Language Extension and Composition with Language Workbenches

Eelco Visser
TU Delft
E.Visser@tudelft.nl

Markus Voelter
Independent/itemis
voelter@acm.org
Different Worlds

Programming Tools ≠ Modeling Tools
Different Worlds

Modeling Tool
!=
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
Why
the difference?
History?
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
Programming == Modeling
We don’t want to model, we want to program!
We don’t want to model, we want to program!

... at different levels of abstraction
... from different viewpoints
... integrated!
We don’t want to model, we want to program!

... 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
Where do we go from here?
A vision for programming
A vision for programming

Enabling Technologies
1. A vision for programming
2. Enabling Technologies
3. Available Tooling
A vision for Programming
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,
Modular Language

Like frameworks and libraries, but with syntax and IDE support.
Not a new idea...
Growing A Language
(Guy L Steele)
Language Workbench

(Martin Fowler)
Language Workbench
(Martin Fowler)

Freely define languages and integrate them
Language Workbench
(Martin Fowler)

use
persistent
abstract
representation
language ::= schema + editors + generators
Language Workbench
(Martin Fowler)

projectional editing
Language Workbench
(Martin Fowler)

persist
incomplete
or
contradictory
information
Language Workbench
(Martin Fowler)

powerful editing testing refactoring debugging groupware

language definition implies IDE definition
Language Workbench
(Martin Fowler)

support for "classical" programming and modeling
Syntax primarily textual
Syntax primarily textual with more symbols

think: mathematics
Syntax
primarily
textual

sometimes
box&line style
Syntax

primarily
textual

sophisticated visualizations
Viewpoints suitable abstractions and notations for each
Viewpoints

Integrated via symbolic references and seamless transitions
custom purpose-built create/include
Viewpoints

Custom Notations

real business expert integration

Business
Viewpoints

Custom Notations

real business expert integration

But that’s another talk...
Viewpoints
Technical

predefined library
configure
Enabling Technologies
Enabling Technologies

Advanced Parsers
Modeling as Programming

... (Mostly) Textual Notations

... Concrete Syntax Storage

... (Fancy) ASCII Editors

... Read-Only Visualizations
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
Programming as Modeling

... (Mostly) Graphical Any kind of 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
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
... storage is not text
... diff/merge must be in tool
... existing text tools don‘t work
Available Tooling
Available Tooling

TU Delft’s Spoofax
Available Tooling

Jetbrains’ Meta Programming System
released in Q3 2009

currently 1.5
Open Source under

Apache 2.0
Build new **standalone** DSLs
Build new **standalone** DSLs
Build DSLs that **reuse** parts of other languages
Build new **standalone** DSLs
Build DSLs that **reuse** parts of other languages

(MPS comes with **BaseLanguage**)

**extend** base language
Build new **standalone** DSLs

Build DSLs that **reuse** parts of other languages

(MPS comes with **BaseLanguage**)
Learn more at:

http://jetbrains.com/mps
http://code.google.com/p/mps-lwc11/wiki/GettingStarted
http://mbeddr.com

http://languageworkbenches.net
Language Extension and Composition with Language Workbenches

The End.