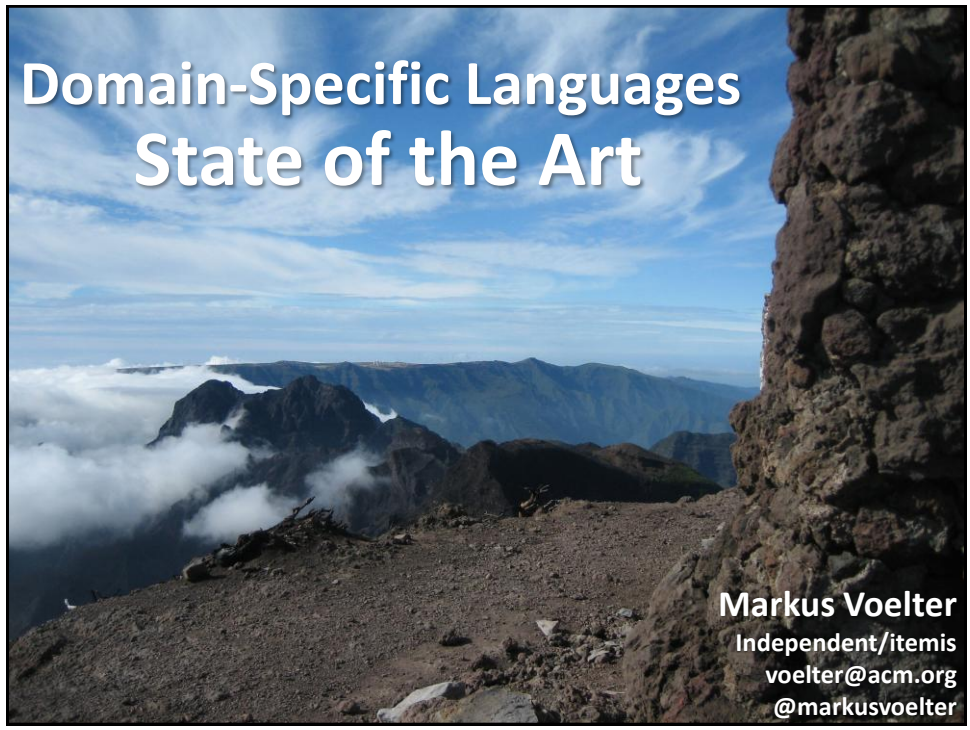
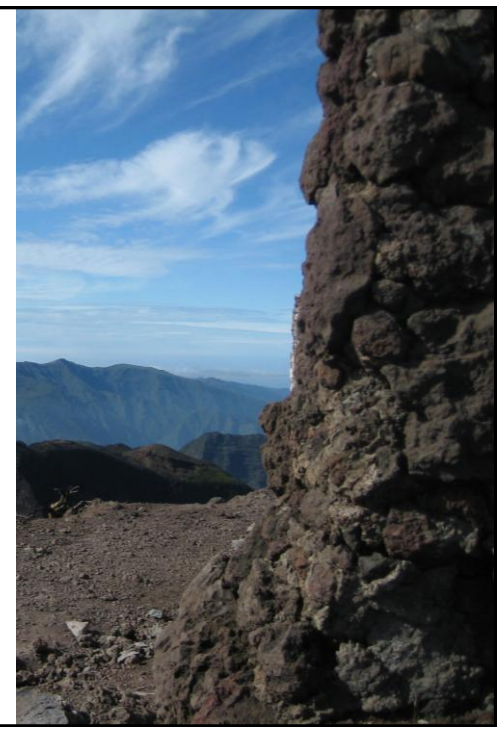
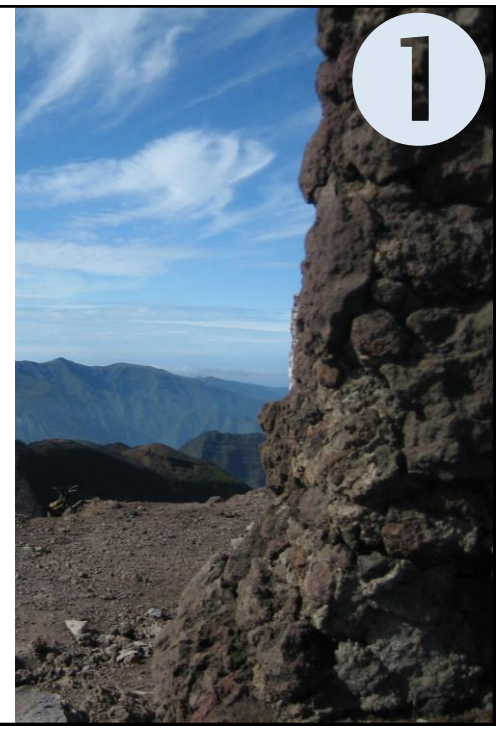


1. What are DSLs?
2. Examples
3. Tools
4. Benefits
5. A bit more theory
6. Where do we go from here



What are DSLs?



programming
started
close to the hardware

abstractions
~computing

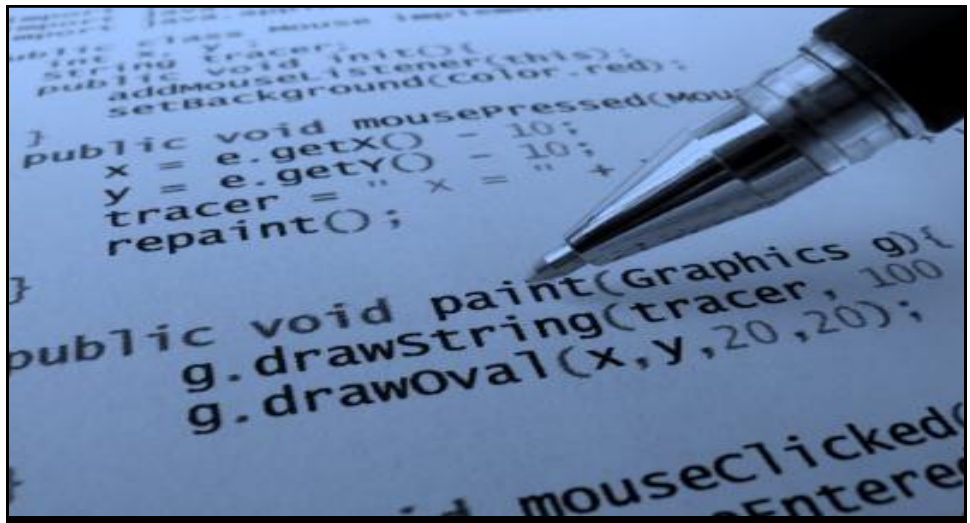
chips

abstractions
~computing

bits

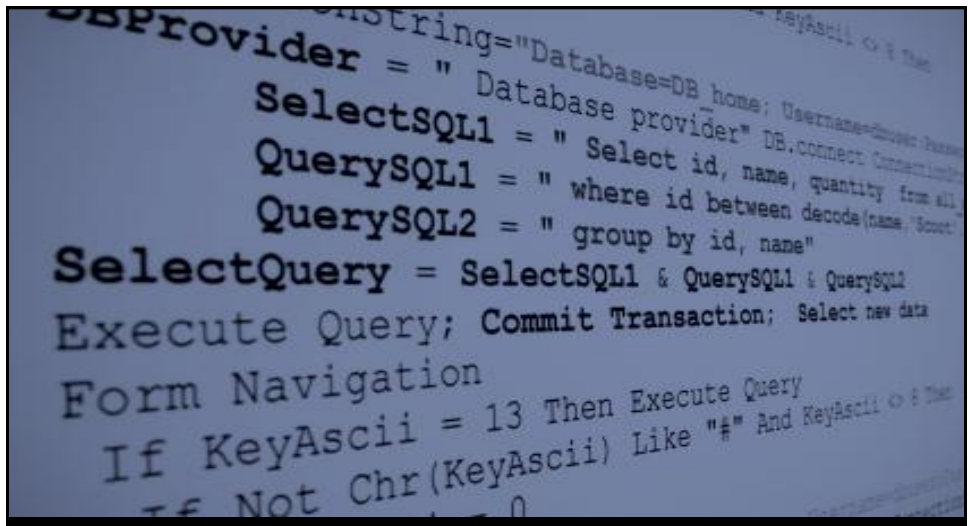
abstractions
~computing

c



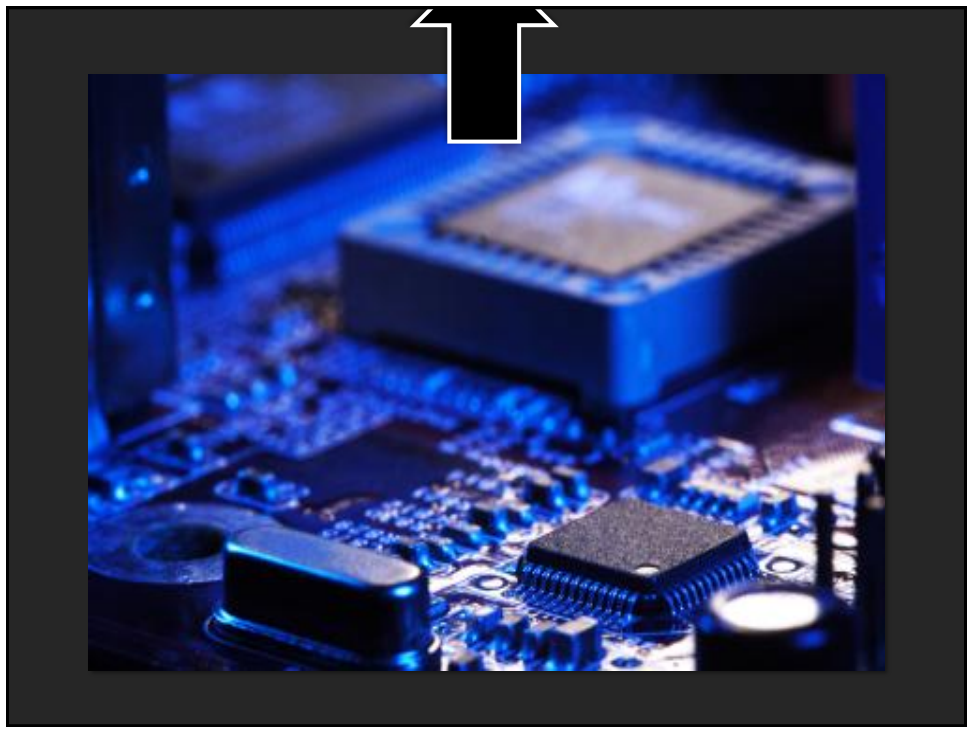
abstractions
~computing?

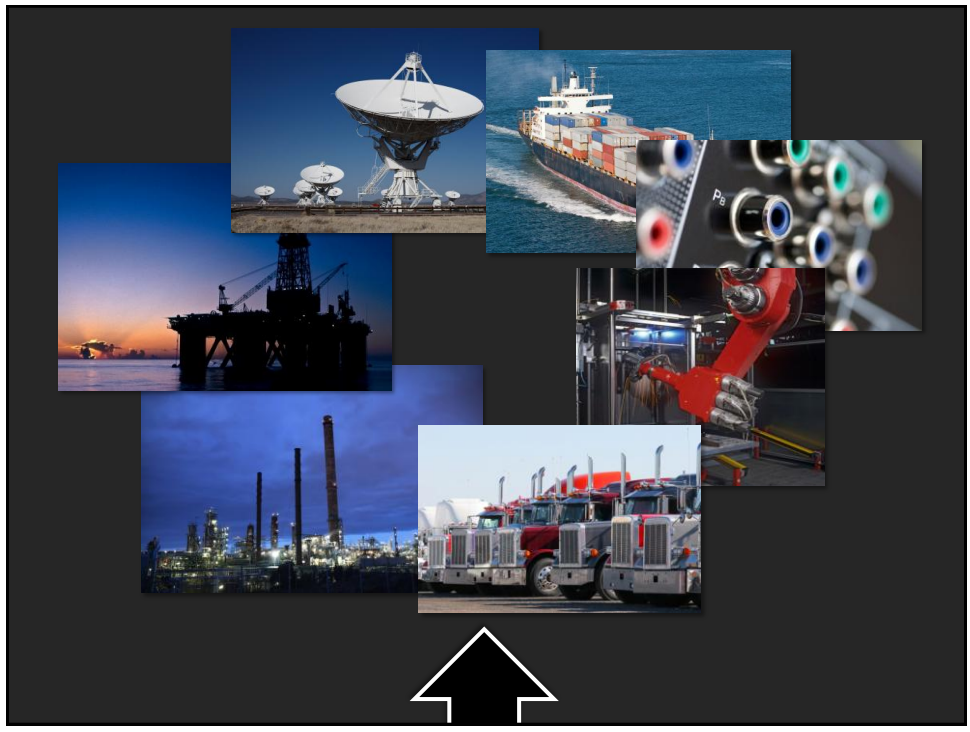
Java



abstractions
~computing?

SQL





general purpose

domain specific

tailor made

effective++

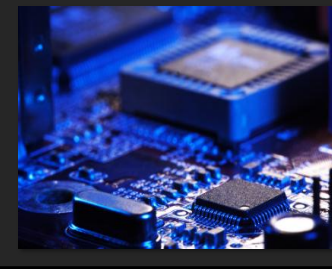
specialized, limited

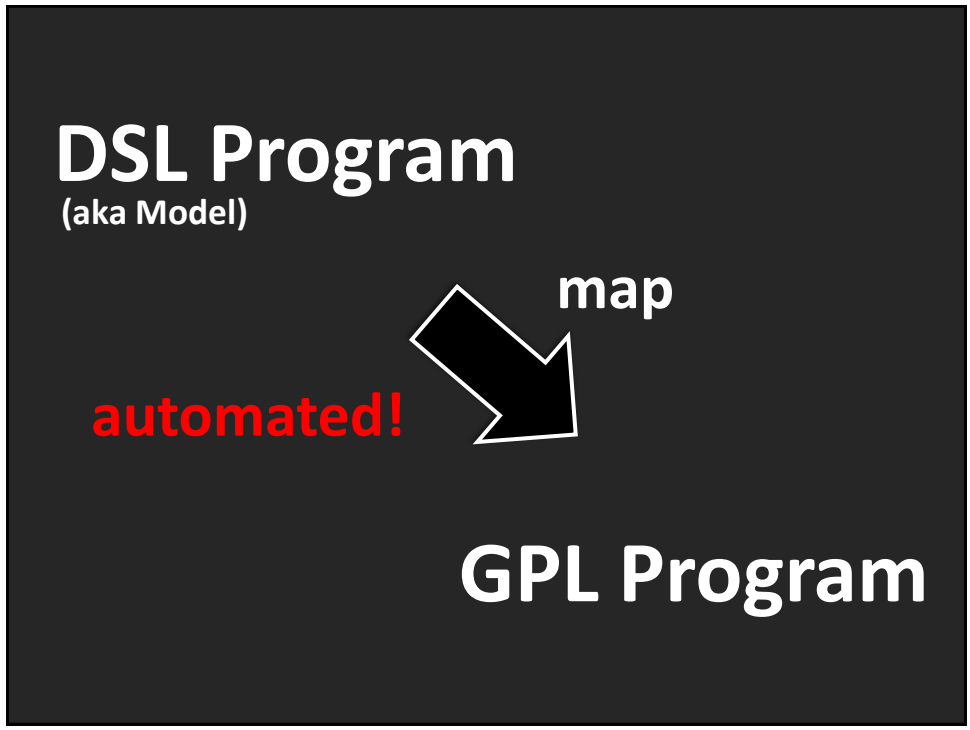
used by experts

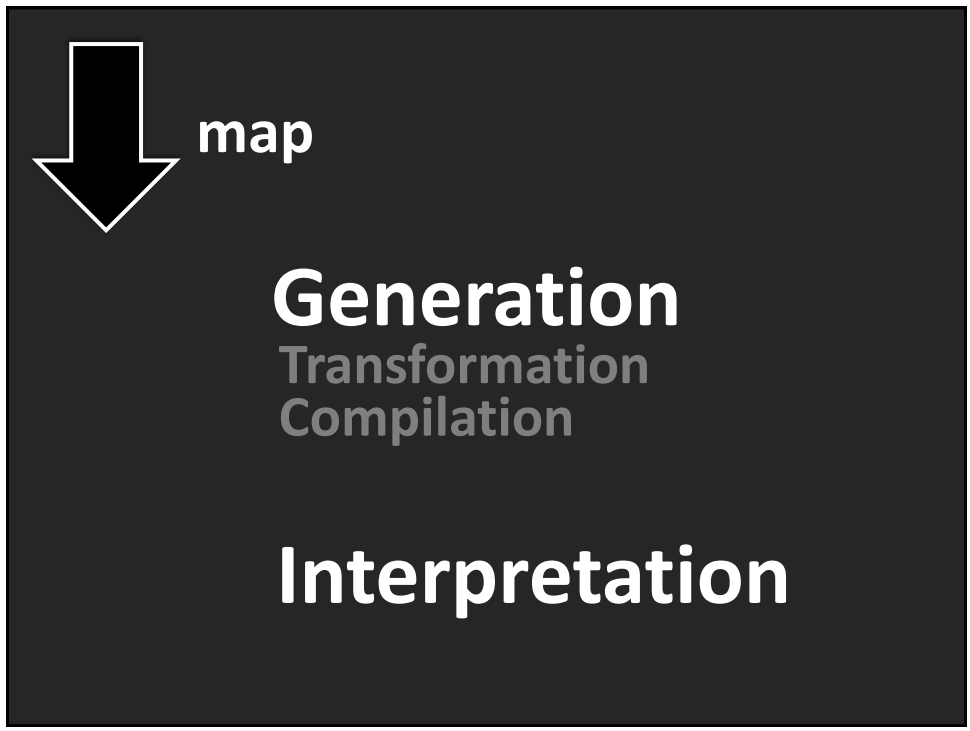
together with other specialized tools

A DSL is a **focussed**, **processable language** for describing a specific **concern** when building a system in a specific **domain**. The **abstractions** and **notations** used are natural/suitable for the **stakeholders** who specify that particular concern.

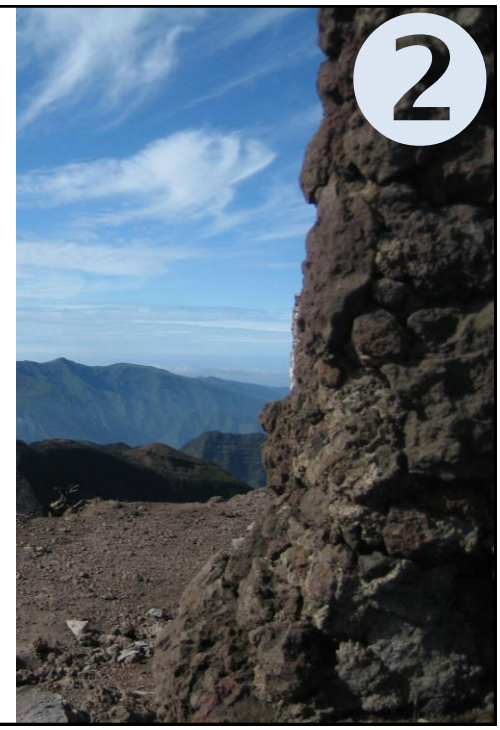
map




DSL Program
(aka Model)
map
automated!
GPL Program

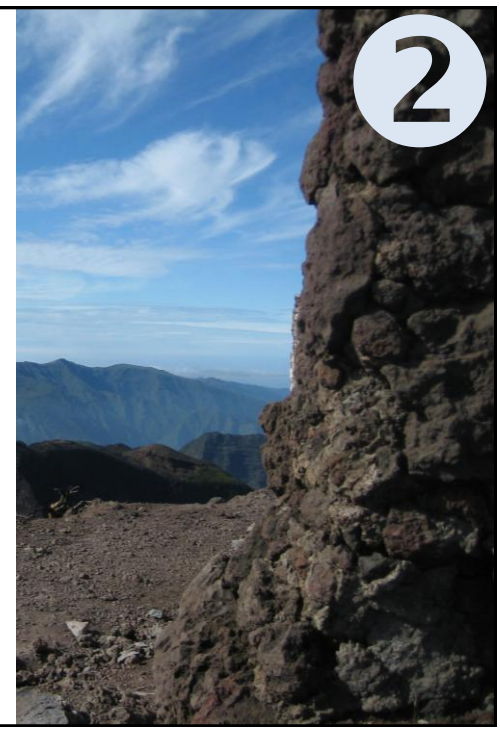

map
Generation
Transformation
Compilation
Interpretation

Examples

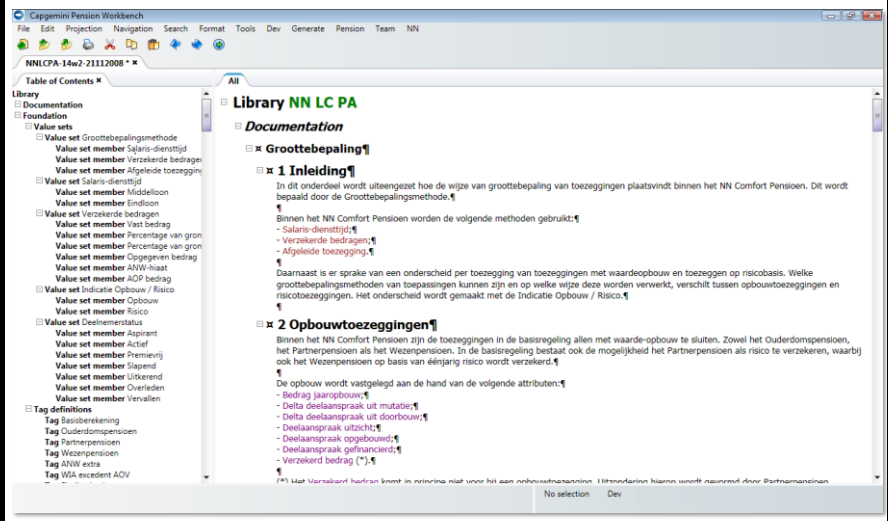


Examples

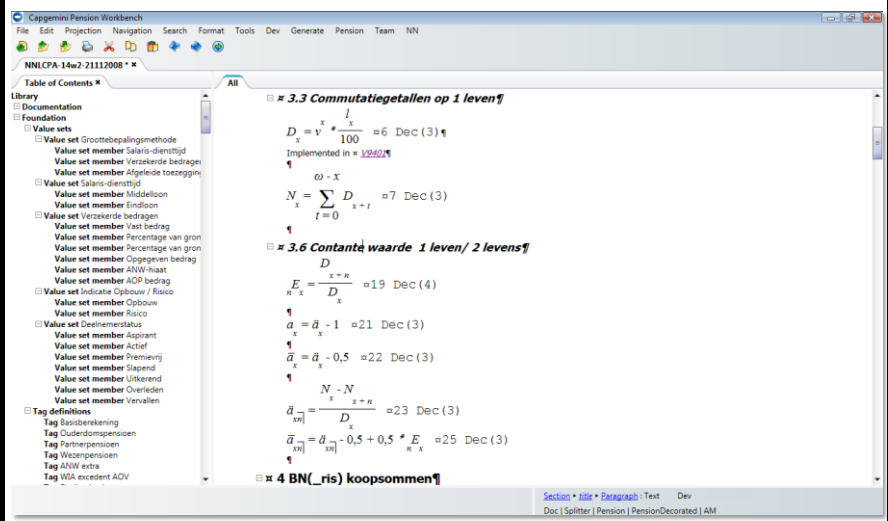
Pension Fund Specification



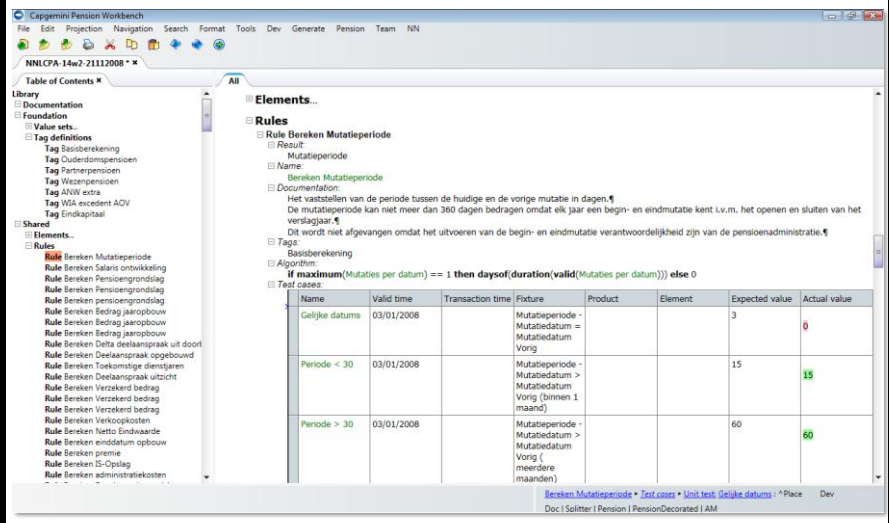
Textual Documentation



Insurance Mathematics



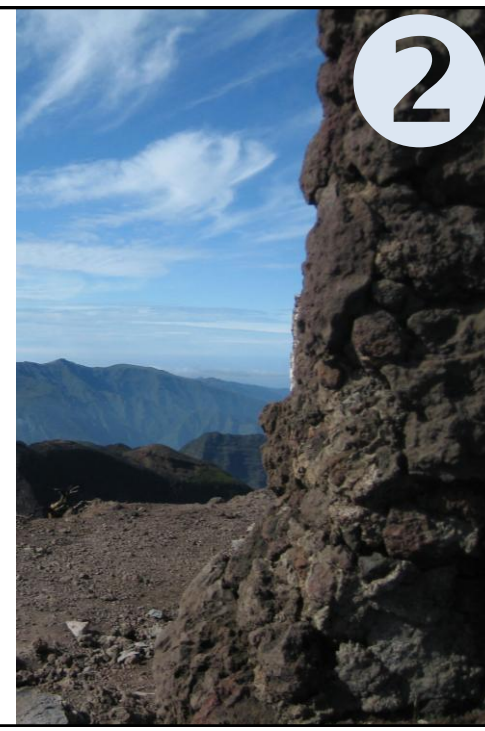
Calculation Rules and Tests



Examples

Math, Science

Java



Define physical Quantities

```
env block Aircraft

  v       : double [m/s] / current aircraft speed

  A       : double [m m] / cross area of the wing

  c_a     : double [1  ] / Auftriebsbeiwert

  c_w     : double [1  ] / Widerstandsbeiwert

  n_wings : double [1  ] / Number of Wings
```

In-IDE Evaluation of Formulas

```
function block Fundamental Stuff
  uses Aircraft, Environment

  The dynamic pressure p_dyn is calculated from the current air density rho and the square of the flight speed v

  exported p_dyn : double [Pa] = 1/2 * rho * v^2        [kg/(m s s)]
    v=0 rho=1.225  -> 0
    v=10 rho=1.225 -> 61.25
    v=20 rho=1.225 -> 245
```

```
function block Stuff on the Wings
  uses Environment, Aircraft, Fundamental Stuff

  Aus dem Staudruck p_dyn lässt sich dann der aktuelle Auftrieb F_A berechnen; die Form wird
   c_a beschrieben  und die Fläche durch  A

  exported F_A : double [N] = p_dyn * A * c_a        [m kg/(s s)]
    c_a=0.3 p_dyn=61.25 A=2  -> 183.75
    p_dyn=61.25 A=10 c_a=0.6 -> 367.5

  Auch der Widerstand F_W berechnet sich entsprechend mit Hilfe des Beiwertes c_w :

  exported F_W : double [N] = p_dyn * A * c_w        [kg m/(s s)]

  Angenommen wir haben mehrere Flügel n_wings am Flugzeug, dann berechnet sich der Autrieb
```

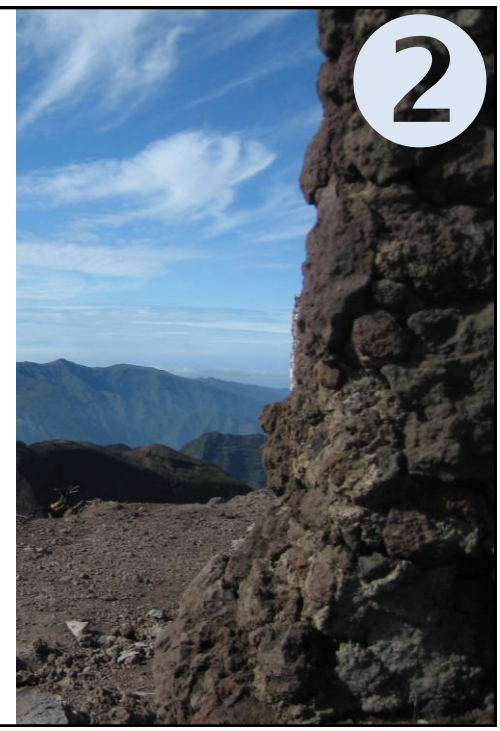
Integration with Java

```
import blocks Environment
              Aircraft
              Stuff on the Wings
public class TestClass extends <none> implements <none> {
  <<static fields>>

  <<static initializer>>
  <<fields>>
  <<properties>>
  <<initializer>>
  public TestClass() {
    <no statements>
  }

  public void m() {
    values air = (| Environment.rho = 1.225 |);
    values planeStatic = (| Aircraft.A = 10, Aircraft.c_a = 0.5, Aircraft.c_w = 0.3, Aircraft.v = 100 |);
    double auftrieb = Stuff on the Wings.F_A (air, planeStatic);
    System.err.println(auftrieb);
  }

  <<static methods>>

  <<nested classifiers>>
}
```

Examples

Fountains

xtext



CONTEXT

SOLUTION

Hardware Structure

```
feature BasicOnePump
    pump compartment cc1
        static compressor c1

feature AtLeastOneZone extends BasicOnePump
    water compartment comp1
        pumped by c1
        compartment levelsensor ct_f1
        light l_f1

feature[f] SuperPowerCompartment
    water compartment adds to f
        superPowerMode

feature WithAlarm
    level alarm a1

fountain StdFountain extends AtLeastOneZone
```

SOLUTION

Behaviour

```
pumping program P1 for AtLeastOneZone + WithAlarm +
                       SuperPowerCompartment[f=comp1]  {
    parameter defaultWaterLevel : int
    parameter superWaterLevel: int
    event superPowerTimeout

    init {
        set comp1->targetHeight = defaultWaterLevel
    }

    start:
        on (comp1->needsPower == true) && !(comp1->isPumping) {
            do comp1->pumpOn
        }
        on comp1->enough {
            do comp1->pumpOff
        }
        on comp1.superPumping->turnedOn {
            set comp1->targetHeight = superWaterLevel
            raise event superPowerTimeout after 20
        }
        on comp1.superPumping->turnedOff or superPowerTimeout {
            set comp1->targetHeight = defaultWaterLevel
        }
}
```

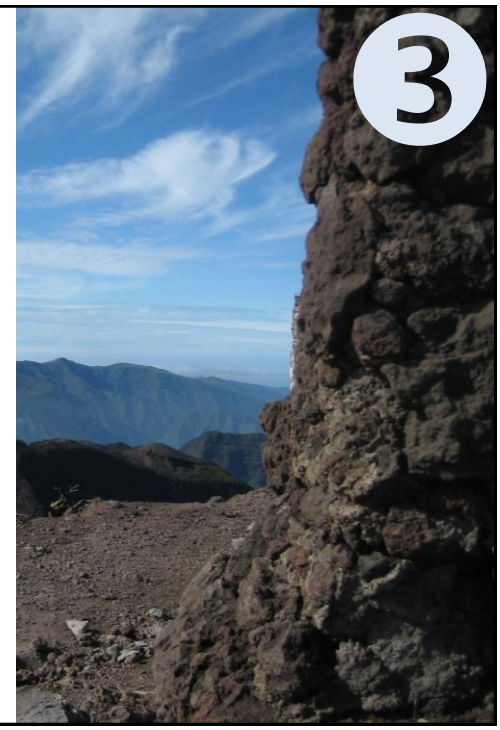
SOLUTION

Plus:

In-IDE Simulator

Unit Test Support

Tools



Tooling!

Editor, Debugger, Testing, Groupware, Scalable, „All in Eclipse“

Tools Tooling

Language Definition Tools

abstract syntax, concrete syntax, constraints

Editor Frameworks

Transformation Languages

Code Generation Tools

xtext

xtext

- 2.0 almost out
- improved performance
- Xbase: expressions for reuse
- Xtend2: „Better Java“, with support for Xpand-like templates

2.0 almost out

Improved performance

Unified generate/compile/build

Debug MPS in MPS

Tables in the editor

(Diagrams planned or 2.1)

Available!

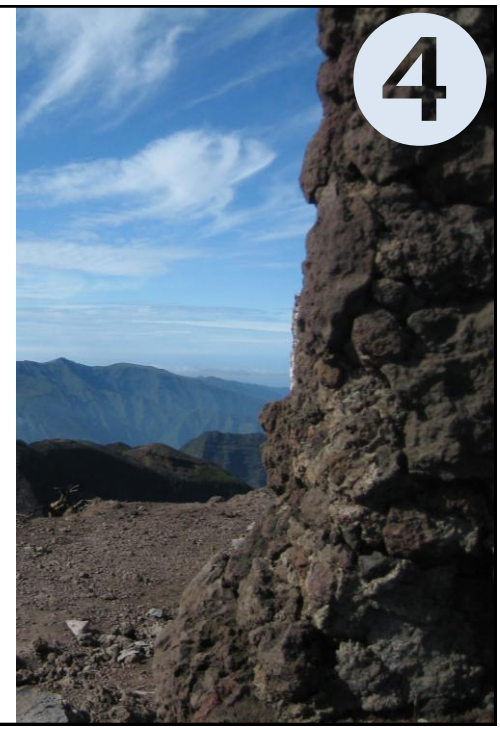
Way More:

- Spoofax
- Rascal
- oomega
- The Whole Platform

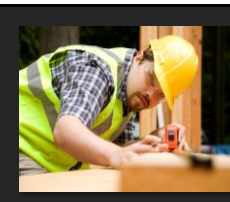
see also

- **languageworkbenches.net**

Benefits



Automation
faster, deterministic



Increased Quality

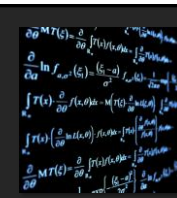
well defined structures allthrough the system

Meaningful Validation

more semantics in the model

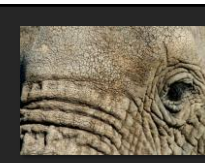
Capture Domain Knowledge

formalized into languages and models



Suitable Notations

textual, graphical, tabular

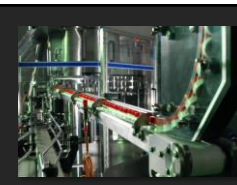


Technology Independence

generate „technology glue code“

Abstraction w/o Runtime Overhead

generator „optimizes away“



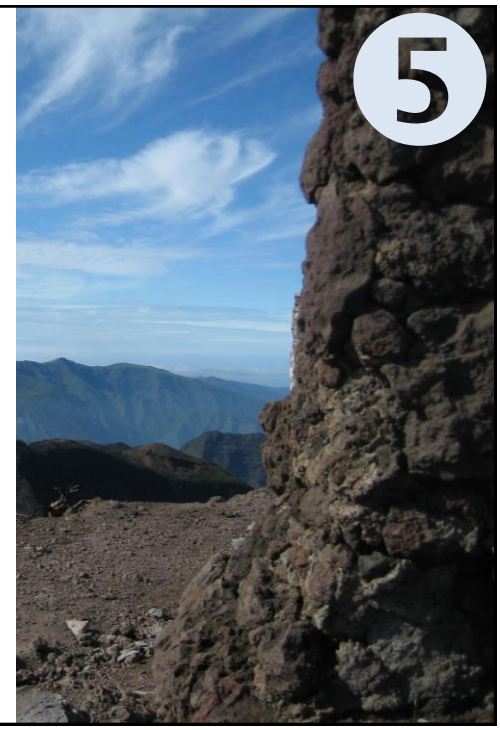
Capture Implementation Strategy

in the generators

Everything is a model

including for example hardware (some) hardware

A bit more theory



Domain Specific Languages

A DSL is a **focussed**, **processable language** for describing a specific **concern** when building a system in a specific **domain**. The **abstractions** and **notations** used are natural/suitable for the **stakeholders** who specify that particular concern.

Linguistic Abstraction

What's the Problem here?

```
// A
int[] arr = ...
for (int i=0; i<arr.size(); i++) {
    sum += arr[i];
}
```

```
// B
List<int> l = ...
for (int i=0; i<arr.size(); i++) {
    l.add( arr[i] );
}
```

Much better with new linguistic abstraction

```
// A'
for (int i in arr) {
    sum += i;
}
```

```
// B'
seqfor (int i in arr) {
    l.add( arr[i] );
}
```

No sophisticated analysis required to understand the semantics of a contruct and treat it "correctly".

Linguistic Abstraction: Bad

What's this? And what's the Problem?

```
var linefollower_states_enum linefollower_currentstate = linefollower_states_enum::STATE_INITIALIZING;

enum linefollower_events_enum { EVENT_INITIALIZED, EVENT_BUMPED, EVENT_BLOCKED, EVENT_UNBLOCKED }

enum linefollower_states_enum { STATE_INITIALIZING, STATE_PAUSED, STATE_RUNNING, STATE_CRASH }

void linefollower_execute( linefollower_events_enum event ) {
  if ( linefollower_currentstate == linefollower_states_enum::STATE_INITIALIZING ) {
    if ( event == linefollower_events_enum::EVENT_INITIALIZED ) {
      if ( true ) {
        linefollower_currentstate = linefollower_states_enum::STATE_RUNNING;
        return;
      }
    }
  }
  if ( linefollower_currentstate == linefollower_states_enum::STATE_PAUSED ) {
    if ( event == linefollower_events_enum::EVENT_UNBLOCKED ) {
      if ( true ) {
        linefollower_currentstate = linefollower_states_enum::STATE_RUNNING;
        return;
      }
    }
  }
  if ( linefollower_currentstate == linefollower_states_enum::STATE_RUNNING ) {
    if ( event == linefollower_events_enum::EVENT_BLOCKED ) {
      if ( true ) {
        linefollower_currentstate = linefollower_states_enum::STATE_PAUSED;
        int16 i = 1;
        return;
      }
    }
    if ( event == linefollower_events_enum::EVENT_BUMPED ) {
      if ( true ) {
        linefollower_currentstate = linefollower_states_enum::STATE_CRASH;
        return;
      }
    }
  }
}
```

Linguistic Abstraction: Better

Much better!

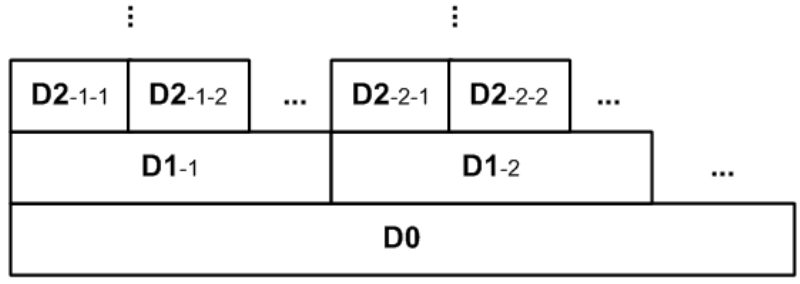
```
statemachine linefollower {
  event initialized;
  event bumped;
  event blocked;
  event unblocked;
  initial state initializing {
    initialized [true] -> running
  }
  state paused {
    entry int16 i = 1;
    unblocked [true] -> running
  }
  state running {
    blocked [true] -> paused
    bumped [true] -> crash
  }
  state crash {
    <<transitions>>
  }
}
```

Linguistic Abstraction: Best?

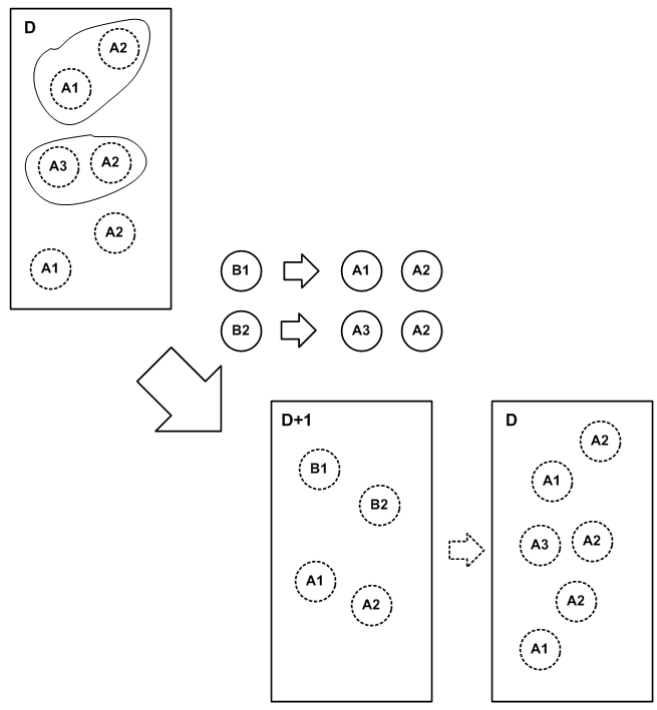
Much better!

linefollower	initializing	paused	running	crash
initialized	true running			
bumped			true crash	
blocked			true paused	
unblocked		true running		

Domains are Hierarchical



Creating Linguistic Abstractions

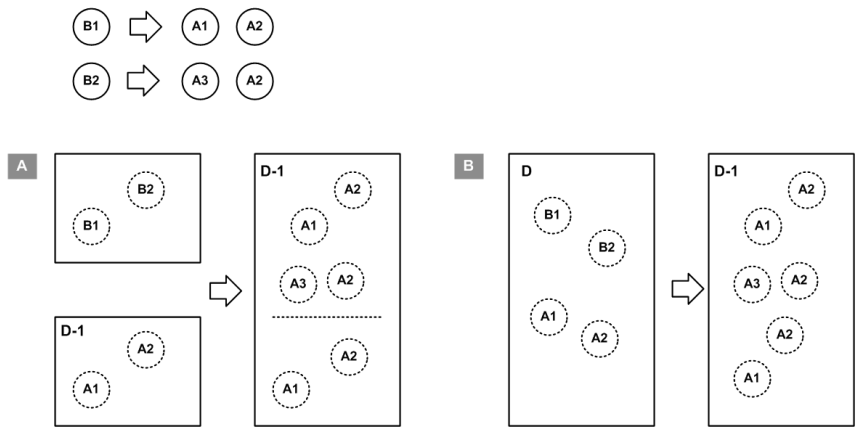


Domain Specific Languages

A DSL is a **language** at D that provides **linguistic abstractions** for **common patterns and idioms** of a language at D-1 when used within the domain D.

A good DSL does **not** require the use of patterns and idioms to express **semantically interesting** concepts in D. Processing tools do not have to do "semantic recovery" on D programs.

Cascading vs. Extension



Cascading vs. Extension

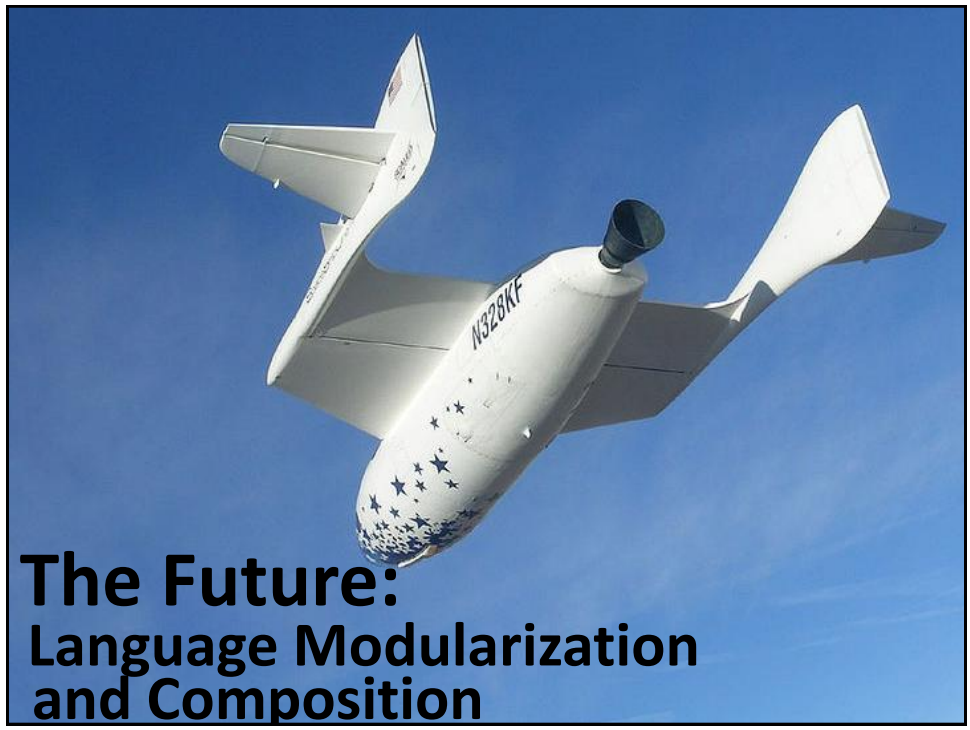
```
constant int FAST = 40;

var avg(int, 10) currentSonar = 250;

doc Statemachine to manage the transition between the initialization and operation p
statemachine linefollower {
  event initialized;
  event bumped;
  event blocked;
  event unblocked;
  initial state initializing {
    initialized [true] -> running
  }
  state paused {
    entry int16 i = 1;
    unblocked [true] -> running
  }
  state running {
    blocked [true] -> paused
    bumped [true] -> crash
  }
  state crash {
    <<transitions>>
  }
}



doc This procedure actually configures the motors based on the speed values passed i
void updateMotorSettings( int left, int right ) {
  nxt_motor_set_speed(MOTOR_PORT_T::NXT_PORT_C, left, 1);
  nxt_motor_set_speed(MOTOR_PORT_T::NXT_PORT_B, right, 1);
}
```

Cascading vs. Extension

```
var avg(int, 10) currentSonar = 250;
```

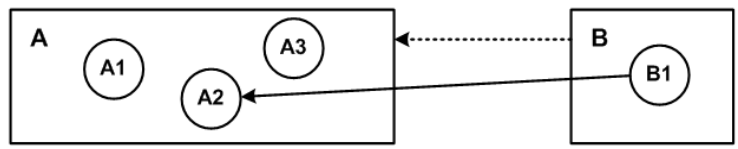
linefollower	initializing	paused	running	crash
initialized	true running			
bumped			true crash	
blocked			true paused	
unblocked		true running		

```
initialize {
  <abstract concept>
  ecrobot_set_light_sensor_active(SENSOR_PORT_T::NXT_PORT_S1);
  ecrobot_init_sonar_sensor(SENSOR_PORT_T::NXT_PORT_S2);
  event linefollower:initialized
}

terminate {
  ecrobot_set_light_sensor_inactive(SENSOR_PORT_T::NXT_PORT_S1);
}

task sonartask cyclic prio = 1 every = 100 {
  currentSonar =/ ecrobot_get_sonar_sensor(SENSOR_PORT_T::NXT_PORT_S2);
      <<hidden, only visible in projection level 'all'>>
}

doc This is the cyclic task that is called every 1ms to do the actual control of the
task run cyclic prio = 1 every = 2 {
  stateswitch linefollower
    state running
      int8 bump = 0;
      bump = ecrobot_get_touch_sensor(SENSOR_PORT_T::NXT_PORT_S3);
      if ( bump == 1 ) {
        event linefollower:bumped
        terminate;
```



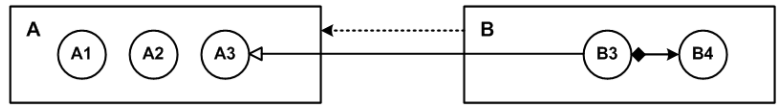
The Future:
Language Modularization and Composition

Langugae Combination



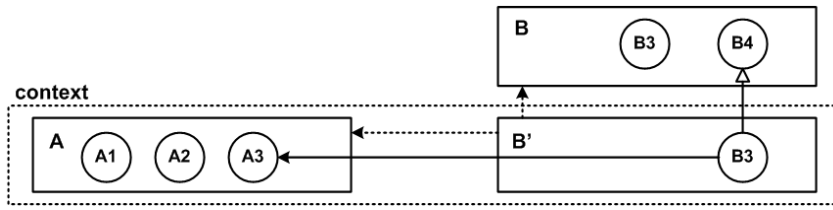
When DSL (for several concerns) are developed from scratch, as a group, then dependencies between the concerns can be materialized as dependencies between the languages and the language concepts

Langugae Extension



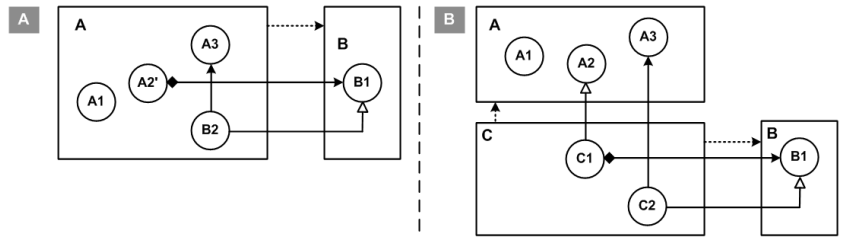
A language B extends another language A if B contains additional language concepts. This means that for programs written in B, all concepts from A are available, plus those defined in B.

Langugae Reuse



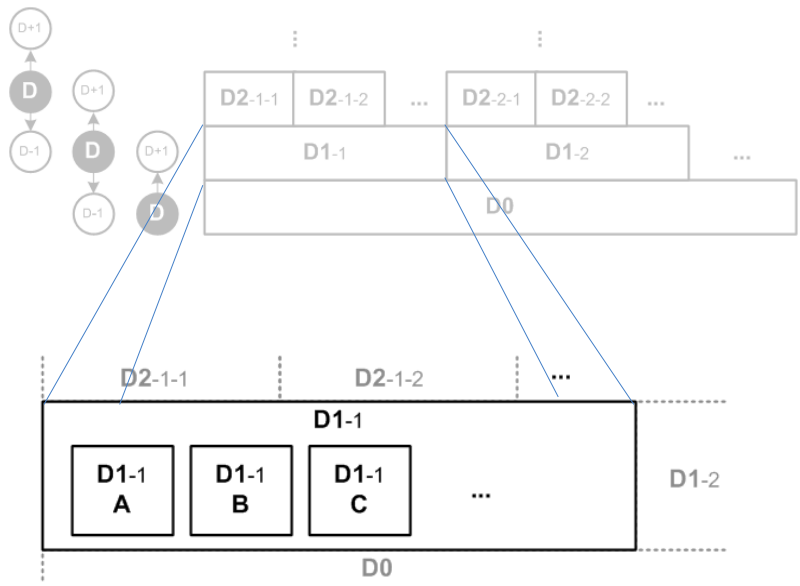
A language has been developed to be used in contexts not known at the time of development. No dependencies allowed! The reusable language has to be extended so it can reference concepts from languages in that context.

Langugae Embedding

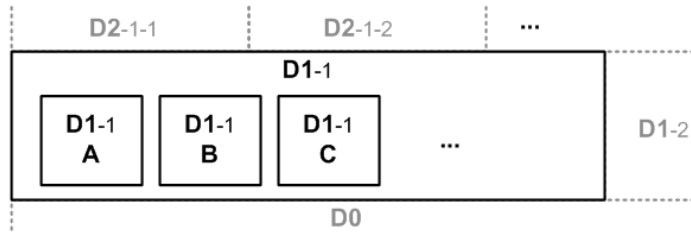


Composition is a special case of reuse, where the reused language is syntactically embedded into languages from the context.

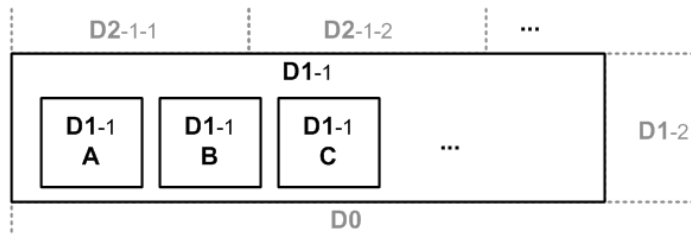
Several Concerns in a Domain



Separate Concerns: Viewpoints

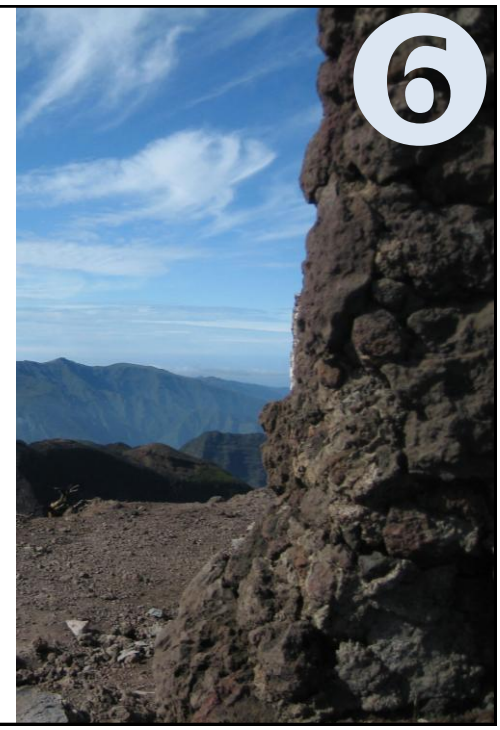


Separate Concerns: Viewpoints



A: mixed
B: separate Viewpoints

Where do we go from here



We don‘t want to model, we want to program!

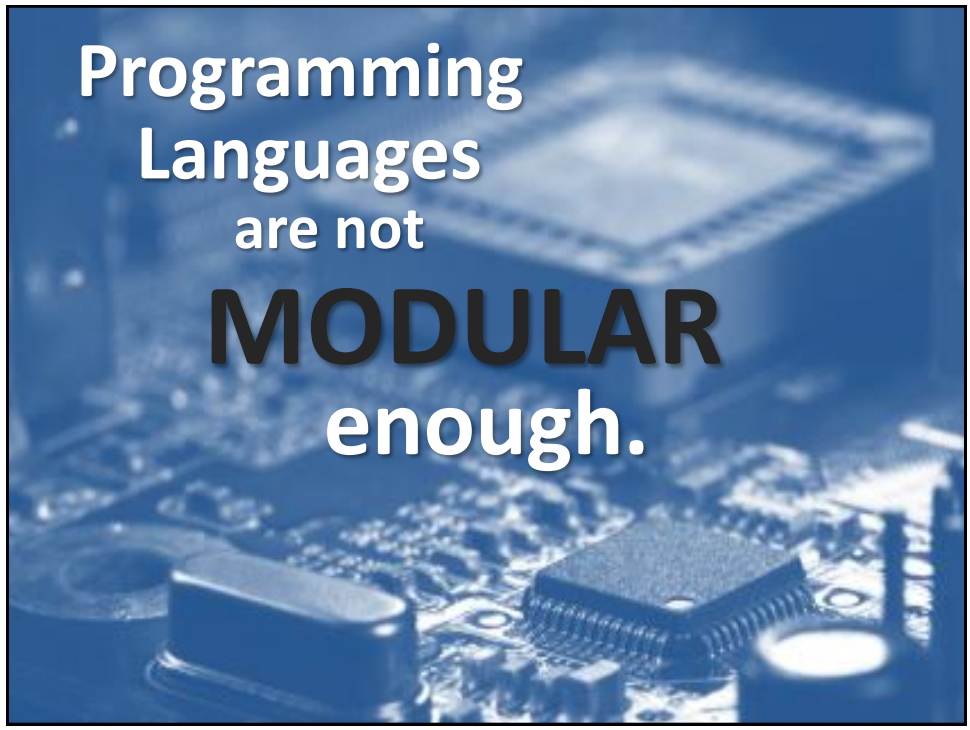
… at different levels of abstraction
… from different viewpoints
… integrated!

We don‘t want to
model,
we want to
program!

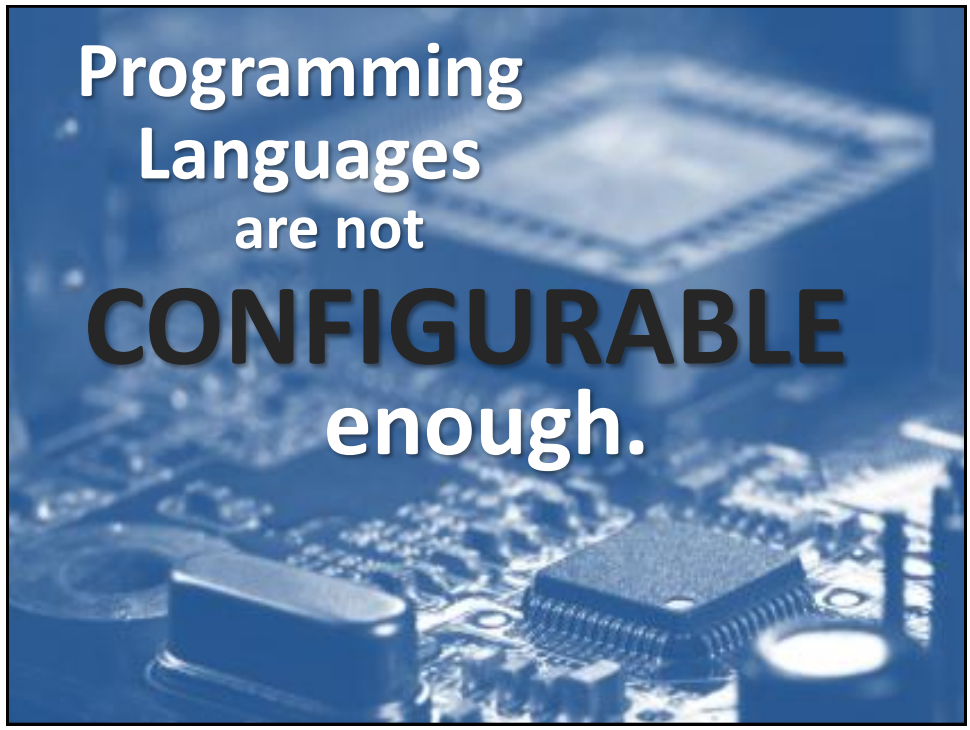
… with different degrees of
domain-specificity
… with suitable notations
… with suitable expressiveness

We don‘t want to
model,
we want to
program!

And always:
precise and tool processable


Programming
Languages
are not
MODULAR
enough.

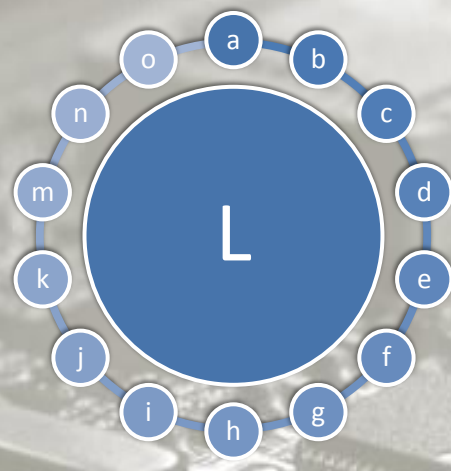
Programming
Languages
are not
COMPOSABLE
enough.


Programming
Languages
are not
CONFIGURABLE
enough.

Programming
Languages
are not
ADAPTABLE
enough.

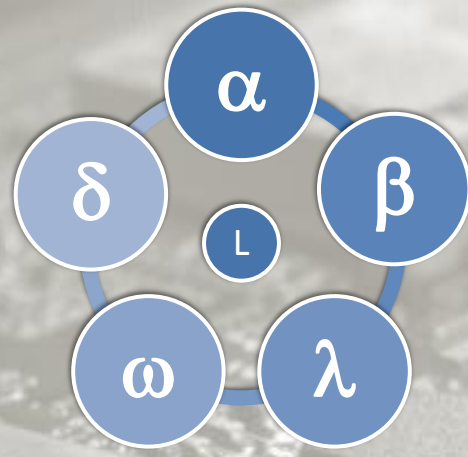
Programming
Language Syntax
is not
FLEXIBLE
enough.

Big Language?



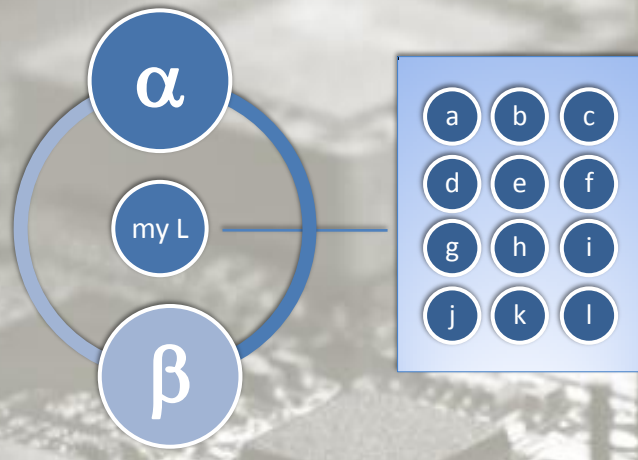
with **many** first class concepts!

Small Language?

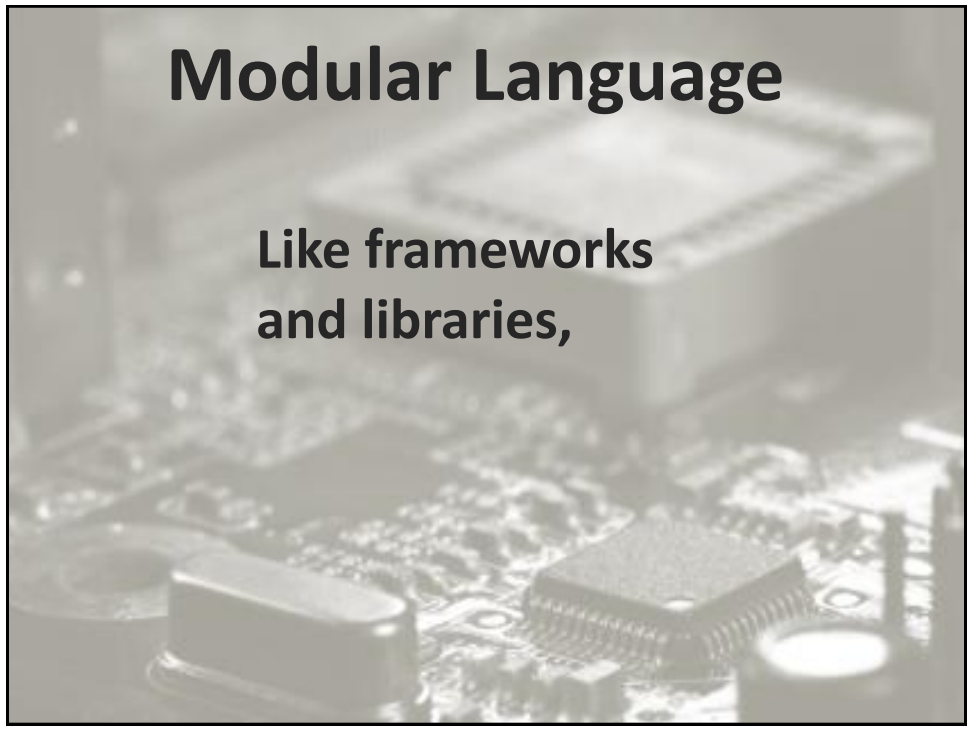


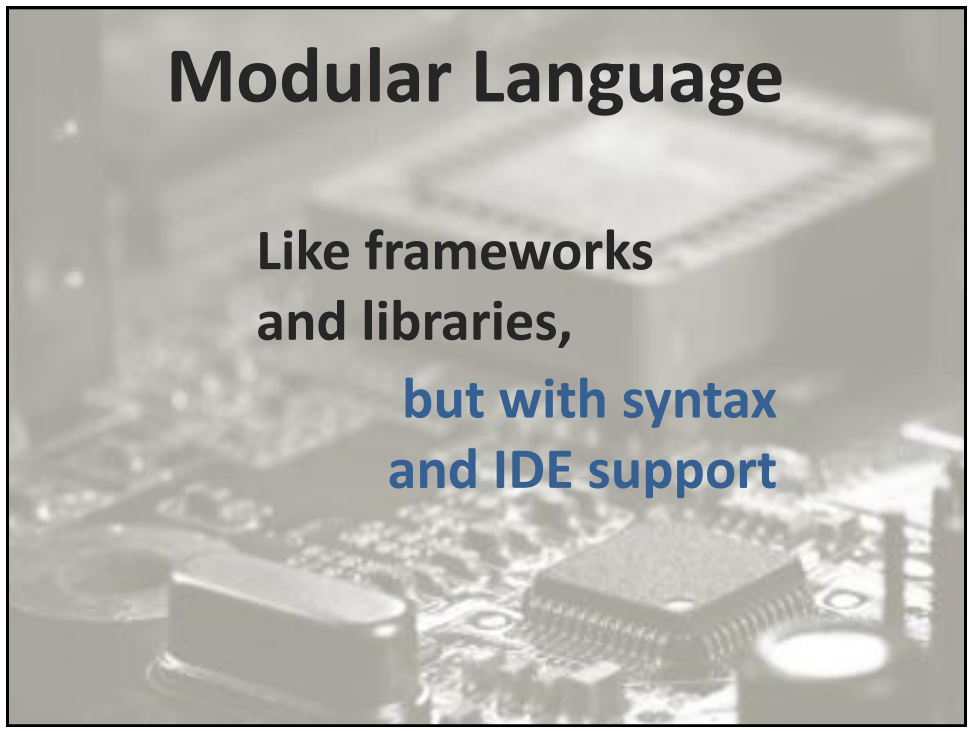
with a few, orthogonal and poweful concepts

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

C + Tasks + State Machines

```
doc This module represents the code for the line follower lego robot. It has a coupl
module main imports OsekKernel, EcAPI, BitLevelUtilies {

  constant int WHITE = 500;

  constant int BLACK = 700;

  constant int SLOW = 20;

  constant int FAST = 40;

  doc Statemachine to manage the t
  statemachine linefollower {
    event initialized;
    initial state initializing {
      initialized [true] -> running
    }
    state running {

    }
  }

  initialize {
    ecrobot_set_light_sensor_activ
    event linefollower:initialized
  }
```

```
doc This is the cyclic task that is called every 1ms to do the actual contro
task run cyclic prio = 2 every = 2 {
  stateswitch linefollower
    state running
      int32 light = 0;
      light = ecrobot_get_light_sensor(SENSOR_PORT_T::NXT_PORT_S1);
      if ( light < ( WHITE + BLACK ) / 2 ) {
        updateMotorSettings(SLOW, FAST);
      } else {
        updateMotorSettings(FAST, SLOW);
      }
    default
      <noop>;
}

doc This procedure actually configures the motors based on the speed values
void updateMotorSettings( int left, int right ) {
  nxt_motor_set_speed(MOTOR_PORT_T::NXT_PORT_C, left, 1);
  nxt_motor_set_speed(MOTOR_PORT_T::NXT_PORT_B, right, 1);
}
```

C + Components

```
exported interface MotorControl {
  void stop(   );
  void setLeftSpeed( int8 speed );
  void setRightSpeed( int8 speed );
}
```

```
exported component Motors {
  provides motorControl : MotorControl;
}
```

```
exported component implementation MotorsNXT : Motors {

  procedure void motorControl.stop( ) {
    nxt_motor_set_speed(MOTOR_PORT_T::NXT_PORT_B, 0, 1);
    nxt_motor_set_speed(MOTOR_PORT_T::NXT_PORT_C, 0, 1);
  }

  procedure void motorControl.setLeftSpeed( int8 speed ) {
    nxt_motor_set_speed(MOTOR_PORT_T::NXT_PORT_C, speed, 1);
  }

  procedure void motorControl.setRightSpeed( int8 speed ) {
    nxt_motor_set_speed(MOTOR_PORT_T::NXT_PORT_B, speed, 1);
  }
}
```

Requirements Traces

DummyRequirementsCollection | LineFollower

```
trace Cyclic
doc This is the cyclic task that is called every 1ms to do the actual control of the
task run cyclic prio = 1 every = 2 {
  trace TwoPhases
  stateswitch linefollower
    state running
      int8 bump = 0;
      bump = ecrobot_get_touch_sensor(SENSOR_PORT_T::NXT_PORT_S3);
      if ( bump == 1 ) {
        event linefollower:bumped
        terminate;
      }
      trace Init
      int32 light = 0;
      light = ecrobot_get_light_sensor(SENSOR_PORT_T::NXT_PORT_S1);
      if ( light < ( WHITE + BLACK ) / 2 ) {
        trace ConsistentSetting        ;
        updateMotorSettings(SLOW, FAST)
      } else {
        trace ConsistentSetting        ;
        updateMotorSettings(FAST, SLOW)
      }
    state crash
      updateMotorSettings(0, 0);
    default
      <noop>;
}
```

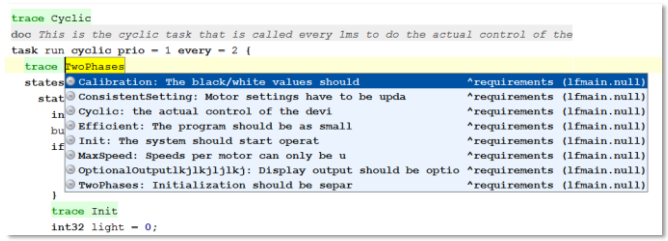
Imported Requirements

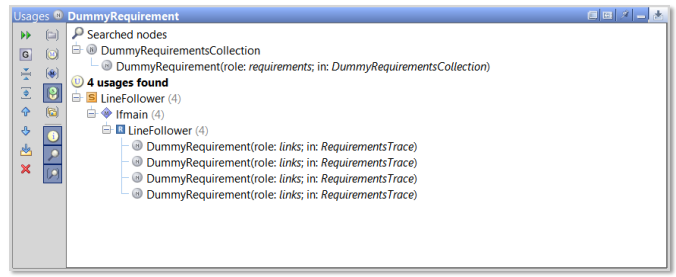
DummyRequirementsCollection

```
DUMMY REQUIREMENTS (to be replaced by interface to real RE tool)
show trace true
```

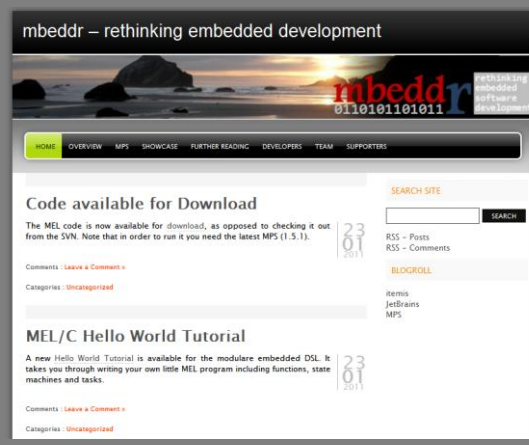
Init	The system should start operating only after it has been initialized property refines TwoPhases
Efficient	The program should be as small regarding memory footprint as possible refines Init refines MaxSpeed
Cyclic	the actual control of the device should be based on a cyclic task
Calibration	The black/white values should be easily calibrated
MaxSpeed	Speeds per motor can only be up to 80
OptionalOutputlkjlkjljlkj	Display output should be optional
TwoPhases	Initialization should be separate from operation
ConsistentSetting	Motor settings have to be updated consistently

Create Link, Go To, Fins Regs





Language Workbenches for Embedded Systems



mbeddr.com

Federal Ministry of Education and Research

KMU-INNOVATIV: IKT

itemis

fortiss
innovation in software and systems

SICK
Sensor Intelligence.

LEAR CORPORATION

THE END.
.coordinates
web www.voelter.de
email voelter@acm.org
twitter markusvoelter
xing http://www.xing.com/profile/Markus_Voelter
linkedin http://www.linkedin.com/pub/0/377/a31