruby  Java
Groovy
Fortran
C

30
Programming Languages are not expressive enough.

Programming Languages are not high-level enough.

Programming Languages are not abstract enough.

Programming Languages are not domain-specific enough.

Programming Languages are not enough.
So?

Modeling

... Higher Level
... Domain Specific Concepts & Notations
... Code Generation Interpretation

Solves the Problem!
But:

Different Worlds
Programming Tools
\n\neq
Modeling Tools

Different Worlds
Modeling Tool
\n\neq
Modeling Tool

Different Worlds
Mix Models and Programs

Different Worlds
Mix Models and Programs
AST Navigation & Query

Different Worlds
Mix Models and Programs
AST Navigation & Query
Integration of 3GL code
Different Worlds
Mix Models and Programs
AST Navigation & Query
Integration of 3GL code
Code Constraints

Compare

Domain Specific Notations and Abstractions
Flexible!
Limited!
Frameworks Libraries (Fluent) APIs

Graphical Textual Forms Tables
Flexible!
Limited!
Textual Trees

Customize Generator or Interpreter
Flexible!
Limited!
Reflection Meta Programs Open Compilers
Define custom or Query
Navigate or Transform
Flexible!
AST APIs
Static Analysis
RegEx

Custom or Validation
Error Checks
Flexible!
Limited!
IDE plugins
Static Analysis
Open Compilers

Different Representations
and Projections
Flexible!
Limited!
Text is Text
Code Folding
Tree Views
Visualizations

Mixing and Composing
Languages
Flexible?
Limited!
Python-to-C-like
Internal DSLs
Embed-As-String
Specific: LINQ

Scalable
Usable
IDE Support

Brittle!
Mature!
Modeling Tools...!?
Brittle! Mature!

Versioning
Diff, Merge
Branching

Gets some
Jobs done.
some people doubt that...

Gets the
Job Done!
everybody agrees...

Why
the difference?

Modeling | Programming
--- | ---
... (Mostly) Textual Notations
... Concrete Syntax
... Storage
... (Fancy) ASCII Editors
... Read-Only
Visualizations
Modeling

... (Mostly) Graphical Notations
... Abstract Syntax Storage
... Projecting Editors
... Different editable views for model

Programming

... (Mostly) Textual Notations
... Concrete Syntax Storage
... (Fancy) ASCII Editors
... Read-Only Visualizations

Why the difference?

It is time for ...

... a Different Perspective

Programming the way we do Modeling?

Modeling the way we do Programming?

Modeling == Programming == Modeling
Where do we go from here?

We don’t want to model, we want to program!

... at different levels of abstraction
... from different viewpoints
... integrated!

... with different degrees of domain-specificity
... with suitable notations
... with suitable expressiveness
We don’t want to model, we want to program!

And always: precise and tool processable

1. Enabling Technologies
2. Available Tooling
3. A vision for programming
Enabling Technologies

Advanced Parser Generators

Modeling as Programming

... (Mostly) Textual Notations
... Concrete Syntax Storage
... (Fancy) ASCII Editors
... Read-Only Visualizations

Custom Syntax

Graphical Textual Symbolic++

IDE Support

Complete Symbolic Integration

Teamwork Debugging Custom Editors

Goto Def Find Refs Refactoring
Xtext-like Tools provide editor support

Custom Editors
Teamwork
Goto Def
Find Refs
Refactoring

Limited to Unicode

how to handle non-character symbols

Graphics !\ne Text
two worlds...
separate editors
... per syntax/viewpoint
... models can still be ref integrated

Enabling Technologies

Projectional Editing

Parser-based
text
... to tree
... to text
**Projectional**

- **tree**
  - to text-lookalike (editor)
  - to other trees ... [*]
  - to text

**Programming as Modeling**

... (Mostly) Graphical Notations
... Abstract Syntax
... Storage
... Projecting Editors
... Different editable views for model

**Language Composition**

There's no parsing.
Unique Language Element Identity.
Unlimited language composition.

**Flexible Notations**

- Textual like ASCII
- Graphical box & line
- Semi-Graphical mathematical

} treated the same can be mixed

**Automatic IDE Extension**

tool support is inherent for languages build with projectional tools

language definition implies IDE definition
**Multiple Notations**

... for the same concepts
- e.g. in different contexts
- or for different tasks

**Partial Projections**

... different views
- ... for different roles/people
- ... only a particular variant

**Storage**

!= Schema

... store arbitrary meta data
- change log
- conflicting information
- variability annotations
- "independent of language schema!
- "aspects", overlay

**Live Programs**

think: spreadsheet

a change to one part of program can lead to (dependent) changes in other parts

**Tree Editing**

... is different from editing text
... try to make it feel like text
... takes some getting used to
but: for more flexible notations a more general editing paradigm is needed

**Infrastructure Integration**

... storage is not text
... diff/merge must be in tool
... existing text tools don’t work
Proprietary Tools

... no standards
... no interop

Available Tooling

Eclipse Xtext

http://eclipse.org/modeling

http://eclipse.org/xtext
Xtext: Specify Grammar

Xtext: Gen. Meta Model

Xtext: Constraints

Xtext: Generated Editor

Xtext: Generated Editor

Xtext: Generated Editor

Code Completion

Syntax Coloring

Custom Keyword Coloring
Realtime Constraint Validation

Customizable Outlines

Code Folding

Goto Definition
Find References
Cross-File References
Model as EMF
Building DSLs with Eclipse Xtext

Available Tooling

Jetbrains’ Programming Meta System

also do...

IntelliJ IDEA Resharper

released in

Q3 2009
currently

1.1 RC1
Build new standalone DSLs
Build DSLs that reuse parts of other languages

(MPS comes with BaseLanguage)
extend base language

Language Extension Example
Language Extension Example

**Old**

```java
ReadWriteLock l = ... 
   l.readLock().lock();
   try {
      //code
   } finally {
      l.readLock().unlock();
   }
```

Language Extension Example

**Old**

```java
ReadWriteLock l = ... 
   l.readLock().lock();
   try {
      //code
   } finally {
      l.readLock().unlock();
   }
```

**New**

```java
ReadWriteLock l = ... 
   lock (l) { 
      //code
   }
```

Structure ➤ Editor ➤ Typesystem ➤ Generator

Structure ➤ Editor ➤ Typesystem ➤ Generator

Structure ➤ Editor ➤ Typesystem ➤ Generator

Structure ➤ Editor ➤ Typesystem ➤ Generator
Language Extension Example

Result behaves like a native base language construct
Language Extension Example

Result behaves like a native base language construct

Language Extension Example

Translated to regular Java code based on the generator

Example Languages

UI Language

Example Languages

HTML Templates

Example Languages

Persistent Classes
DEMO II

Building DSLs with JetBrains MPS

A vision for Programming

Programming Languages are not *

**MODULAR**

effective enough.

Programming Languages are not *

**COMPOSABLE**

effective enough.
Programming Languages are not CONFIGURABLE enough.

Programming Languages are not ADAPTABLE enough.

Programming Language Syntax is not FLEXIBLE enough.

Big Language? with many first class concepts!

Small Language? with a few, orthogonal and powerful concepts

Modular Language with many optional, composable concepts
Modular Language
Like frameworks and libraries,
but with syntax and IDE support

Not a new idea...

Growing A Language
(Guy L. Steele)
Freely define languages and integrate them

language ::= schema + editors + generators

use persistent abstract representation

projectional editing

Language Workbench
(Martin Fowler)
Language Workbench (Martin Fowler)

Persist incomplete or contradictory information

Language Workbench (Martin Fowler)

Powerful editing, testing, refactoring, debugging, groupware

Language Workbench (Martin Fowler)

Support for "classical" programming and modeling

Syntax primarily textual

Syntax primarily textual

Syntax primarily textual

with more symbols

think: mathematics

sometimes box&line style
Syntax primarily textual sophisticated visualizations

Viewpoints suitably abstractions and notations for each

Viewpoints Integrated via symbolic references and seamless transitions

Viewpoints Business custom purpose-built create/include

Viewpoints Business Custom Notations real business expert integration
But that’s another talk...

Example Languages
Adding *matrices* to C in an embedded environment.

**Currently:**

1. Declare Data Structures in XML
2. Generate Headers
3. Implement manually in C

**Currently:**

Matrices **not supported** in XML format and generator.
Currently:

**Tool team**

would have to do the extension

... a lot of work

... busy

... one central tool


Currently:

No real **compiler support** in the resulting C code

... type checks

... operator overloading

... generics (matrix<int>)

... matrix syntax?


Better Solution

```
qmatrixtlbv[3,3] ModelMatrix
    p partially_1 p

vector<int>& modelvector = temps;

vector<int>& gaussvecint = gmatrixint* A, vector<int>& b |
    if (gaussvecint... a A x b) return A \ b
    return 0;

vector<int>& eng = ModelMatrix * modelvector;

int i = gmatrixint A |
    return \sum_{i=1}^{m} \sigma (i) \prod_{j=1}^{n} a_{i,j}(i);
```

Better Solution

```
qmatrixtlbv[3,3] ModelMatrix
    p partially_1 p

vector<int>& modelvector = temps;

vector<int>& gaussvecint = gmatrixint* A, vector<int>& b |
    if (gaussvecint... a A x b) return A \ b
    return 0;

vector<int>& eng = ModelMatrix * modelvector;

int i = gmatrixint A |
    return \sum_{i=1}^{m} \sigma (i) \prod_{j=1}^{n} a_{i,j}(i);
```
Better Solution

highlight syntax for
vectors and matrices

Better Solution

operator
overloading

Better Solution

static optimization

Better Solution

math notation

Better Solution

a separate
language module

used only by those
who really need it
In addition: PLE Variability

In addition: PLE Variability

In addition: PLE Variability

In addition: PLE Variability

In addition: PLE Variability

Available Tooling

Available Tooling

Intentional Software Domain Workbench
Dec 15, 2008
currently at 1.4

Commercial Product

Eval available upon request

Statemachine Example

Statemachine Example

Native Projection

Tabular Projection
Statemachine Example

Textual DSL Projection

Java Projection

Ruby Projection

Two projections side/side

Two projections side/side
Pension Workbench Example

Text Editing Domain

Insurance Mathematics Domain

Pension Workbench Example

Pension Contract Rules Domain

All in one Document

Symbolically integrated
Domain Specific Languages (for Product Lines)

Feb 3 – 4, 2010
Eindhoven, NL

Xtext

Feb 12./13.11.2009: Hamburg
Feb 18./19.02.2010: Hamburg
Feb 25./26.02.2010: Stuttgart
Feb 25./26.02.2010: München

itemis

Eclipse Modeling (GEF, EMF, GMF)

Feb 23./27.11.2009: Bonn
Feb 30./04.12.2009: München

itemis

THE END. Really.