mbeddr
How we built it and what we have learned

Markus Völter

Bernd Kolb, Dan Ratiu, Domenik Pavletic, Kolja Dumman, Sascha Lisson, Tamas Szabo, Niko Stotz, Zaur Molotnikov

voelter { ingenieurbüro für softwaretechnologie //

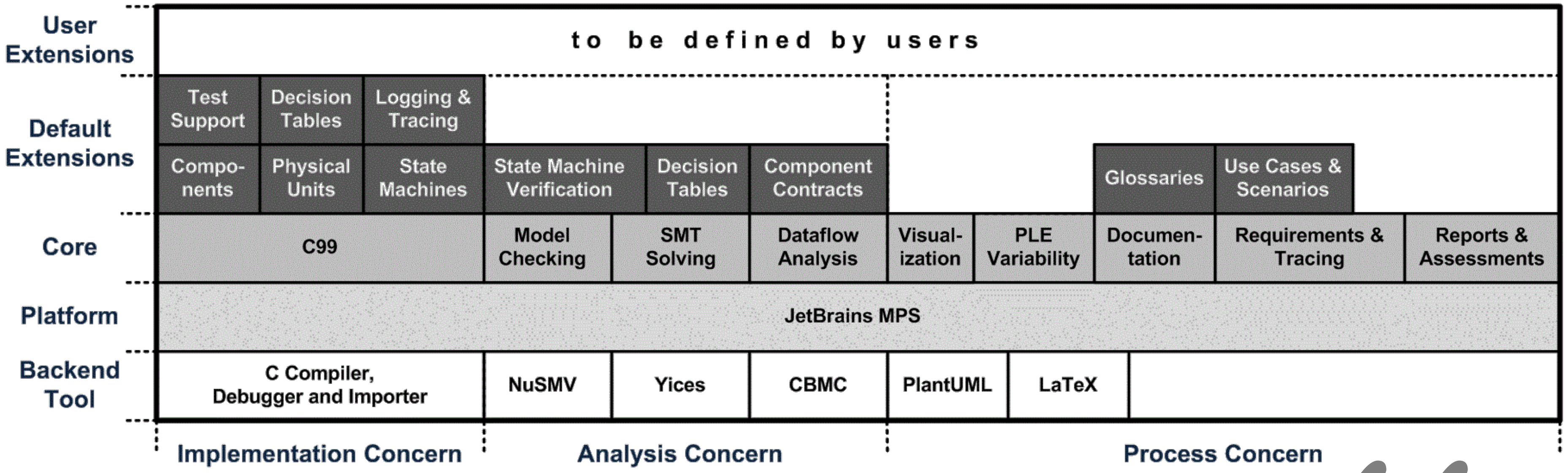
voelter@acm.org
www.voelter.de
@markusvoelter

1

mbeddr

mbeddr

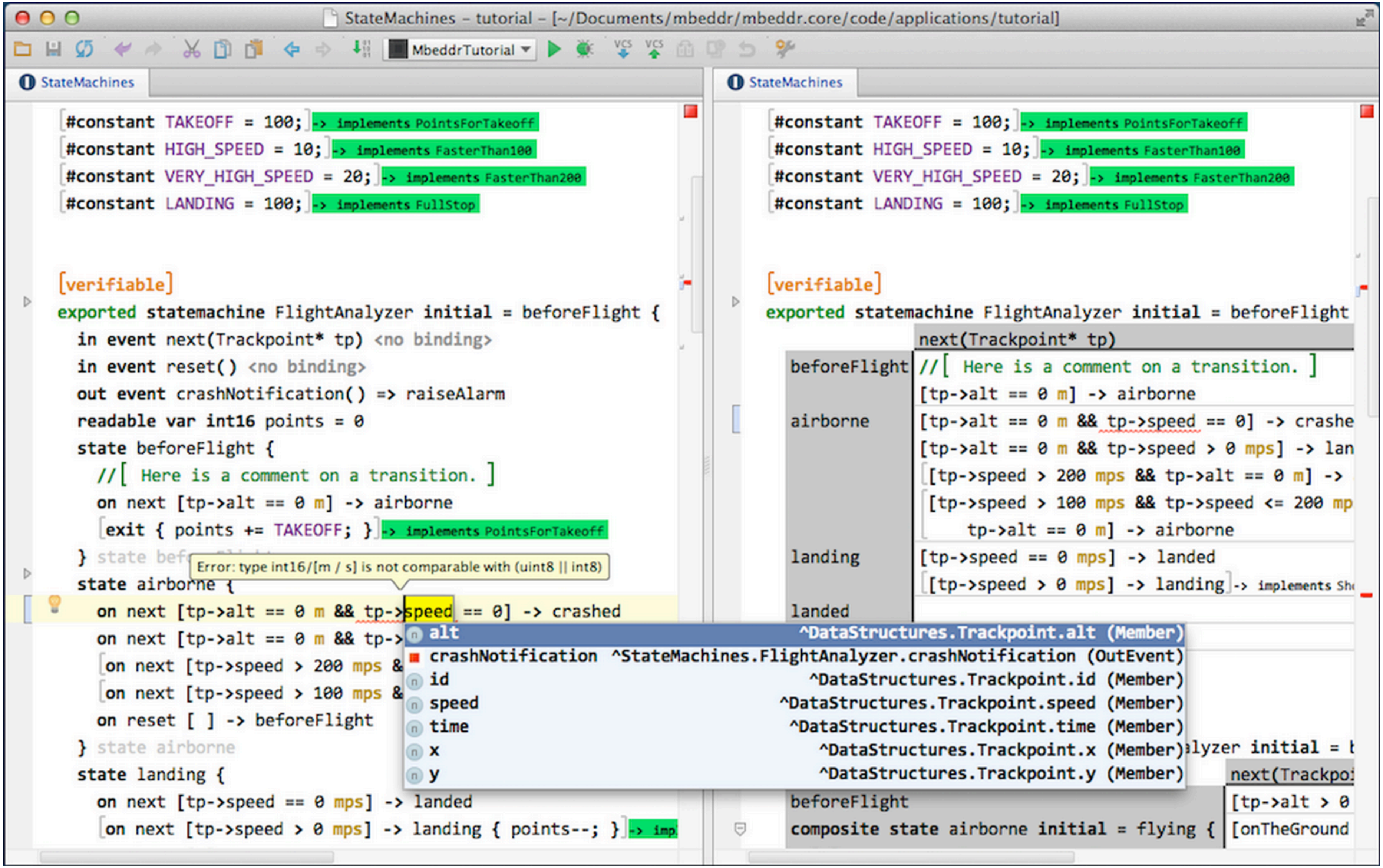
An extensible set of integrated languages for embedded software engineering.



	Implementation Concern	Analysis Concern	Process Concern
User Extensions	to be defined by users		
Default Extensions	Test Support Decision Tables Logging & Tracing Compo-nents Physical Units State Machines	State Machine Verification Decision Tables Component Contracts	Glossaries Use Cases & Scenarios
Core	C99	Model Checking SMT Solving Dataflow Analysis	Visual-ization PLE Variability Documen-tation Requirements & Tracing Reports & Assessments
Platform	JetBrains MPS		
Backend Tool	C Compiler, Debugger and Importer	NuSMV Yices CBMC	PlantUML LaTeX

„Specific Languages“

mbeddr


StateMachines - tutorial - [~/Documents/mbeddr/mbeddr.core/code/applications/tutorial]
MbeddrTutorial
StateMachines
StateMachines

```
#constant TAKEOFF = 100; -> implements PointsForTakeoff
#constant HIGH_SPEED = 10; -> implements FasterThan100
#constant VERY_HIGH_SPEED = 20; -> implements FasterThan200
#constant LANDING = 100; -> implements FullStop


[verifiable]
exported statemachine FlightAnalyzer initial = beforeFlight {
  in event next(Trackpoint* tp) <no binding>
  in event reset() <no binding>
  out event crashNotification() => raiseAlarm
  readable var int16 points = 0
  state beforeFlight {
    //[ Here is a comment on a transition. ]
    on next [tp->alt == 0 m] -> airborne
    exit { points += TAKEOFF; } -> implements PointsForTakeoff
  } state bef
  state airborne {
    on next [tp->alt == 0 m && tp->speed == 0] -> crashed
    on next [tp->alt == 0 m && tp->
    on next [tp->speed > 200 mps &
    on next [tp->speed > 100 mps &
    on reset [ ] -> beforeFlight
  } state airborne
  state landing {
    on next [tp->speed == 0 mps] -> landed
    on next [tp->speed > 0 mps] -> landing { points--; } -> imp
```

Error: type int16/[m / s] is not comparable with (uint8 || int8)

```
alt                  ^DataStructures.Trackpoint.alt (Member)
crashNotification    ^StateMachines.FlightAnalyzer.crashNotification (OutEvent)
id                   ^DataStructures.Trackpoint.id (Member)
speed                ^DataStructures.Trackpoint.speed (Member)
time                 ^DataStructures.Trackpoint.time (Member)
x                    ^DataStructures.Trackpoint.x (Member)
y                    ^DataStructures.Trackpoint.y (Member)
```

```
#constant TAKEOFF = 100; -> implements PointsForTakeoff
#constant HIGH_SPEED = 10; -> implements FasterThan100
#constant VERY_HIGH_SPEED = 20; -> implements FasterThan200
#constant LANDING = 100; -> implements FullStop


[verifiable]
exported statemachine FlightAnalyzer initial = beforeFlight
```

	next(Trackpoint* tp)
beforeFlight	//[Here is a comment on a transition.] [tp->alt == 0 m] -> airborne
airborne	[tp->alt == 0 m && tp->speed == 0] -> crashe [tp->alt == 0 m && tp->speed > 0 mps] -> lan [tp->speed > 200 mps && tp->alt == 0 m] -> [tp->speed > 100 mps && tp->speed <= 200 mp tp->alt == 0 m] -> airborne
landing	[tp->speed == 0 mps] -> landed [tp->speed > 0 mps] -> landing -> implements Sh
landed	

```
lyzer initial = 
```

	next(Trackpoi
beforeFlight	[tp->alt > 0
composite state airborne initial = flying {	[onTheGround

mbeddr

itemis

fortiss

BMW Car IT

Open Source @ eclipse.org
Eclipse Public License 1.0
http://mbeddr.com

Bundesministerium
für Bildung
und Forschung

mbeddr

itemis France: Smart Meter

First significant mbeddr project
ca. 100,000 LoC
about to be finished
great modularity due to components
uses physical units extensively
great test coverage due to special extensions

mbeddr

ACCEnT Control.Lab

LMS INTERNATIONAL
Researchpark Haasrode 1237 | Interleuvenlaan 68 | B-3001 Leuven [Belgium]
T +32 16 384 200 | F +32 16 384 350 | info@lmsintl.com | www.lmsintl.com

Worldwide
For the address of your local representative, please visit www.lmsintl.com/lmsworldwide

LMS is a leading provider of test and mechatronic simulation software and engineering services in the automotive, aerospace and other advanced manufacturing industries. As a business segment within Siemens PLM Software, LMS provides a unique portfolio of products and services for manufacturing companies to manage the complexities of tomorrow's product development by incorporating model-based mechatronic simulation and advanced testing in the product development process. LMS tunes into mission-critical engineering attributes, ranging from system dynamics, structural integrity and sound quality to durability, safety and power consumption. With multi-domain and mechatronic simulation solutions, LMS addresses the complex engineering challenges associated with intelligent system design and model-based systems engineering. Thanks to its technology and more than 1250 dedicated people, LMS has become the partner of choice of more than 5000 manufacturing companies worldwide. LMS operates in more than 30 key locations around the world.

Siemens PLM Software

SIEMENS

LMS®

A Siemens Business

mbeddr

20+ Projects in various stages

by various “Big Name” companies.

Approach also used in other Domains

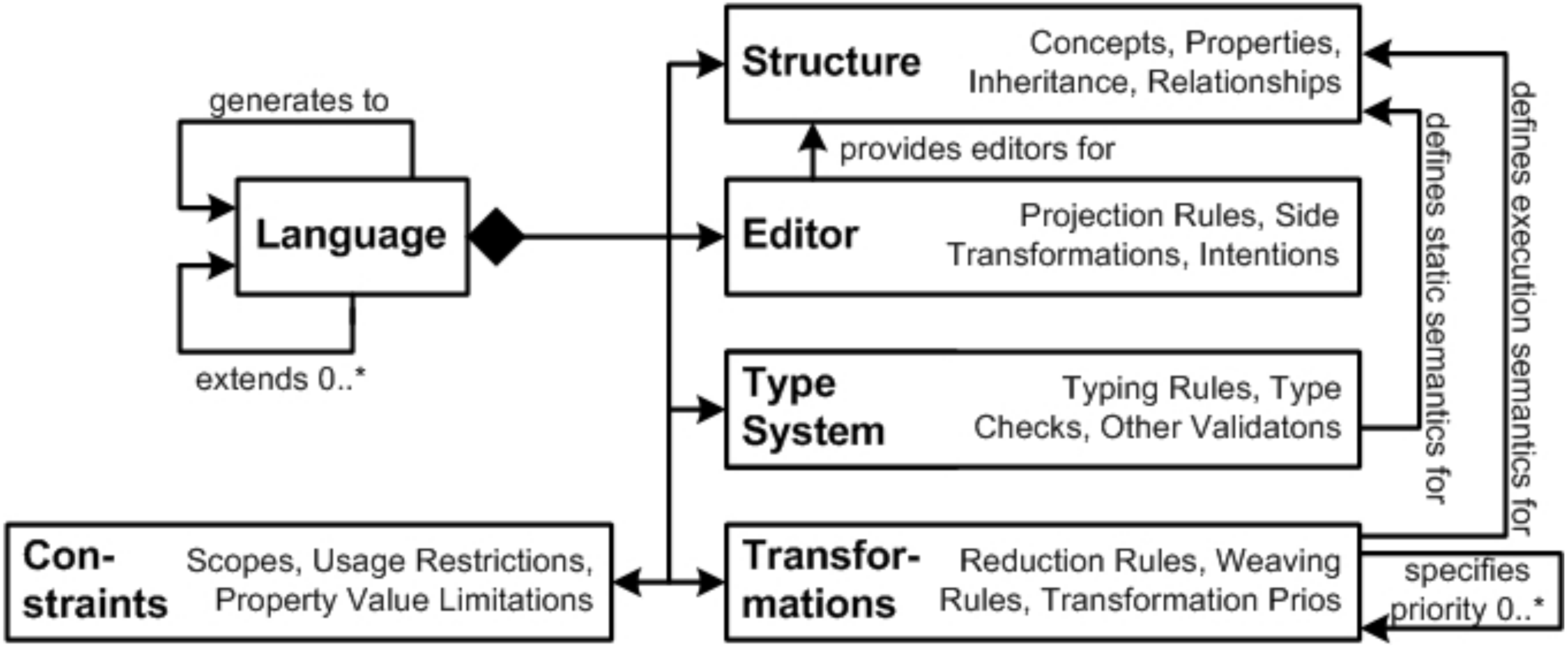
Insurance, Finance

2

The Language Workbench

Open Source
Apache 2.0
http://jetbrains.com/mps

[Language Workbench]

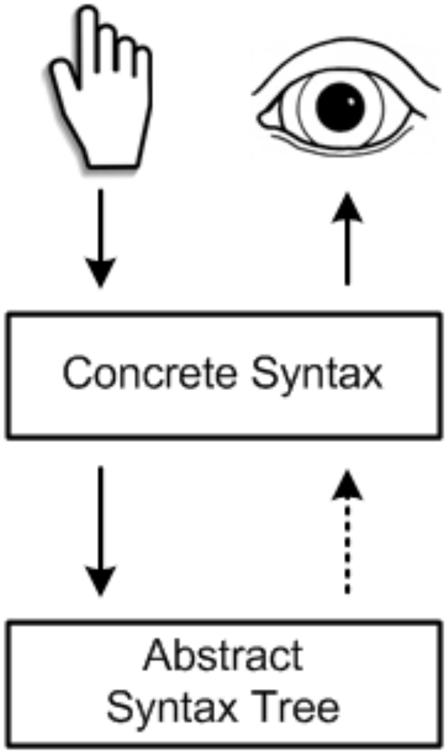

generates to
Language
extends 0..*
Structure
Concepts, Properties, Inheritance, Relationships
provides editors for
Editor
Projection Rules, Side Transformations, Intentions
Type System
Typing Rules, Type Checks, Other Validatons
Con- straints
Scopes, Usage Restrictions, Property Value Limitations
Transfor- mations
Reduction Rules, Weaving Rules, Transformation Prios
defines static semantics for
defines execution semantics for
specifies priority 0..*

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

Projectional Editing

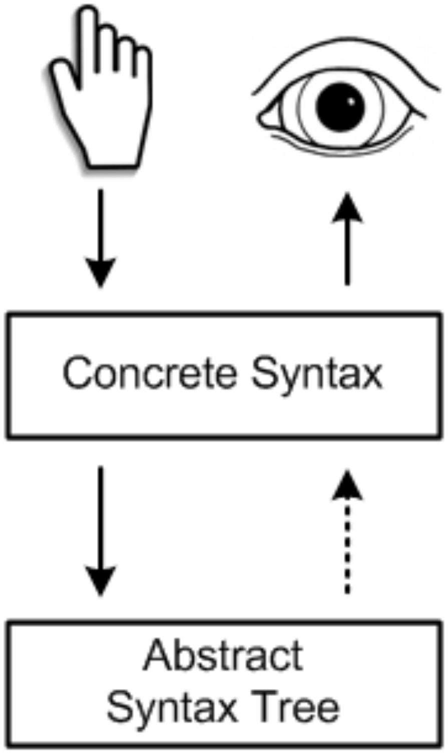
[Projectional Editing]

Parsing



[Projectional Editing]

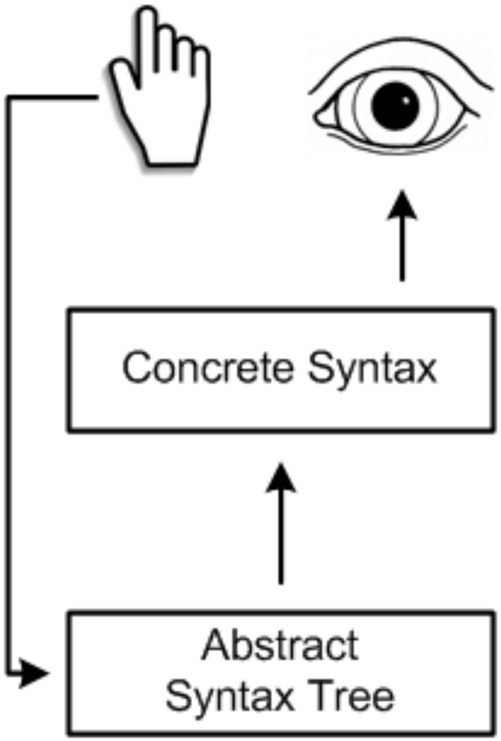
Parsing



Concrete Syntax

Abstract Syntax Tree

Projectional Editing



Concrete Syntax

Abstract Syntax Tree

[Projectional Editing]

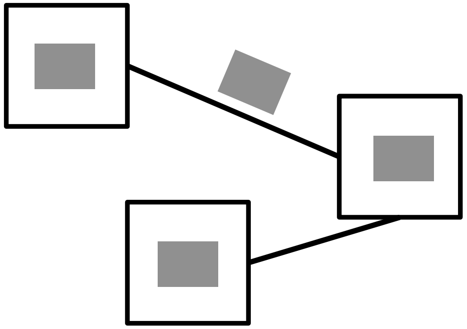
Syntactic Flexibility

Regular Code/Text

Mathematical

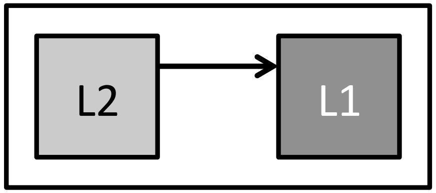
Tables

Graphical



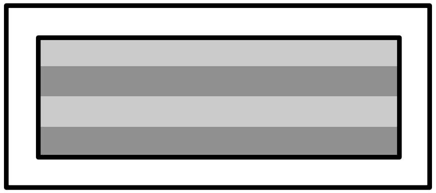
[Projectional Editing]

Language Composition



Separate Files

Type System
Transformation
Constraints

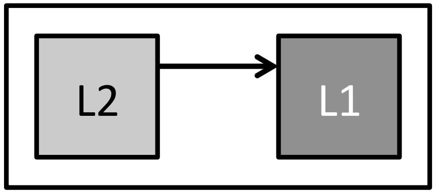


In One File

Type System
Transformation
Constraints
Syntax
IDE

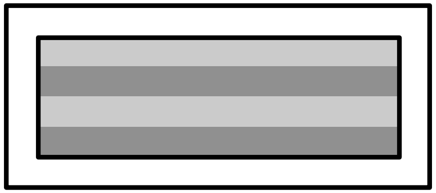
[Projectional Editing]

Language Composition



Separate Files

Type System
Transformation
Constraints



In One File

Type System
Transformation
Constraints
Syntax
IDE

mbeddr

50+ extensions to C
10+ extensions to requirements lang.

3

Bottom Line Up-Front

Fundamentally it was a success.

Fundamentally it was a success.

Resarch Project Completed

Fundamentally it was a success.

Resarch Project Completed

Lively OS project

Fundamentally it was a success.

Resarch Project Completed

Lively OS project

Paying customers

Fundamentally it was a success.

Resarch Project Completed

Lively OS project

Paying customers

Expanded to other domains

Fundamentally it was a success.

Resarch Project Completed

Lively OS project

Paying customers

Expanded to other domains

Learned a lot

Fundamentally it was a success.

Resarch Project Completed

Lively OS project

Paying customers

Expanded to other domains

Learned a lot

Fundamentally it was a success.

Resarch Project Completed

Lively OS project

Paying customers

Expanded to other domains

Learned a lot

Papers + my PhD

Fundamentally it was a success.

Resarch Project Completed

Lively OS project

Paying customers

Expanded to other domains

Learned a lot

Papers + my PhD

New Research Opportunities

Fundamentally it was a success.

But there were Problems/Challenges/ Lessons Learned

Good Experience.

Good Experience. +

Neutral Observation ○

Good Experience. +

Neutral Observation O

Problem/Challenge −

4

Lessons: mbeddr-related

Default extensions are useful, in particular components, state machines and units.

Easy and useful to add customer specific extensions.

The non-code languages (Req, PLE, Doc) are more useful and important than we initially thought.

RCP version of MPS crucial for end users. We had underestimated this.

Decided not to make extensions BL independent, they are actually C extensions and cannot be used with other base languages.

mbeddr requires a fundamental change in how people develop software.

Makes it hard to "sell".

Integration with analysis
tools work and is useful,
but performance
and config of the analysis
is still an issue.
(leaky abstraction)

Do more verification on code level than on model level because of consistency problem with code.

Writing **optimizing** generators is hard.

Underestimated importance of style of generated code.

Some extensions had to be redone (units) because we didn't get them right the first time.

Splitting C into several languages not so useful – dependencies!

Some of our "C cleanups" were not sustainable.

Importer more challenging than expected (because of #preprocessor)

5

Lessons: MPS-related

Modularity works in principle and practice

MPS‘ approach scales to non-trivial and many languages.

Flexible notations actually work and are useful in practice.

Decoupling Notation from Language works.

+

MPS is easily extensible with new notational styles.

Editor Usability less and less of a Problem as MPS evolved/s.

VCS integration works well (diff/merge)

MPS can be extended with the same means – bootstrapped.

Language Testing works well enough to stabilize non-trivial languages (and type systems).

MPS also supports debugging of DSLs – even though we had to extend the mechanism

MPS had quite a number of bugs and a few conceptual problems. We worked with JetBrains to resolve them.

Type system is the most challenging aspect of language definition.

No direct support for detecting semantic interations between languages

Modularity:
Sometimes base language requires change (introduction of abstract class or interface)

Model Migration upon language change is sometimes tedious.
To be fixed in 3.2

Renaming languages is sometimes painful (because of bugs)

Cross-model generation not possible – being worked on right now.

Ability to create additional language aspects missing (you can existing ones)

Debugger definition separate from generator; leads to duplication

Tracing back from the generated code to the model is not always consistent; problems for debugger and analysis.

−

many aspects of
language definition
too „procedural" and
hence hard to analyze.

Due to the open world assumption of MPS, there is a "feeling of incompleteness“ in aspects like e.g. in lifting analyses results.

6

Lessons:
Life in General

A govenment project that really worked together on **one** tool – rare!

+

mbeddr was only possible because of a highly motivated small team.

JetBRAINS
develop with pleasure!

mbeddr was only possible because of itemis support.

mbeddr was only possible because of JetBRAINS support.

Underestimated "overhead": installer, docs, ...

Not enough time for refactorings - as usual.

More and more team leading and organization for me and Bernd.

+

The best 3 years in my professional life so far.

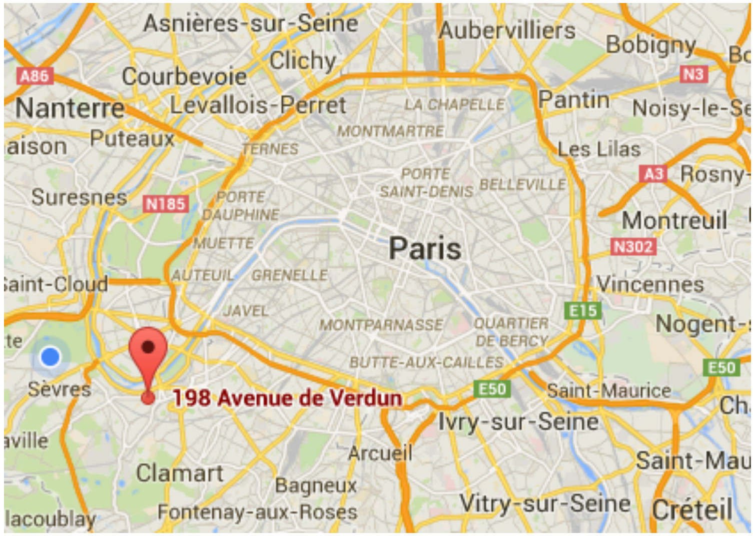
Thank you!

itemis France

- **Company profile**
  - Founded in 2008
  - Based in Issy-les-Moulineaux
  - 7 employees
  - Jeune Entreprise Innovante

Focus

- Model-driven tools
- Embedded systems



eclipse

mbeddr

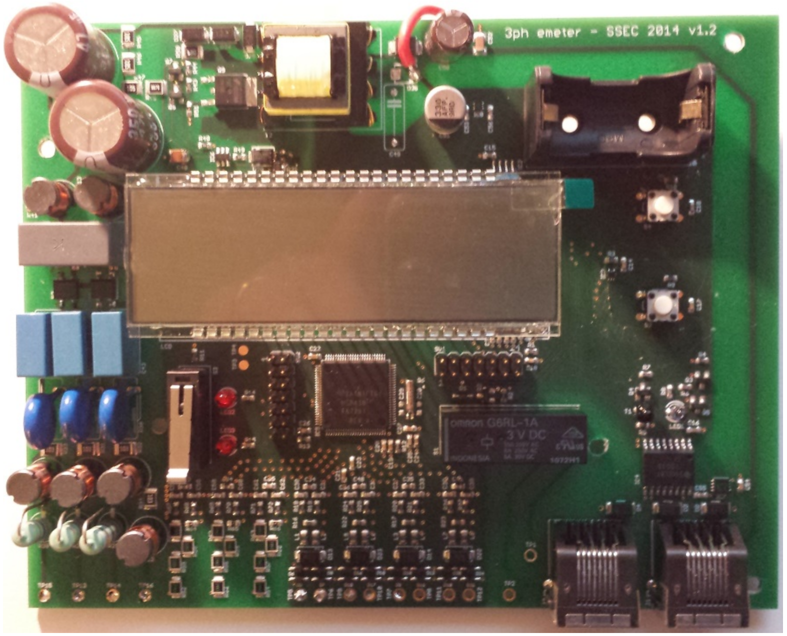
Internship („stage“): Smart Metering

Context

- 3-phase smart meter for Saudi Arabian market
- Measurement of electrical energy consumption, data analysis, automatic meter reading
- Fully developed by itemis France (hardware, software, casing)

Internship task

- Development of advanced smart metering functions (multi-tariff support, consumption profiles, etc.)
- Integration on embedded target (Texas Instruments MSP430)
- Using mbeddr and C



Contact

Stephan Eberle

R & D Director France

+33 6 64 18 39 10

stephan.eberle@itemis.com