Supporting
Diverse Notations
In MPS' Projectional Editor

Markus Völter and Sascha Lisson

with **Bernd Kolb, Domenik Pavletic, Kolja Dumman, Tamas Szabo, Niko Stotz, Dan Ratiu, Zaur Molotnikov,**

voelter ingenieurbüro für softwaretechnologie // itemis

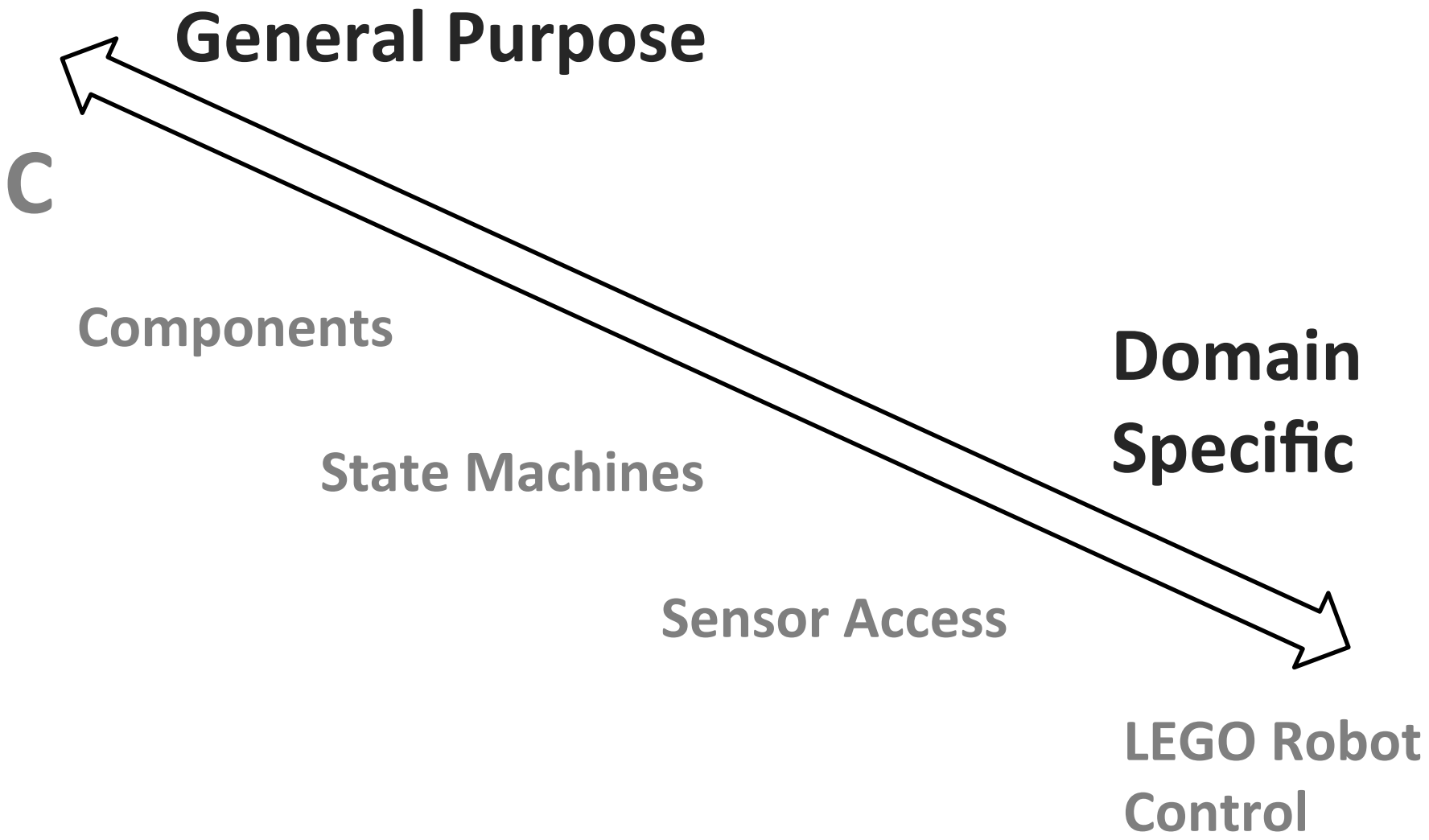
voelter@acm.org
www.voelter.de
@markusvoelter

1

Languages, Notations

Models, Programs

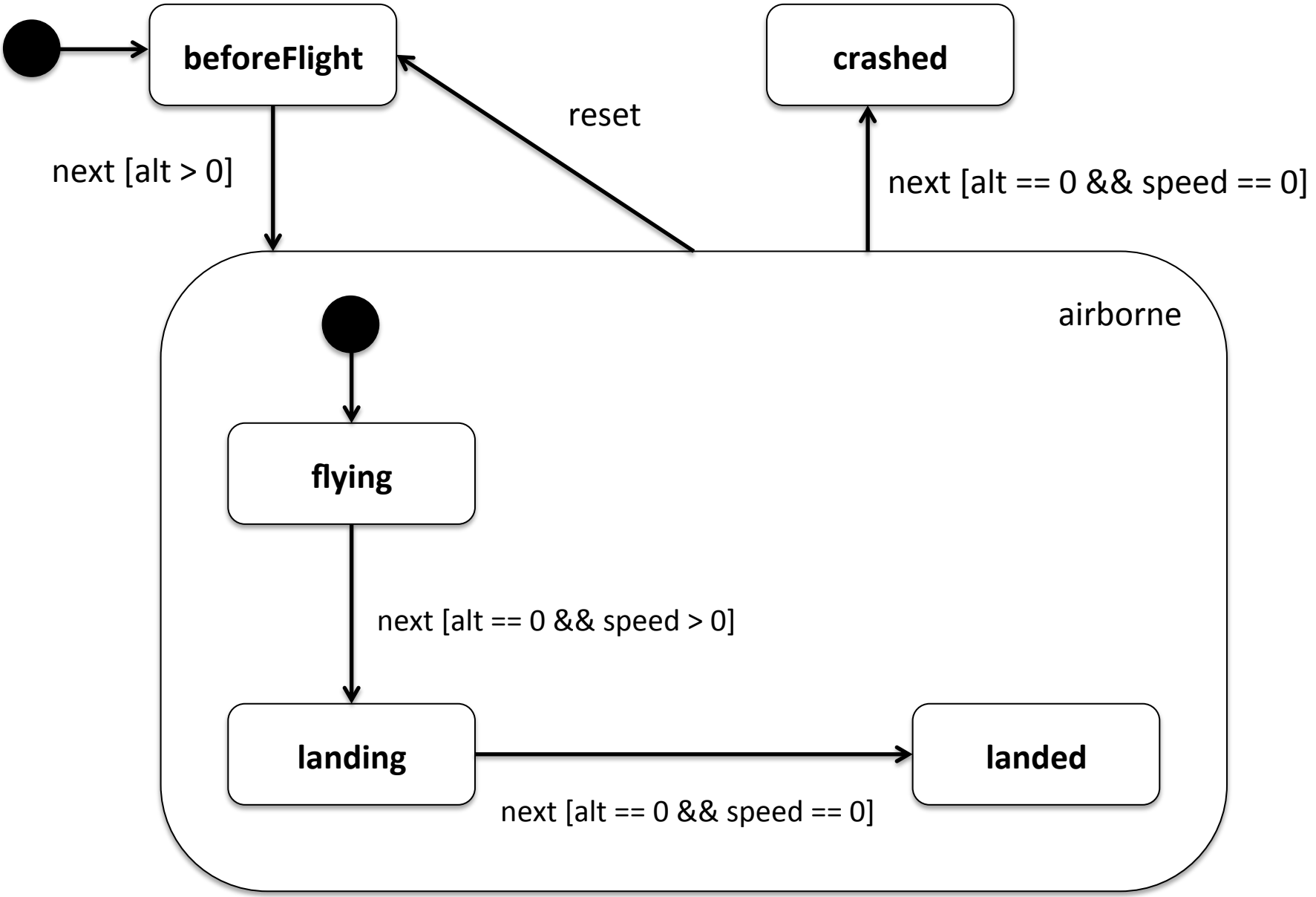
	more in GPLs	more in DSL
Domain Size	large and complex	smaller and well-defined
Designed by	guru or committee	a few engineers and domain experts
Language Size	large	small
Turing-completeness	almost always	often not
User Community	large, anonymous and widespread	small, accessible and local
In-language abstraction	sophisticated	limited
Lifespan	years to decades	months to years (driven by context)
Evolution	slow, often standardized	fast-paced
Incompatible Changes	almost impossible	feasible


General Purpose
C
Components
State Machines
Sensor Access
Domain
Specific
LEGO Robot
Control

Model or Code?

```
Trackpoint* makeTP(uint16 alt, int16 speed) {
  static int8 trackpointCounter = 0;
  trackpointCounter++;
  Trackpoint* tp = ((Trackpoint*) malloc(sizeof Trackpoint));
  tp->id = trackpointCounter;
  tp->timestamp = trackpointCounter;
  tp->alt = alt
  tp->speed = speed
  return tp;
}
```

Model or Code?



Model or Code?

```
statemachine HierarchicalFlightAnalyzer initial = beforeFlight {
  in next()
  in reset()
  out crashNotification() -> raiseAlarm
  state beforeFlight {
    on next [tp->alt > 0 m] -> airborne
  }
  composite state airborne initial = flying {
    on reset [ ] -> beforeFlight
    on next [tp->alt == 0 m && tp->speed == 0 mps] -> crashed
    state flying {
      on next [tp->alt == 0 m && tp->speed > 0 mps] -> landing
      on next [tp->speed > 200 mps] -> airborne
      on next [tp->speed > 100 mps] -> airborne
    }
    state landing {
      on next [tp->speed == 0 mps] -> landed
      on next [ ] -> landing
    }
    state landed {
    }
  }
  state crashed {
  }
}
```

Model or Code?

```
statemachine HierarchicalFlightAnalyzer initial = beforeFlight {
  in next(Trackpoint* tp)
  in reset()
  out crashNotification() -> raiseAlarm
  readable var int16 points = 0
  state beforeFlight {
    on next [tp->alt > 0 m] -> airborne
    exit { points += TAKEOFF; }
  }
  composite state airborne initial = flying {
    on reset [ ] -> beforeFlight { points = 0; }
    on next [tp->alt == 0 m && tp->speed == 0 mps] -> crashed
    state flying {
      on next [tp->alt == 0 m && tp->speed > 0 mps] -> landing
      on next [tp->speed > 200 mps] -> airborne { points += VERY_HIGH_SPEED; }
      on next [tp->speed > 100 mps] -> airborne { points += HIGH_SPEED; }
    }
    state landing {
      on next [tp->speed == 0 mps] -> landed
      on next [ ] -> landing { points--; }
    }
    state landed {
      entry { points += LANDING; }
    }
  }
  state crashed {
    entry { send crashNotification(); }
  }
}
```

Model or Code?

Does it really matter?

What is the difference?

Who cares?

GEMOC 2014

“To cope with complexity, modern software-intensive systems are often split in **different concerns**, which serve diverse stakeholder groups and thus must address a variety of **stakeholder concerns**. These different concerns are often associated with **specialized description languages** and technologies, which are based on **concern-specific** problems and **solution concepts**.”

GEMOC 2014

“different concerns

stakeholder concerns

specialized description languages

concern-specific solution concepts.”

GEMOC 2014

“

different concerns

stakeholder concerns

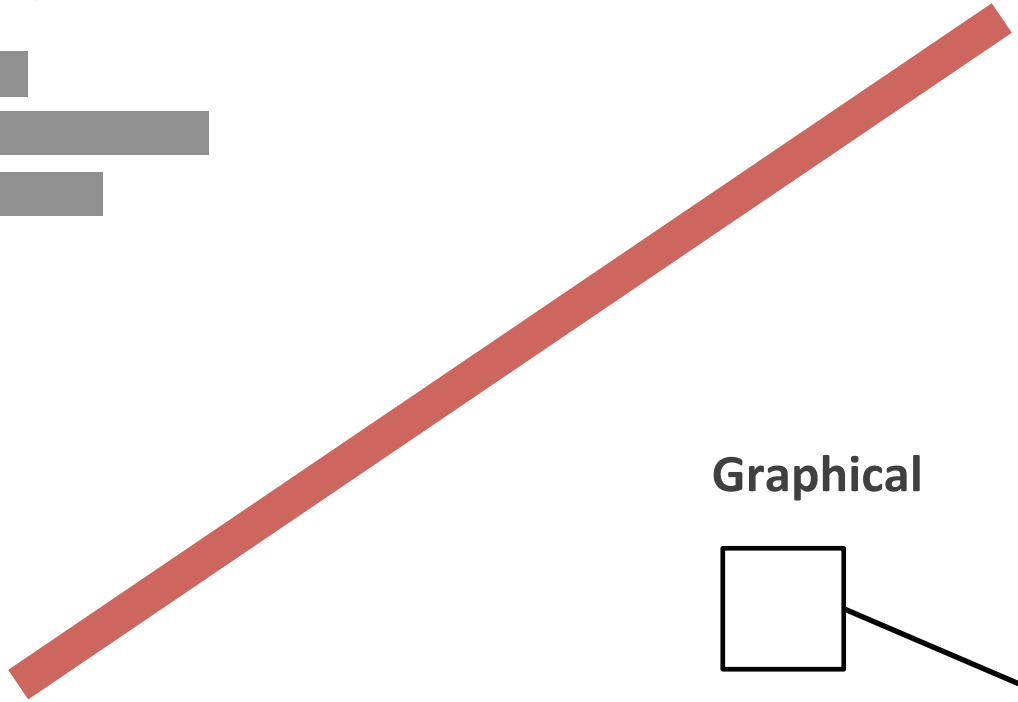
specialized description languages

concern-specific **notations.**

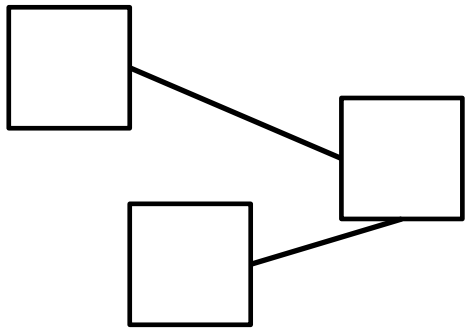
”

[Diverse Notations]

Regular Code/Text



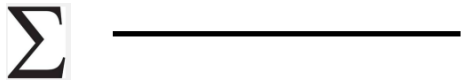
Graphical



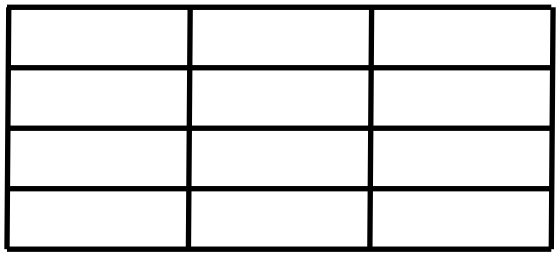
[Diverse Notations]

Regular Code/Text

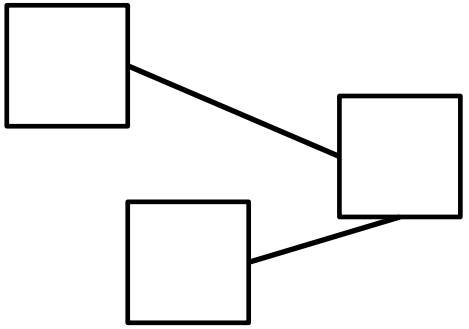
Mathematical



Tables



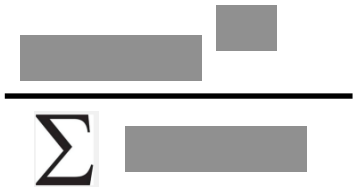
Graphical



[Diverse Notations]

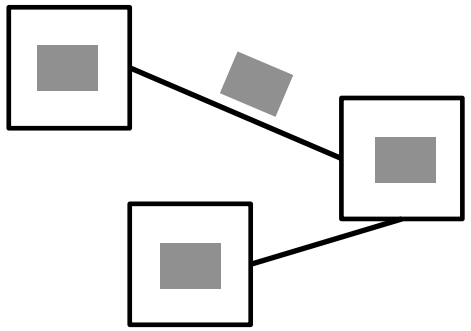
Regular Code/Text

Mathematical

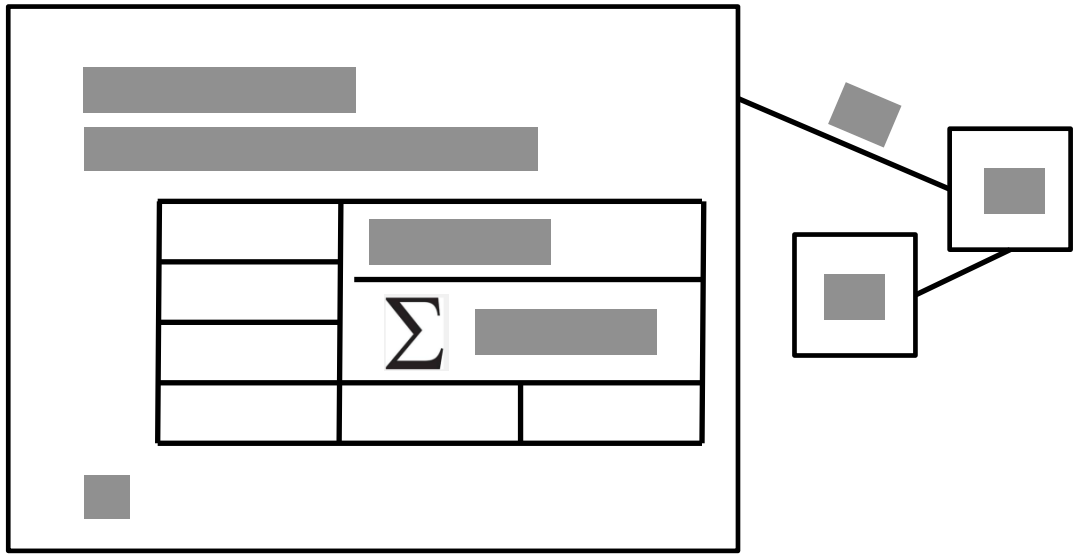


Tables

Graphical



[Diverse Notations]

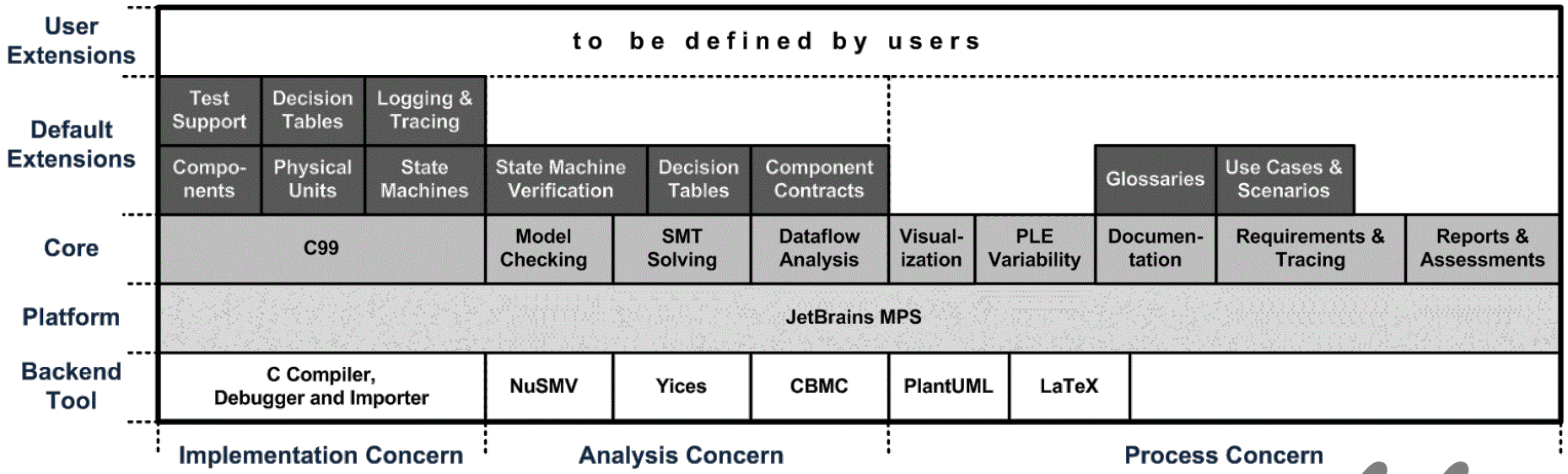


2

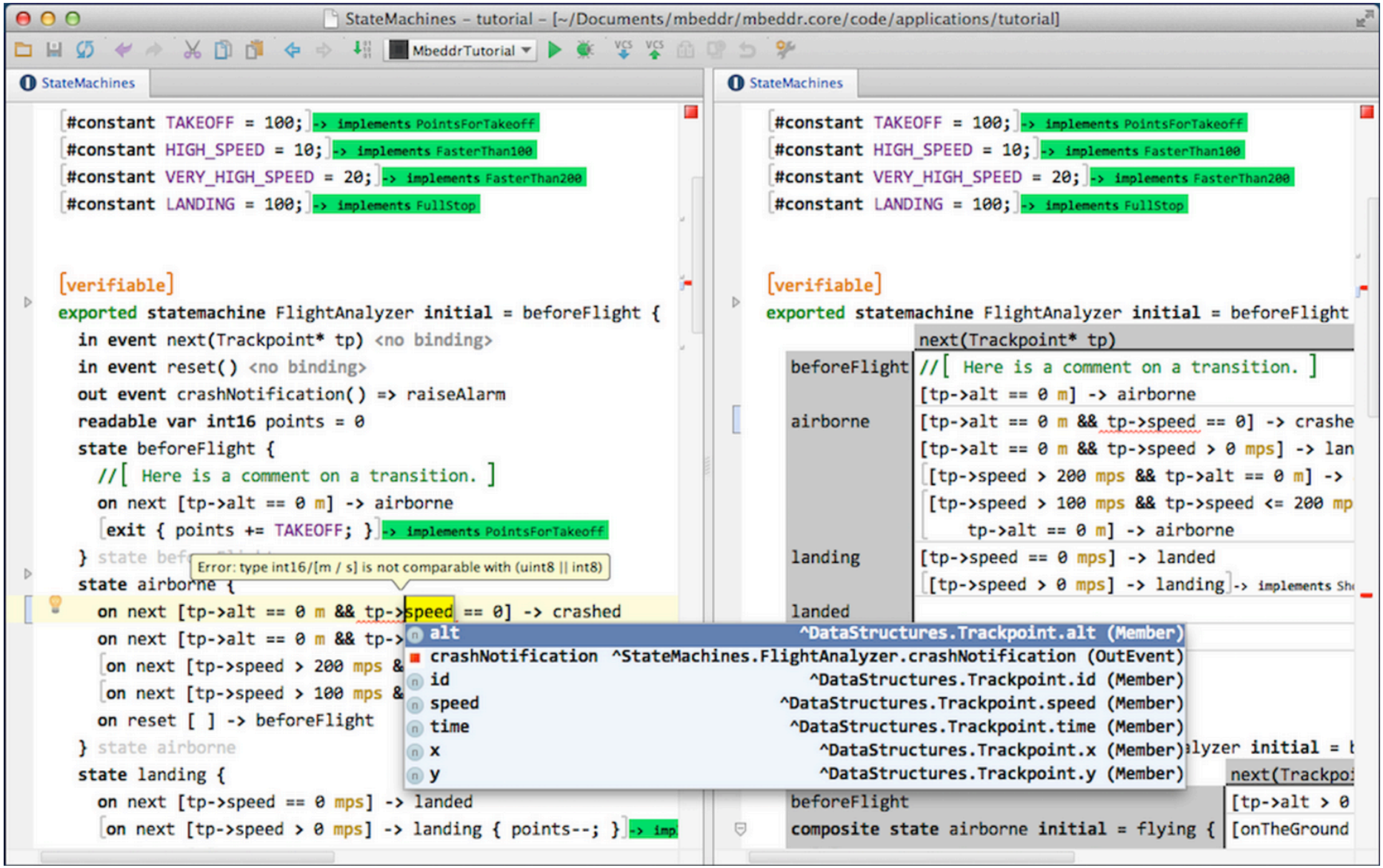
mbeddr

mbeddr

An extensible set of integrated languages for embedded software engineering.



“Specific Languages”


mbeddr
StateMachines – tutorial – [~/Documents/mbeddr/mbeddr.core/code/applications/tutorial]
MbeddrTutorial
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
Error: type int16/[m / s] is not comparable with (uint8 || int8)
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
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)
StateMachines
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 = b
next(Trackpo
beforeFlight
[tp->alt > 0
composite state airborne initial = flying {
[onTheGround

fortiss itemis

BMW Car IT

Bundesministerium
für Bildung
und Forschung

Research Project 2011 - 2013

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

Commercial Use and Extension

itemis
SIEMENS

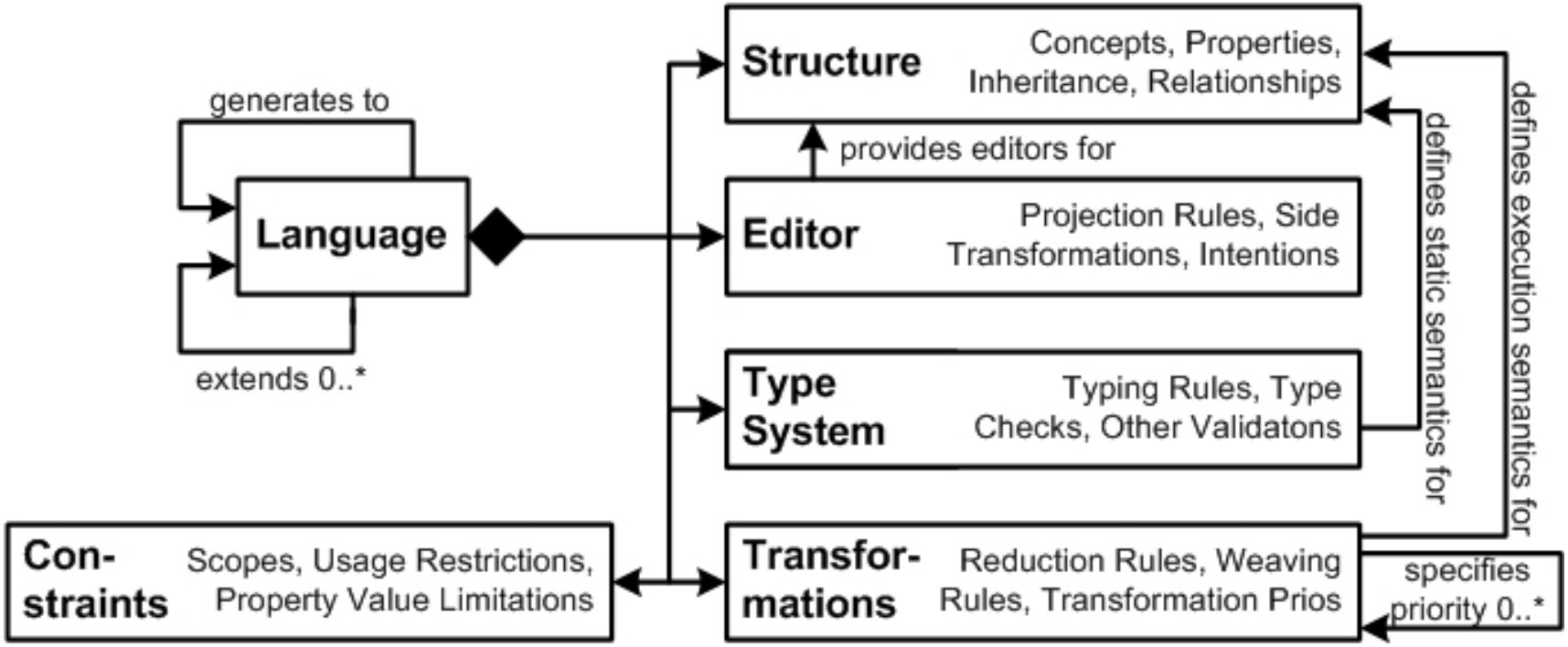
Research Platform

3

JetBrains MPS

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

[Language Workbench]

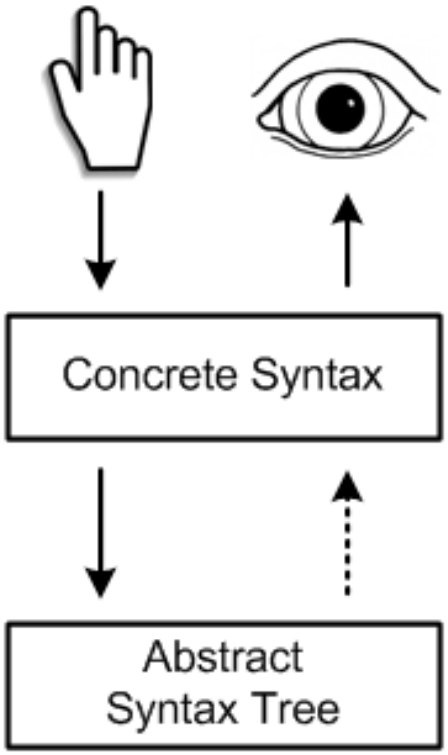


+ Refactorings, Find Usages, Syntax Coloring, Debugging, ...

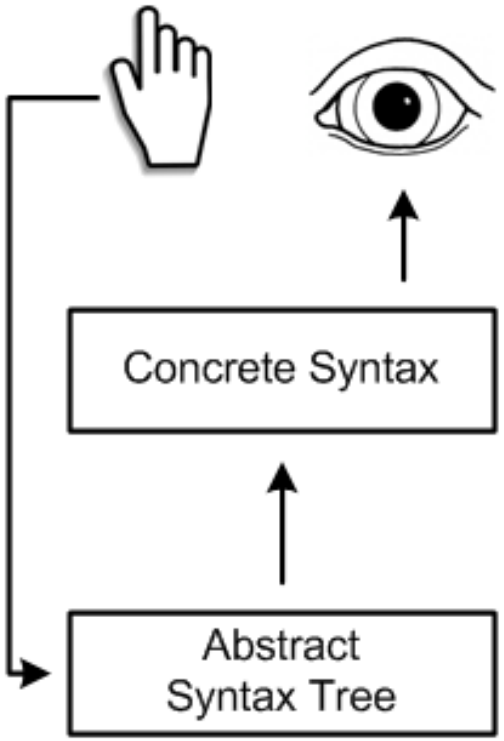
Projectional Editing

[Projectional Editing]

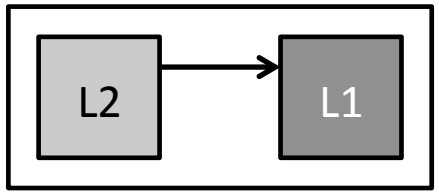
Parsing



Projectional Editing



[Projectional Editing]
Language Composition



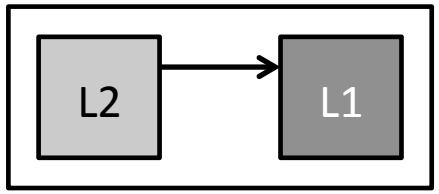
Separate Files

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.

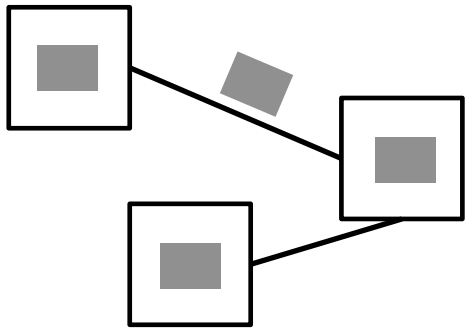
[Projectional Editing]
Syntactic Flexibility

Regular Code/Text

Mathematical

Tables

Graphical



[Projectional Editing]
Syntactic Flexibility

Regular Code/Text

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

Mathematical

```
double midnight2(int32 a, int32 b, int32 c) {
  return (-b + √(b² - Σ_{i=1}^{4} a * c)) / (2 * a);
} midnight2 (function)
```

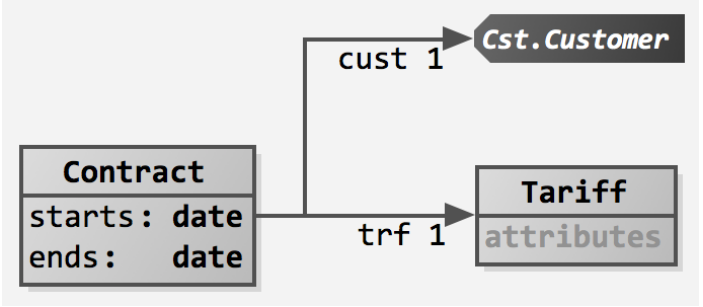
Tables

```
int16 decide(int8 spd, int8 alt) {
  return
```

	spd > 0	spd > 100
alt < 0	1	1
alt == 0	10	20
alt > 0	30	40
alt > 100	50	60

```
} decide (function)
```

Graphical



DEMO

Thank you!

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