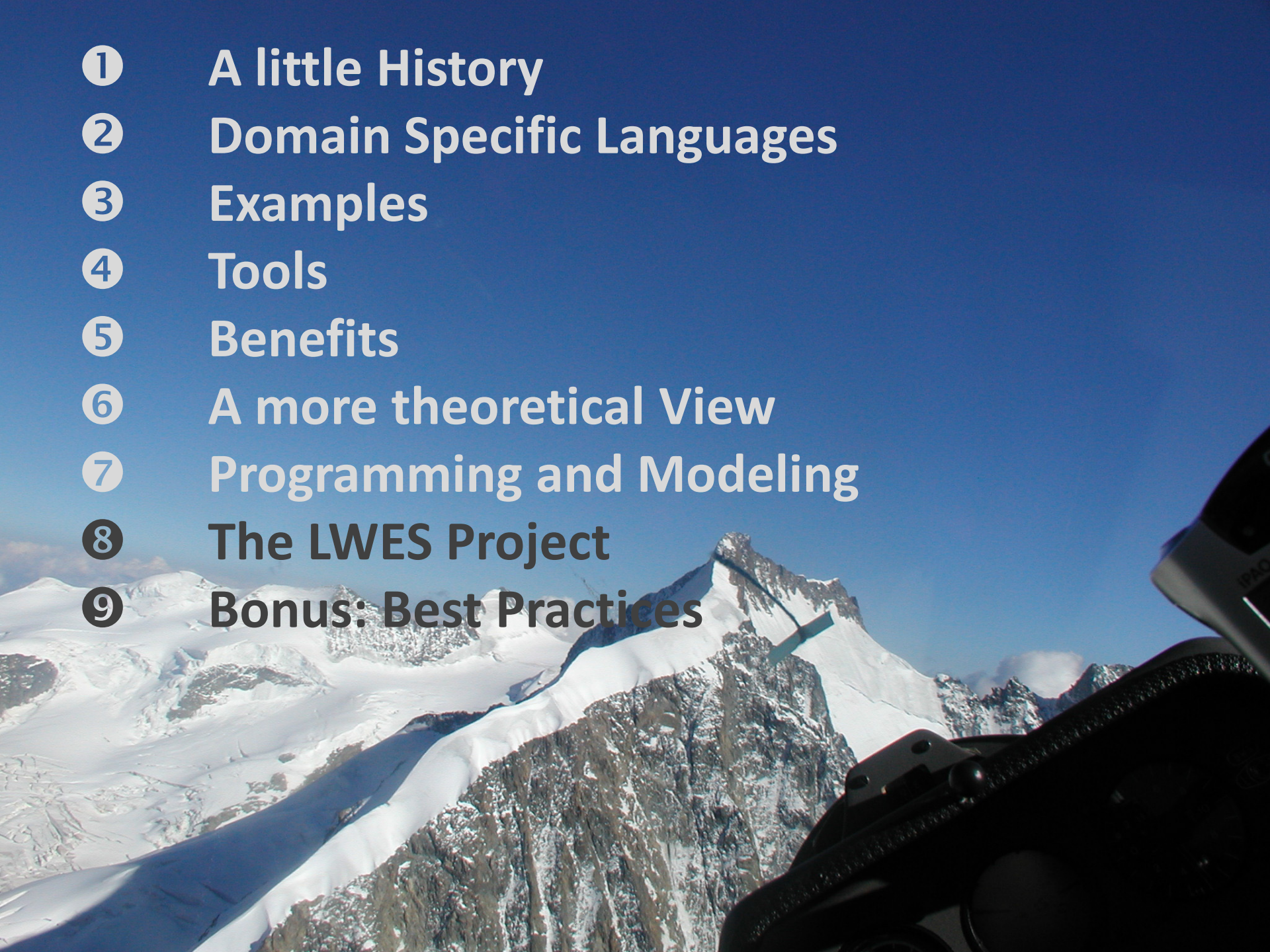


# Domain Specific Languages

A photograph of a snowy mountain range under a clear blue sky. The mountains are covered in snow and have sharp peaks. In the bottom right corner, a camera lens is visible, suggesting the photo was taken from a camera. The overall scene is bright and clear.

**Markus Voelter**  
Independent/itemis  
[voelter@acm.org](mailto:voelter@acm.org)

- 
- ① A little History
  - ② Domain Specific Languages
  - ③ Examples
  - ④ Tools
  - ⑤ Benefits
  - ⑥ A more theoretical View
  - ⑦ Programming and Modeling
  - ⑧ The LWES Project
  - ⑨ Bonus: Best Practices

1

# A little History





**programming**

started

**close to the hardware**

**abstractions**

~

**computing**

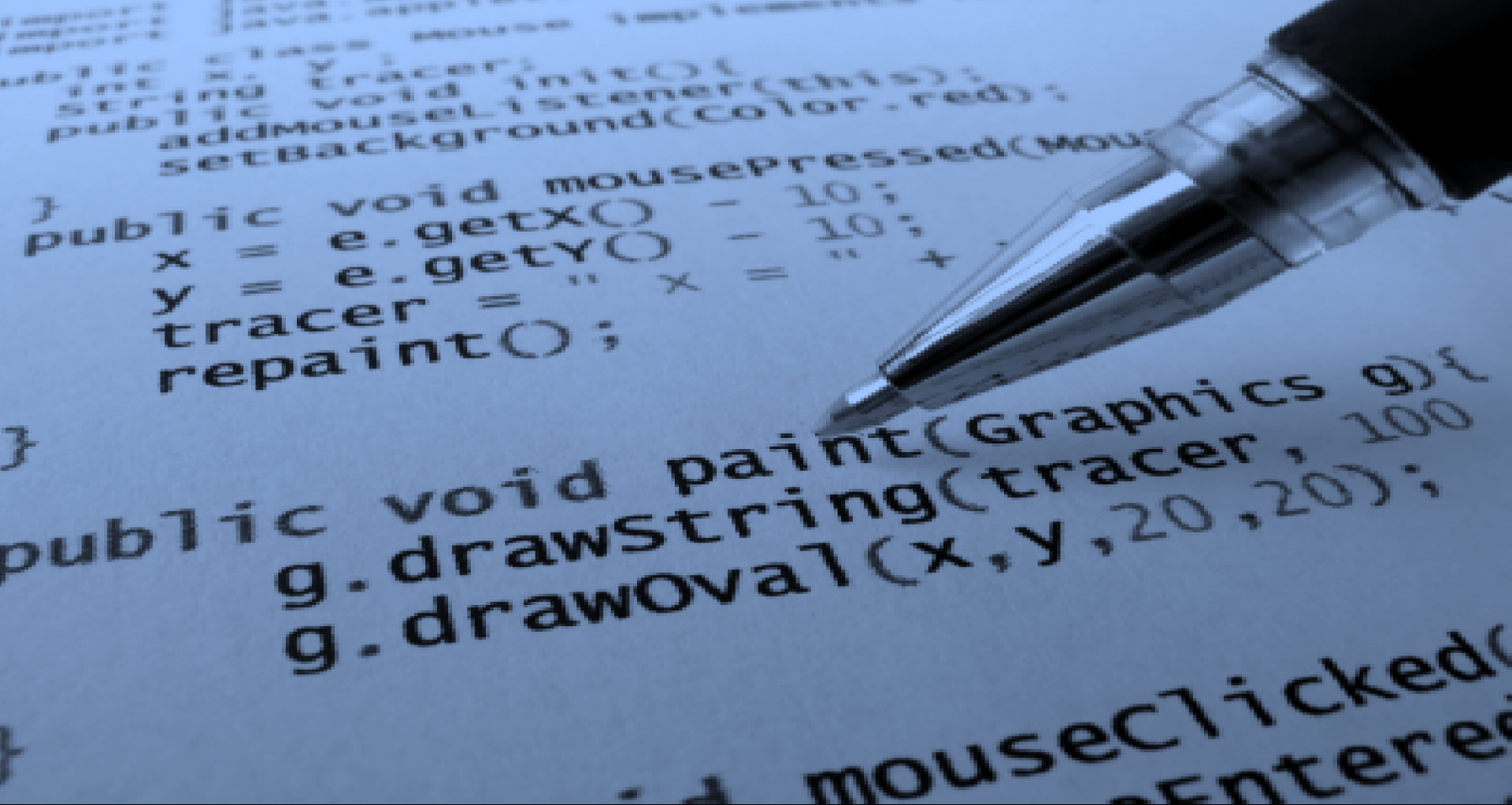
**chips**



**abstractions**  
~  
**computing**

**bits**



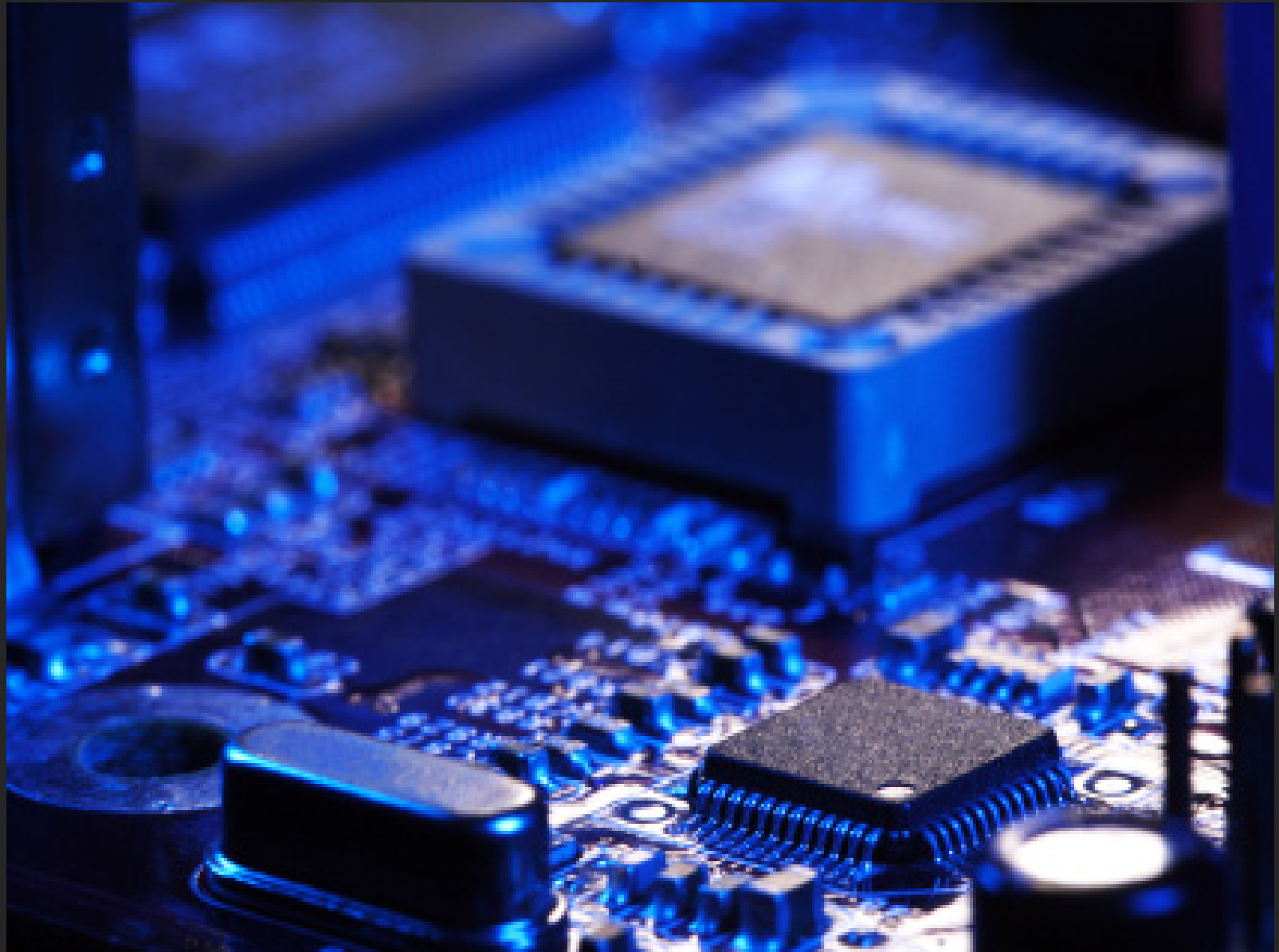


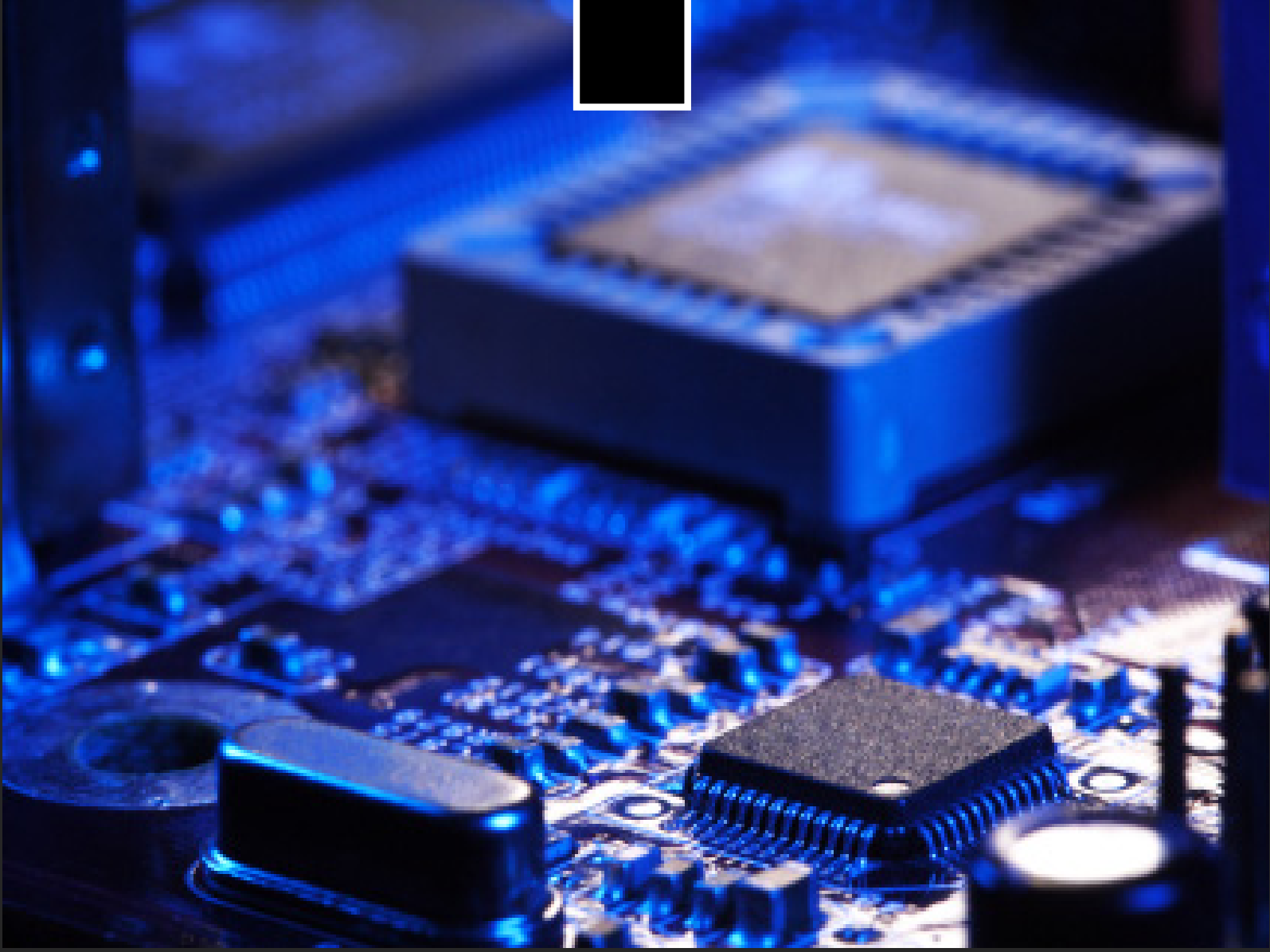
abstractions  
~  
computing?

```
ConnectionString="Database=DB_home; Username=dbuser; Password=  
DBProvider = " Database provider" DB.connect ConnectionStr  
SelectSQL1 = " Select id, name, quantity from all w  
QuerySQL1 = " where id between decode(name, 'Scott'  
QuerySQL2 = " group by id, name"  
SelectQuery = SelectSQL1 & QuerySQL1 & QuerySQL2  
Execute Query; Commit Transaction; Select new data  
Form Navigation  
If KeyAscii = 13 Then Execute Query  
If Not Chr(KeyAscii) Like "#" And KeyAscii <> 8 Then
```

**abstractions**  
~  
**computing?**



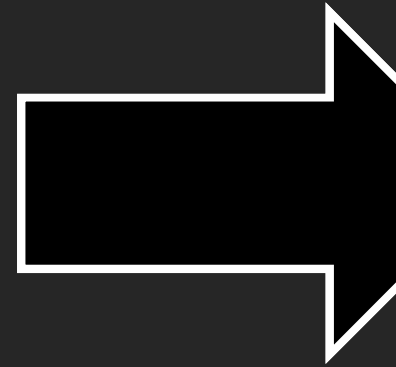




?







**general purpose**



**domain specific**

**tailor made  
effective++**

**specialized, limited**

**used by experts**

**together with other  
specialized tools**



2

# 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.

A DSL is a **focussed, processable**

**language** for describing a specific

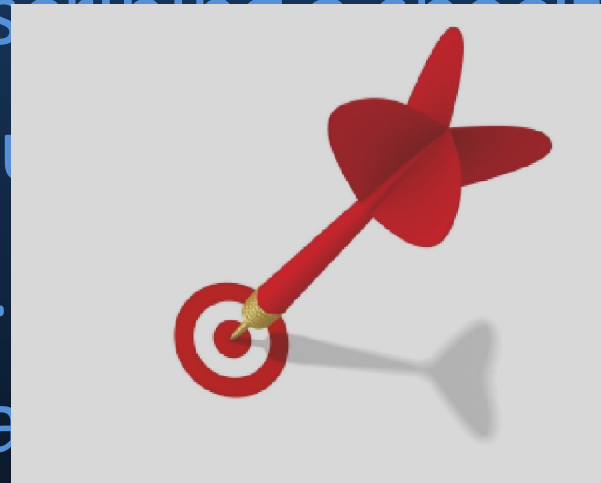
**concern** when built in a

specific **domain**.

**notations** used are available for

the **stakeholders** who specify that

particular concern.



# DSL

A DSL is a **focussed, processable** language for describing a specific concern in a system in a specific notation. DSLs are actions and notations suitable for the state of the system. DSLs verify that particular concern.



# DSL

A DSL is a **focussed, processable language** for describing a specific

**concern** within a specific domain. It uses **notations** understood by the **stakeholders** of a particular concern.



a system in a **restricted** set of **notations** and **constraints** that are **natural/suitable** for the domain. It **specifies** that

# DSL

A DSL is a **focussed, processable language** for describing a specific **concern** when building a system in a specific domain. **notations** use the **stakeholder** particular con

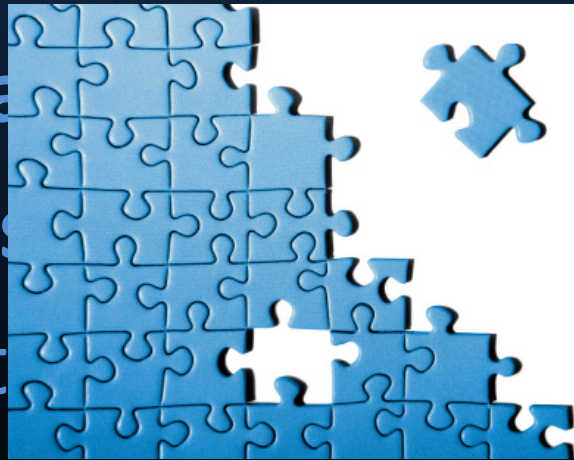


A DSL is a **focussed, processable language** for describing a specific **concern** when building a system in a specific **domain**. The **abstractions** and **notations** used are **tailored** for the **stakeholders** of a particular concern.



A DSL is a **focussed, processable language** for describing a specific **concern** when building a system in a specific **domain**. The **abstractions** and

**notation** is natural/suitable for the **users** who specify that part



A DSL is a focussed, processable language for describing a specific concern when

specific domain

notations used

the stakeholders

particular concern

The image shows several handwritten mathematical formulas on a blackboard background. The formulas are related to probability distributions and calculus. The first formula is  $\frac{\partial}{\partial \theta} \mathbb{M}T(\xi) = \frac{\partial}{\partial \theta} \int_{\mathbb{R}_n} T(x) f(x, \theta) dx = \int_{\mathbb{R}_n} \frac{\partial}{\partial \theta} T(x) f(x, \theta) dx$ . The second formula is  $\frac{\partial}{\partial a} \ln f_{a, \sigma^2}(\xi_1) = \frac{(\xi_1 - a)}{\sigma^2} f_{a, \sigma^2}(\xi_1) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left\{-\frac{(\xi_1 - a)^2}{2\sigma^2}\right\}$ . The third formula is  $\int_{\mathbb{R}_n} T(x) \cdot \frac{\partial}{\partial \theta} f(x, \theta) dx = \mathbb{M}\left(T(\xi) \cdot \frac{\partial}{\partial \theta} \ln L(\xi, \theta)\right)$ . The fourth formula is  $\int_{\mathbb{R}_n} T(x) \cdot \left(\frac{\partial}{\partial \theta} \ln L(x, \theta)\right) \cdot f(x, \theta) dx = \int_{\mathbb{R}_n} T(x) \cdot \left(\frac{\frac{\partial}{\partial \theta} f(x, \theta)}{f(x, \theta)}\right) \cdot f(x, \theta) dx$ . The fifth formula is  $\frac{\partial}{\partial \theta} \mathbb{M}T(\xi) = \frac{\partial}{\partial \theta} \int_{\mathbb{R}_n} T(x) f(x, \theta) dx = \int_{\mathbb{R}_n} \frac{\partial}{\partial \theta} T(x) f(x, \theta) dx = \int_{\mathbb{R}_n} T(x) \cdot \left(\frac{\partial}{\partial \theta} \ln L(x, \theta)\right) \cdot f(x, \theta) dx = \int_{\mathbb{R}_n} T(x) \cdot \left(\frac{\frac{\partial}{\partial \theta} f(x, \theta)}{f(x, \theta)}\right) \cdot f(x, \theta) dx = \int_{\mathbb{R}_n} T(x) \cdot \left(\frac{\partial}{\partial \theta} \ln L(x, \theta)\right) \cdot f(x, \theta) dx$ .

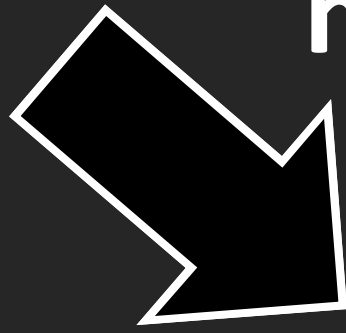


A DSL is a **focussed, processable language** for describing a specific **concern** when building a system in a specific **domain**. The **notations** used are named for the **stakeholders** who have a particular concern.

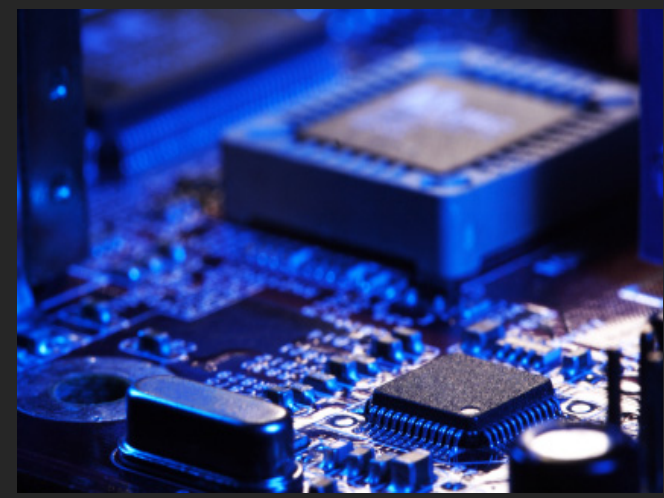




execute?



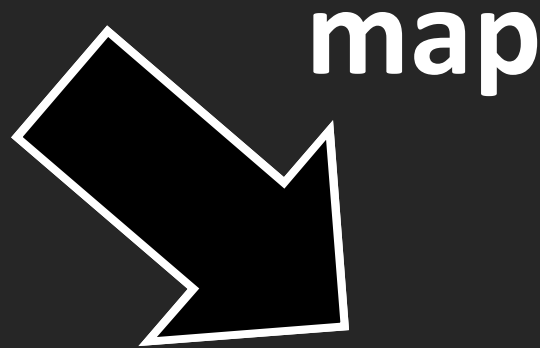
map



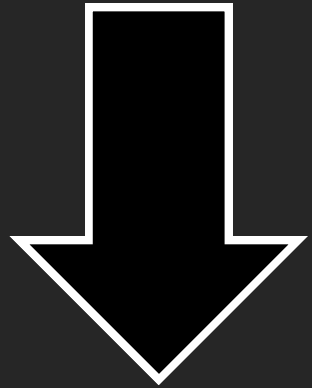
# DSL Program

(aka Model)

**automated!**



# GPL Program



map

**Generation**

Transformation

Compilation

**Interpretation**

Language Oriented Programming

Domain Specific Modeling

Generative Programming

Model Driven Development

Model Integrated Computing

Model Based Development

Model Driven Engineering

Model Driven Architecture

Model Based Engineering

Model Driven Software Development

# Activities

**Analysing Domains**

**Defining Languages**

*Adapting/Selecting*

**Building Editors**

**Transforming Models**

**Building Generators**

**Building Frameworks**

# Activities

**Analysing Domains**

**Defining Languages**

*Adapting/Selecting*

**Building Editors**

**Transforming Models**

**Building Generators**

**Building Frameworks**

**... and using all of that to build apps**



**internal**

**vs.**

**external**

compiled

vs.

interpreted

**customization**

**VS.**

**configuration**

**graphical**

**vs.**

**textual**

3

# Examples



**Example 1:**

# **Embedded Protocol Handler**



# Component Specification

```
processing DigitalIn "BI" moduletype Ox08 hal = DigitalInHAL {  
  
    datatypes {  
        SinglePointIndicationWithoutTime;  
        SinglePointIndicationWithTime;  
        DoublePointIndicationWithoutTime;  
        DoublePointIndicationWithTime;  
        BitStringTypeI8BitWithoutTime;  
        BitStringTypeI8BitWithTime;  
    }  
  
    parametertypes {  
        DataType default {  
            subattr db0 # intendedDataType == pdt SinglePointIndicationWithTime;  
        };  
        DebounceFilterTime default {  
            attr filterTimeInMs == Ox02;  
            subattr db1 # SP == Ox00;  
            subattr db1 # IN == Ox00;  
        } ;  
        MaximumOscillatingFrequency;  
    }  
  
    function READDATA () : ProcessData;  
    function WRITEDATA(input : ProcessData);  
  
    struct ProcessData {  
        int8 channel;  
        int8 fixData[4];  
    }  
  
    struct Memory {  
        int8 state;  
        ProcessData data;  
    }  
  
    instance memory Memory ;  
}
```

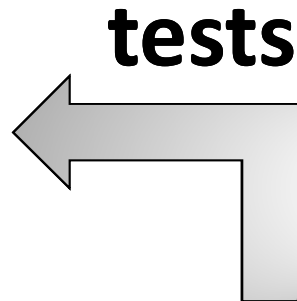
# Message Format Definition

```
procedure writeRegisterNumberZ requestCode 0x29 {
  request: struct request1 {
    int8 acc pattern {
      2:b00;
      6:parentRequestCode;
    };
    int8 registerAddress;
  } ;
  reply: struct dontCareReply {
    int8 statusByte patternref statusByte;
    int8 dontCare patternref defaultReturn;
  } ;
  request: struct request2 {
    int8 registerType pattern {
      4:b0000;
      4:registerType;
    };
    int8 registerAddress;
    int8 registerdata [2];
  } ;
}
```

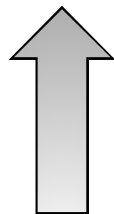


# Testing

```
procedure writeRegisterNumberZ requestCode 0x29 {
  request: struct request1 {
    int8 acc pattern {
      2:b00;
      6:parentRequestCode;
    };
    int8 registerAddress;
  };
  reply: struct dontCareReply {
    int8 statusByte patternref statusByte;
    int8 dontCare patternref defaultReturn;
  };
  request: struct request2 {
    int8 registerType pattern {
      4:b0000;
      4:registerType;
    };
    int8 registerAddress;
    int8 registerdata [2];
  };
}
```

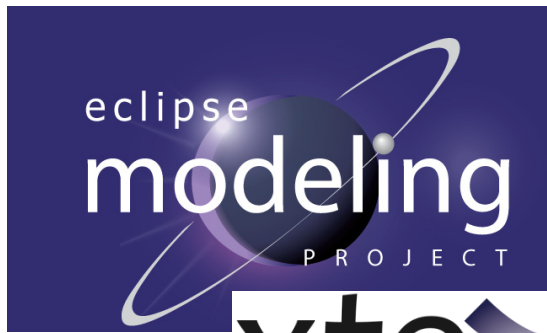


```
test writeRegisterNumberZ for dip writeRegisterNumberZ {
  send request1 {
    attr registerAddress == reg parameterInstruction;
  };
  expect dontCareReply {
    subattr statusByte # standardStatus == 2;
  };
  send request2 {
    subattr registerType # registerType == 3;
    attr registerAddress == reg parameterInstruction;
    attr registerdata == 0x77;
    subattr registerdata # channelNumber == 5;
  };
}
```



refines

```
register parameterInstruction address 0x37 struct {
  int8 db1 pattern {
    2:b00;
    6:channelNumber;
  };
};
```



# Eclipse Modeling Eclipse Xtext

Example 2:

# Pension Fund Specification



# Textual Documentation

The screenshot displays the Capgemini Pension Workbench application. The window title is "Capgemini Pension Workbench" and the menu bar includes "File", "Edit", "Projection", "Navigation", "Search", "Format", "Tools", "Dev", "Generate", "Pension", "Team", and "NN". The main workspace shows a "Table of Contents" on the left and a detailed view of the "Library NN LC PA" on the right. The "Table of Contents" lists various value sets and tag definitions. The main view shows the "Documentation" section, which includes an introduction and a section on "Opbouwtoezeggingen" (Accumulation Contributions).

**Table of Contents**

- Library
  - Documentation
  - Foundation
    - Value sets
      - Value set Groottebepalingsmethode
        - Value set member Salaris-diensttijd
        - Value set member Verzekerde bedragen
        - Value set member Afgeleide toezegging
      - Value set Salaris-diensttijd
        - Value set member Middelloon
        - Value set member Eindloon
      - Value set Verzekerde bedragen
        - Value set member Vast bedrag
        - Value set member Percentage van gron
        - Value set member Percentage van gron
        - Value set member Opgegeven bedrag
        - Value set member ANW-hiaat
        - Value set member AOP bedrag
      - Value set Indicatie Opbouw / Risico
        - Value set member Opbouw
        - Value set member Risico
      - Value set Deelnemerstatus
        - Value set member Aspirant
        - Value set member Actief
        - Value set member Premievrij
        - Value set member Slapend
        - Value set member Uitkerend
        - Value set member Overleden
        - Value set member Vervallen
      - Tag definitions
        - Tag Basisberekening
        - Tag Ouderdomspensioen
        - Tag Partnerpensioen
        - Tag Wezenpensioen
        - Tag ANW extra
        - Tag WIA excedent AOV

# Insurance Mathematics

SOLUTION

Capgemini Pension Workbench

File Edit Projection Navigation Search Format Tools Dev Generate Pension Team NN

NNLCPA-14w2-21112008 \* x

Table of Contents \* All

Library

- [-] Documentation
- [-] Foundation
  - [-] Value sets
    - [-] Value set Groottebepalingmethode
      - Value set member Salaris-diensttijd
      - Value set member Verzekerde bedragen
      - Value set member Afgeleide toezegging
    - [-] Value set Salaris-diensttijd
      - Value set member Middelloon
      - Value set member Eindloon
    - [-] Value set Verzekerde bedragen
      - Value set member Vast bedrag
      - Value set member Percentage van gron
      - Value set member Percentage van gron
      - Value set member Opgegeven bedrag
      - Value set member ANW-hiaat
      - Value set member AOP bedrag
    - [-] Value set Indicatie Opbouw / Risico
      - Value set member Opbouw
      - Value set member Risico
    - [-] Value set Deelnemerstatus
      - Value set member Aspirant
      - Value set member Actief
      - Value set member Premievrij
      - Value set member Slapend
      - Value set member Uitkerend
      - Value set member Overleden
      - Value set member Vervallen
    - [-] Tag definitions
      - Tag Basisberekening
      - Tag Ouderdomspensioen
      - Tag Partnerpensioen
      - Tag Wezenpensioen
      - Tag ANW extra
      - Tag WIA excedent AOV

[-] **3.3 Commutatietellingen op 1 leven**

$$D_x = v^x * \frac{l_x}{100} \quad \text{26 Dec (3)}$$

Implemented in [V9401](#)

$$\omega - x$$

$$N_x = \sum_{t=0}^{\omega - x} D_{x+t} \quad \text{27 Dec (3)}$$

[-] **3.6 Contante waarde 1 leven/ 2 levens**

$$E_x = \frac{D_x}{D_x^{x+n}} \quad \text{19 Dec (4)}$$

$$a_x = \ddot{a}_x - 1 \quad \text{21 Dec (3)}$$

$$\bar{a}_x = \ddot{a}_x - 0,5 \quad \text{22 Dec (3)}$$

$$\ddot{a}_{x:n} = \frac{N_x - N_{x+n}}{D_x} \quad \text{23 Dec (3)}$$

$$\bar{a}_{x:n} = \ddot{a}_{x:n} - 0,5 + 0,5 * E_{n|x} \quad \text{25 Dec (3)}$$

[-] **4 BN(\_ris) koopsommen**

Section | title | Paragraph | Text | Dev  
Doc | Splitter | Pension | PensionDecorated | AM

# Calculation Rules and Tests

The screenshot displays the Capgemini Pension Workbench interface. The main window shows the 'Table of Contents' on the left and the 'All' view of the 'Rule Bereken Mutatieperiode' in the center. The rule's documentation and algorithm are visible, along with a table of test cases.

**Table of Contents (Left Panel):**

- Library
  - Documentation
  - Foundation
    - Value sets...
    - Tag definitions
      - Tag Basisberekening
      - Tag Ouderdomspensioen
      - Tag Partnerpensioen
      - Tag Wezenpensioen
      - Tag ANW extra
      - Tag WIA excedent AOV
      - Tag Eindkapitaal
  - Shared
    - Elements...
    - Rules
      - Rule Bereken Mutatieperiode
      - Rule Bereken Salaris ontwikkeling
      - Rule Bereken Pensioengrondslag
      - Rule Bereken pensioengrondslag
      - Rule Bereken Bedrag jaaropbouw
      - Rule Bereken Bedrag jaaropbouw
      - Rule Bereken Delta deelaanspraak uit doort
      - Rule Bereken Deelaanspraak opgebouwd
      - Rule Bereken Toekomstige dienstjaren
      - Rule Bereken Deelaanspraak uitzicht
      - Rule Bereken Verzekerd bedrag
      - Rule Bereken Verzekerd bedrag
      - Rule Bereken Verzekerd bedrag
      - Rule Bereken Verkoopkosten
      - Rule Bereken Netto Eindwaarde
      - Rule Bereken einddatum opbouw
      - Rule Bereken premie
      - Rule Bereken IS-Opslag
      - Rule Bereken administratiekosten

## Elements...

## Rules

### Rule Bereken Mutatieperiode

#### Result:

Mutatieperiode

#### Name:

Bereken Mutatieperiode

#### Documentation:

Het vaststellen van de periode tussen de huidige en de vorige mutatie in dagen.¶

De mutatieperiode kan niet meer dan 360 dagen bedragen omdat elk jaar een begin- en eindmutatie kent i.v.m. het openen en sluiten van het verslagjaar.¶

Dit wordt niet afgevangen omdat het uitvoeren van de begin- en eindmutatie verantwoordelijkheid zijn van de pensioenadministratie.¶

#### Tags:

Basisberekening

#### Algorithm:

**if maximum(Mutaties per datum) == 1 then daysof(duration(valid(Mutaties per datum))) else 0**

#### Test cases:

Name	Valid time	Transaction time	Fixture	Product	Element	Expected value	Actual value
Gelijke datums	03/01/2008		Mutatieperiode - Mutatiedatum = Mutatiedatum Vorig			3	0
Periode < 30	03/01/2008		Mutatieperiode - Mutatiedatum > Mutatiedatum Vorig (binnen 1 maand)			15	15
Periode > 30	03/01/2008		Mutatieperiode - Mutatiedatum > Mutatiedatum Vorig ( meerdere maanden)			60	60

Bereken Mutatieperiode > Test cases > Unit test: Gelijke datums : ^Place Dev

Doc | Splitter | Pension | PensionDecorated | AM



# Intentional Software's Domain Workbench

Example 3:

# Radars Systems Engineering

---





# Component Definition

```
test.cdsl x test.mm
import "classpath:/test.mm"

quantity voltage is double
quantity temperature is double

source component Sensor {
  produces m: {measurement: voltage, sensortemp: temperature}
  behavior {
    m.measurement <= sensorM[]
    m.sensortemp <= sensorT[]
  }
}

processing component TempCalibration {
  consumes input: Sensor::m
  produces calibrated: { measurement: voltage }
  behavior {
    calibrated.measurme
  }
}

processing component Process
  consumes input: { measur
  produces earthtemp: { te
  behavior {
    earthtemp.temp <= pr
  }
}

sink component Output {
  consumes t : Processor::earthtemp
}

system satellite {
  s: Sensor
  tc: TempCalibration
  p: Processor
  o: Output

  s.m -> tc.input
  tc.calibrated -> p.input
  p.earthtemp -> o.t

  export o.t.temp as temperature
  export s.m.measurement as originalMeasurement
}
}
```

# Component Behavior Specification



```
ComponentDSL.xtext test.m ✕
BeginPackage["MappingSatellite`"]

Begin["`Private`"]

sensorM[t_Int] := 23*t
sensorT[t_Int] := 300

calibrate[ m_Double, temp_Double ] := m - temp/10

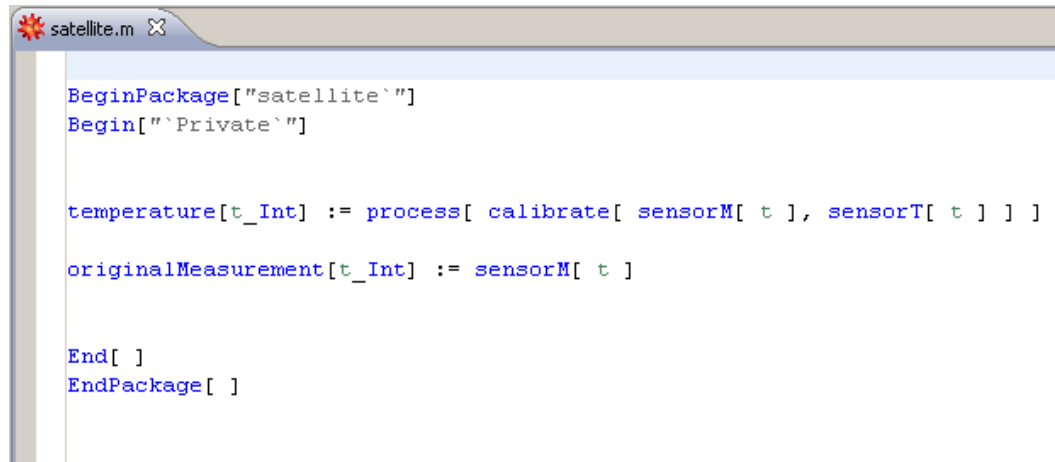
process[ m_Double ] := m*3

End[ ]

EndPackage[ ]

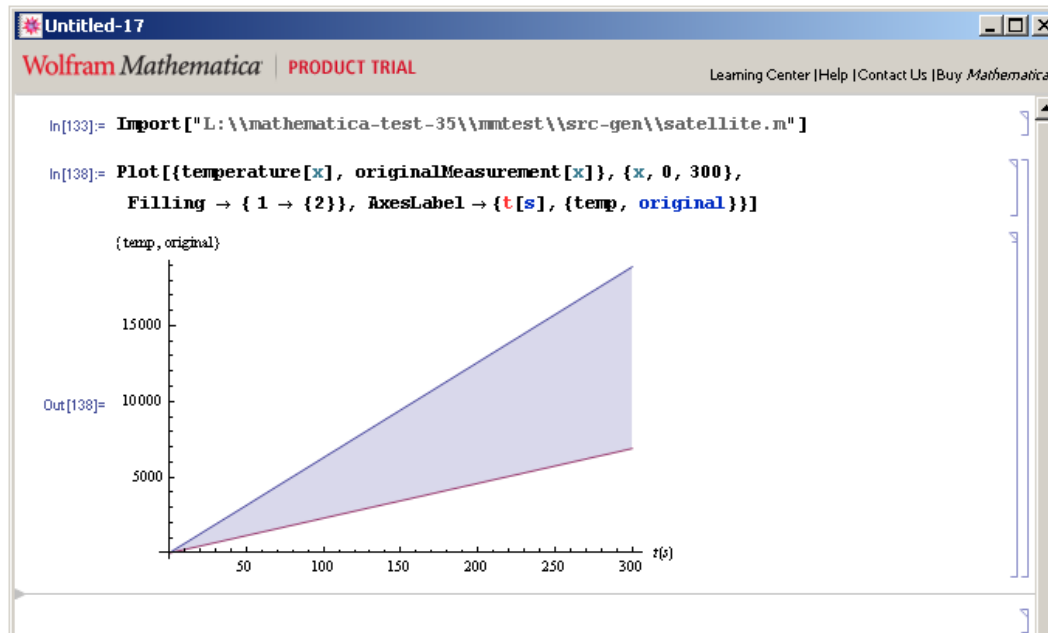
(* Mathematica Raw Program *)
```

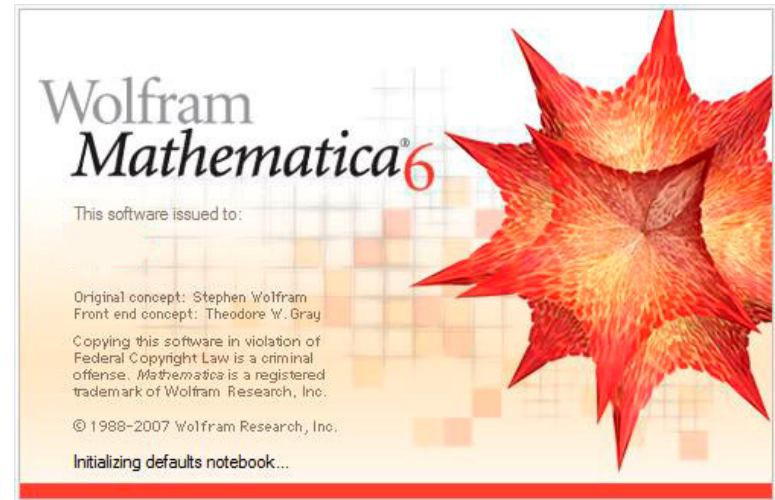
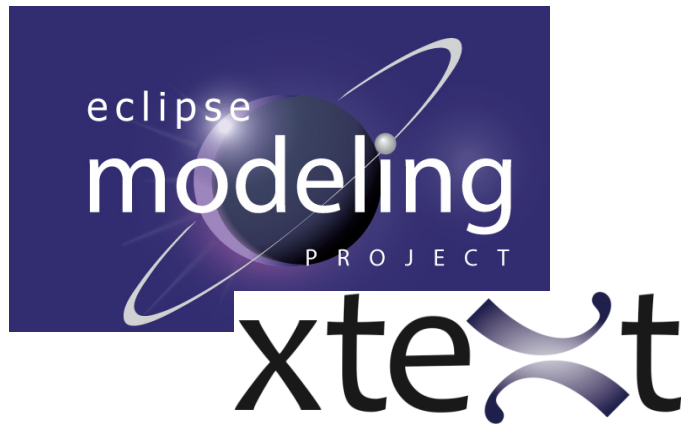
# Resulting System Behaviour



```
satellite.m ✕  
  
BeginPackage("satellite")  
Begin("`Private`")  
  
temperature[t_Int] := process[ calibrate[ sensorM[ t ], sensorT[ t ] ] ]  
  
originalMeasurement[t_Int] := sensorM[ t ]  
  
End[ ]  
EndPackage[ ]
```

# Analysis





**Eclipse Modeling**  
**Eclipse Xtext**  
**Wolfram Mathematica**  
**Mathematica Workbench**

Example 4:

# Alarm System Menus

---



# Menu Structure

```
import "classpath:/units.md"
import "classpath:/software.swc"

namespace s1

    uses units

    condition Locked
    condition BlinkingLight

    menu Normal label "Standardmenu"
        item unlockNow sys(TurnOffAlarm) if Locked
            button label "Unlock"
        submenu Manual label "Manual Settings"
            item alarmLevel sys(AlarmLevel)
                valuerange SoundLevel restrict 10..80
            item useLight sys(TurnOffAlarm) if BlinkingLight
                bool
        end
        submenu autoLocking label "Automatic Locking"
            item startTime sys(TurnOnAlarm)
                valuerange Time
            item endTime sys(TurnOffAlarm)
                valuerange Time
            template areaSettings [size=15, area=1, sw=sys(TurnOnAlarm)] area1Settings
            template areaSettings [size=10, area=2, sw=sys(TurnOnAlarm)] area2Settings
        end
    end

    template [size: int, area: int, sw: swref] areaSettings
        item onOrOff sys(TurnOffAlarm) labelexpr "Autolock "+area+" on/off"
            bool true = label(size) "On" false = label "Off"
        item test sys(AlarmLevel) label "Test"
            bool
        item alarmLevel sys(AlarmLevel)
            valuerange SoundLevel restrict size..80
    end

    menu Expert extends Normal
        item master sys(UnlockNow) afterItem unlockNow
            bool
    end

end
```

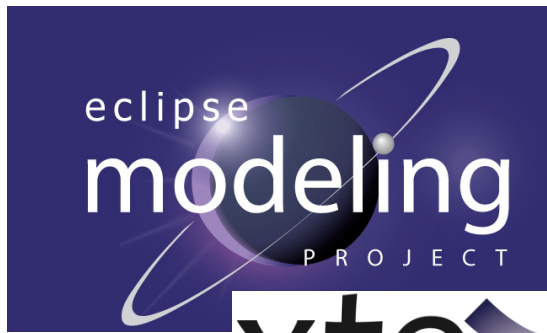
# Software Components

```
message TurnOffAlarm
message TurnOnAlarm
message AlarmLevel
message UnlockNow

component AlarmManager {
  receives TurnOffAlarm
  receives TurnOnAlarm
  receives AlarmLevel
}

component MasterSwitch {
  receives UnlockNow
}
```





xtext

# Eclipse Modeling

## Eclipse Xtext

Example 5:

# Requirements Tracability

---



# 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 <code>refines TwoPhases</code>
Efficient	The program should be as small regarding memory footprint as possible <code>refines Init refines MaxSpeed</code>
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
OptionalOutputlkljlkjljlkj	Display output should be optional
TwoPhases	Initialization should be separate from operation
ConsistentSetting	Motor settings have to be updated consistently

# Program Code with Annotations (green)

```
DummyRequirementsCollection x LineFollower x

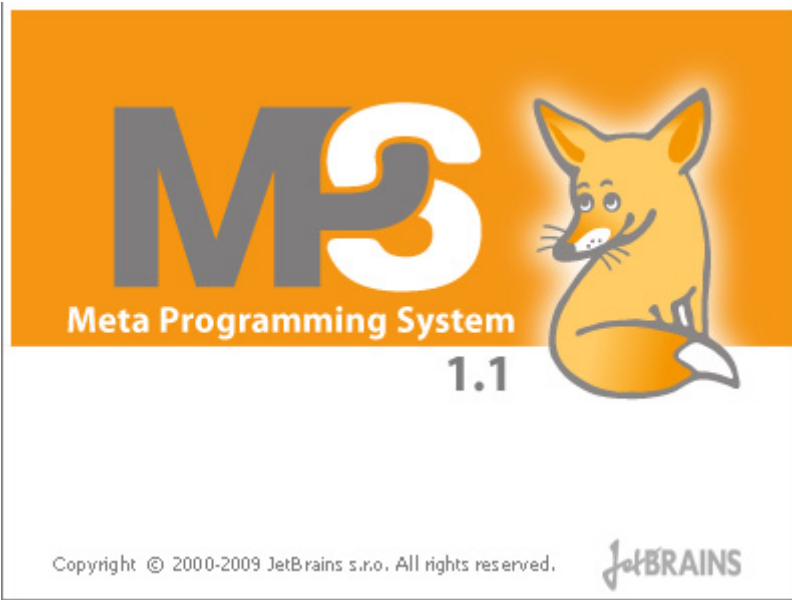
trace Cyclic
doc This is the cyclic task that is called every 1ms to do the actual control of the
task run cyclic prio = 1 every = 2 {
  trace TwoPhases
  stateswitch linefollower
  state running
    int8 bump = 0;
    bump = erobot_get_touch_sensor(SENSOR_PORT_T::NXT_PORT_S3);
    if ( bump == 1 ) {
      event linefollower:bumped
      terminate;
    }
  trace Init
  int32 light = 0;
  light = erobot_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>;
}
}
```

# Selecting from the Requirements

```
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
  states
    Calibration: The black/white values should ^requirements (lfmain.null)
  stat
    ConsistentSetting: Motor settings have to be upda ^requirements (lfmain.null)
  in
    Cyclic: the actual control of the devi ^requirements (lfmain.null)
  bu
    Efficient: The program should be as small ^requirements (lfmain.null)
  if
    Init: The system should start operat ^requirements (lfmain.null)
    MaxSpeed: Speeds per motor can only be u ^requirements (lfmain.null)
    OptionalOutputlkjlkjlkjlkj: Display output should be optio ^requirements (lfmain.null)
    TwoPhases: Initialization should be separ ^requirements (lfmain.null)
  }
  trace Init
  int32 light = 0;
```

## Find Usages of Requirements

The screenshot shows a window titled "Usages DummyRequirement". The window contains a tree view of the project structure. The tree shows a "Searched nodes" section with "DummyRequirementsCollection" and "DummyRequirement(role: requirements; in: DummyRequirementsCollection)". Below this, there is a section "4 usages found" with "LineFollower (4)" and "Ifmain (4)". Under "LineFollower (4)", there are four instances of "DummyRequirement(role: links; in: RequirementsTrace)".



# JetBrains MPS

# Example 5: OSGi-based System

---



# Component Specification

```
subsystem the.world.scenarios {

    public:

        immutable type ProblemReport {
            problem: string
            severity: int
            emergency: bool
        }

        interface Radio {
            report( a: ProblemReport ): void
        }

        interface Press {
            broadcast( a: string ): string
        }

    private:

        component Houston {
            provides uplink: Radio
            requires press: Press
        }

        component PrintingPress {
            provides source: Press
        }
}

subsystem the.moon.scenarios {

    uses the.world.scenarios

    private:

        component Armstrong {
            task sayHello scheduled onceUponStartup
            requires home: Radio [0..1]
        }
}
```



# Component Specification

```
subsystem the.world.scenarios {

    public:

        immutable type ProblemReport {
            problem: string
            severity: int
            emergency: bool
        }

        interface Radio {
            report( a: ProblemReport ): void
        }

        interface Press {
            broadcast( a: string ): string
        }

    private:

        component Houston {
            provides uplink: Radio
            requires press: Press
        }

        component PrintingPress {
            provides source: Press
        }
}

subsystem the.moon.scenarios {

    uses the.world.scenarios

    private:

        component Armstrong {
            task sayHello scheduled onceUponStartup
            requires home: Radio [0..1]
        }
}

import "classpath:/twoworlds.compsdl"

scenario Simple : the.moon.scenarios, the.world.scenarios
actor IRP ==> { Armstrong |

    var $m = ProblemReport {
        severity = 12
        emergency = true
        problem = "there's cheese"
    }

    home.report(msg#$m) -> { Houston |
        var $res = press.broadcast( < #msg.problem > ) -> {
            return "thanks"
        }
        log.info "they replied " 647x551
        log.info $res
        assert equals returnValueIsThanks : $res == "thanks"
        var $res2 = press.broadcast( < #msg.problem > ) -> {
            return "again"
        }
        log.info "and then they replied "
        log.info $res2
        assert equals returnValueIsAgain : $res2 == "again"
    }
}
}
```

# Component Specification

```
subsystem the.world.scenarios {

  public:

    immutable type ProblemReport {
      problem: string
      severity: int
      emergency: bool
    }

    interface Radio {
      report( a: ProblemReport ): void
    }

    interface Press {
      broadcast( a: string ): string
    }

  private:

    component Houston {
      provides uplink: Radio
      requires press: Press
    }

    component PrintingPress {
      provides source: Press
    }
}

subsystem the.moon.scenarios {

  uses the.world.scenarios

  private:

    component Armstrong {
      task sayHello scheduled onceUponStartup
      requires home: Radio [0..1]
    }
}
```

```
import "classpath:/twoworlds.compdsl"

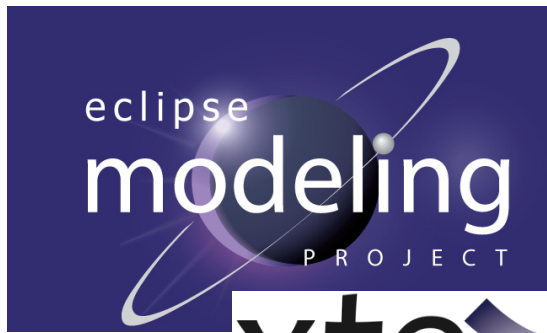
scenario Simple : the.moon.scenarios, the.world.scenarios
actor IRP ==> { Armstrong |

  var $m = ProblemReport {
    severity = 12
    emergency = true
    problem = "there's cheese"
  }

  home.report(msg#$m) -> { Houston |
    var $res = press.broadcast( < #msg.problem > ) -> {
      return "thanks"
    }
    log.info "they replied "
    log.info $res
    assert equals returnValueIsThanks : $res == "thanks"
    var $res2 = press.broadcast( < #msg.problem > ) -> {
      return "again"
    }
    log.info "and then they replied "
    log.info $res2
    assert equals returnValueIsAgain : $res2 == "again"
  }
}

import "classpath:/twoworlds.compdsl"
import "classpath:/twoworlds.scenario"

system SunSystem scenario Simple {
  node moon {
    subsystem the.moon.scenarios
  }
  node earth {
    subsystem the.world.scenarios
  }
}
```



# Eclipse Modeling

## Eclipse Xtext

**Example 6:**

# **Math, Science and Java**

---



```
env block Aircraft

v      : double [ $\frac{m}{s}$ ] / 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
```

```
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] =  $\frac{1}{2} * \rho * v^2$   $[\frac{kg}{m s s}]$ 
```

```
[ 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] =  $p_{dyn} * A * c_a$   $[\frac{m kg}{s s}]$ 
```

```
[ 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] =  $p_{dyn} * A * c_w$   $[\frac{kg m}{s s}]$ 
```

Angenommen wir haben mehrere Flügel  $n_{wings}$  am Flugzeug, dann berechnet sich der Auftrieb

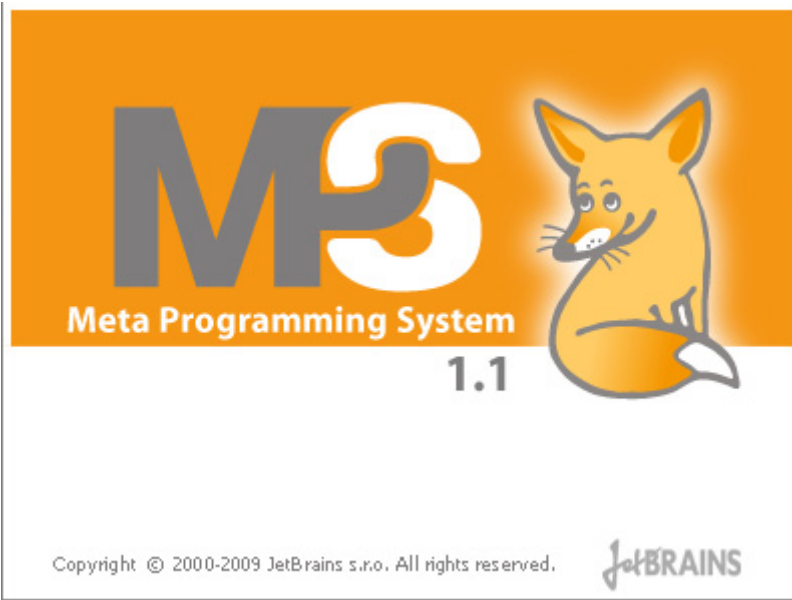
```
[import blocks Environment
    Aircraft
    Stuff on the Wings]
public class TestClass extends <none> implements <none> {
    <<static fields>>

    <<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>>
}
```



# JetBrains MPS

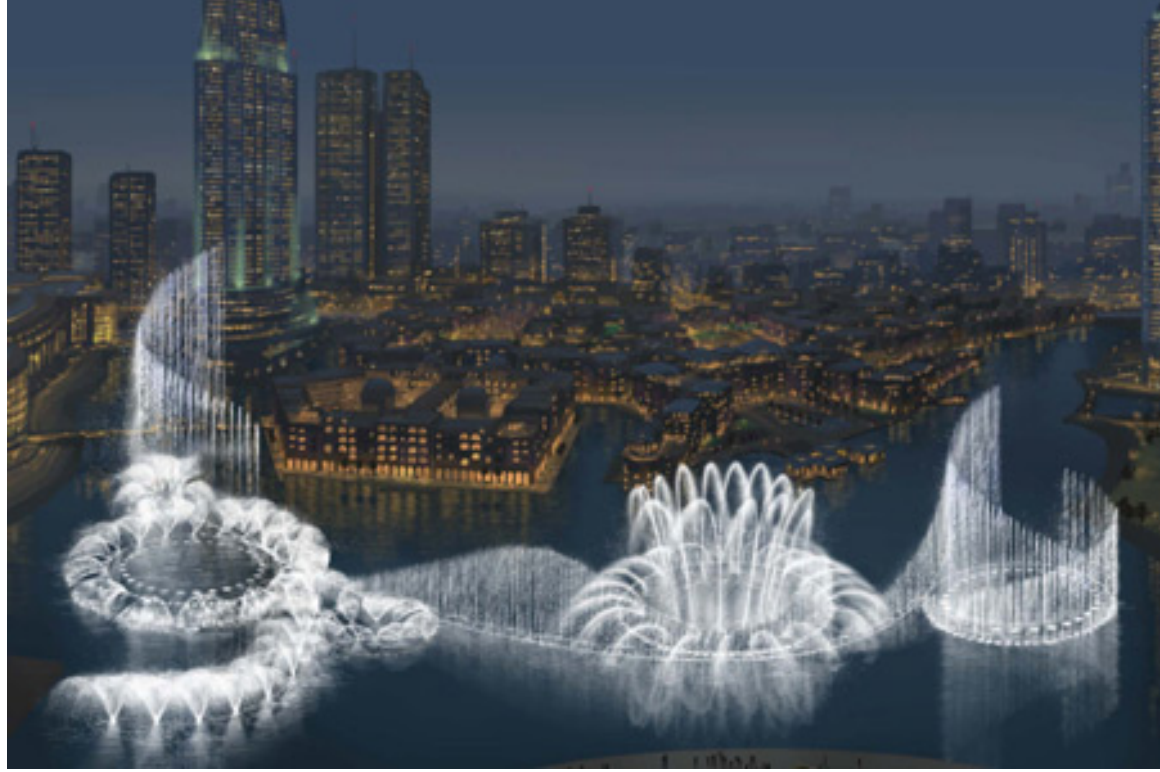


# Example 7: Fountains

---



# CONTEXT



# Hardware Structure

```
feature BasicOnePump
  pump compartment c1
  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
```

# Behaviour

```
pumping program P1 for AtLeastOneZone + WithAlarm +
    SuperPowerCompartment[f=comp1] {

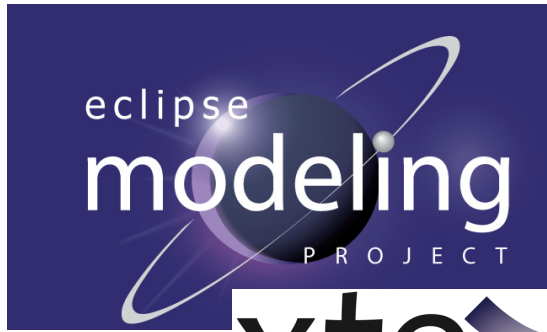
    parameter defaultWaterLevel : int
    parameter superWaterLevel: int
    event superPowerTimeout

    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
        }
    }
}
```

**Plus:**

**In-IDE Simulator  
Unit Test Support**



xtext

# Eclipse Modeling

## Eclipse Xtext

# More Examples8: Miscellaneous

---



# Hearing Aids

DOMAIN





# Refrigerators



DOMAIN

# BPEL Designer

Java - purchaseOrderProcess - Eclipse SDK

File Edit Navigate Search Project Run Window Help

Package Explorer Hierarchy

- org.eclipse.bpel.examples.orderprocessing [c...]
- OASISSampleProcess.bpel 1.1 (ASCII -kk...)
- OASISSampleProcess.bpelex 1.1 (ASCII -kk...)
- OASISSampleProcess.wsdl 1.1 (ASCII -kkv...)
- purchase.xsd 1.1 (ASCII -kkv...)
- org.eclipse.bpel.examples.simple [dev.eclipse...]

purchaseOrderProcess

- Selection Tool
- Marquee Tool
- Actions
- Empty
- Invoke
- Receive
- Reply
- Assign
- Control
- Switch
- Pick
- While
- Repeat Until
- Wait
- Faults
- Exit
- Throw
- Rethrow
- Compensate
- Zoom In
- Zoom Out

Outline

- Interface Partners
- Reference Partners
- Variables
- Correlation Sets
- Sequence
  - Receive
  - Flow
  - Reply

purchaseOrderProcess

- Interface Pa...
- purchasing
- invoicing
- shipping
- Reference P...
- scheduling
- Variables
- PO
- Invoice
- POFault
- shippingRequest
- shippingInfo
- shippingSchedule
- Correlation S...

Problems Javadoc Declaration Synchronize Properties

**Receive**

Description

Details

Partner Link: purchasing [Browse...](#)

Interface: purchaseOrderPT

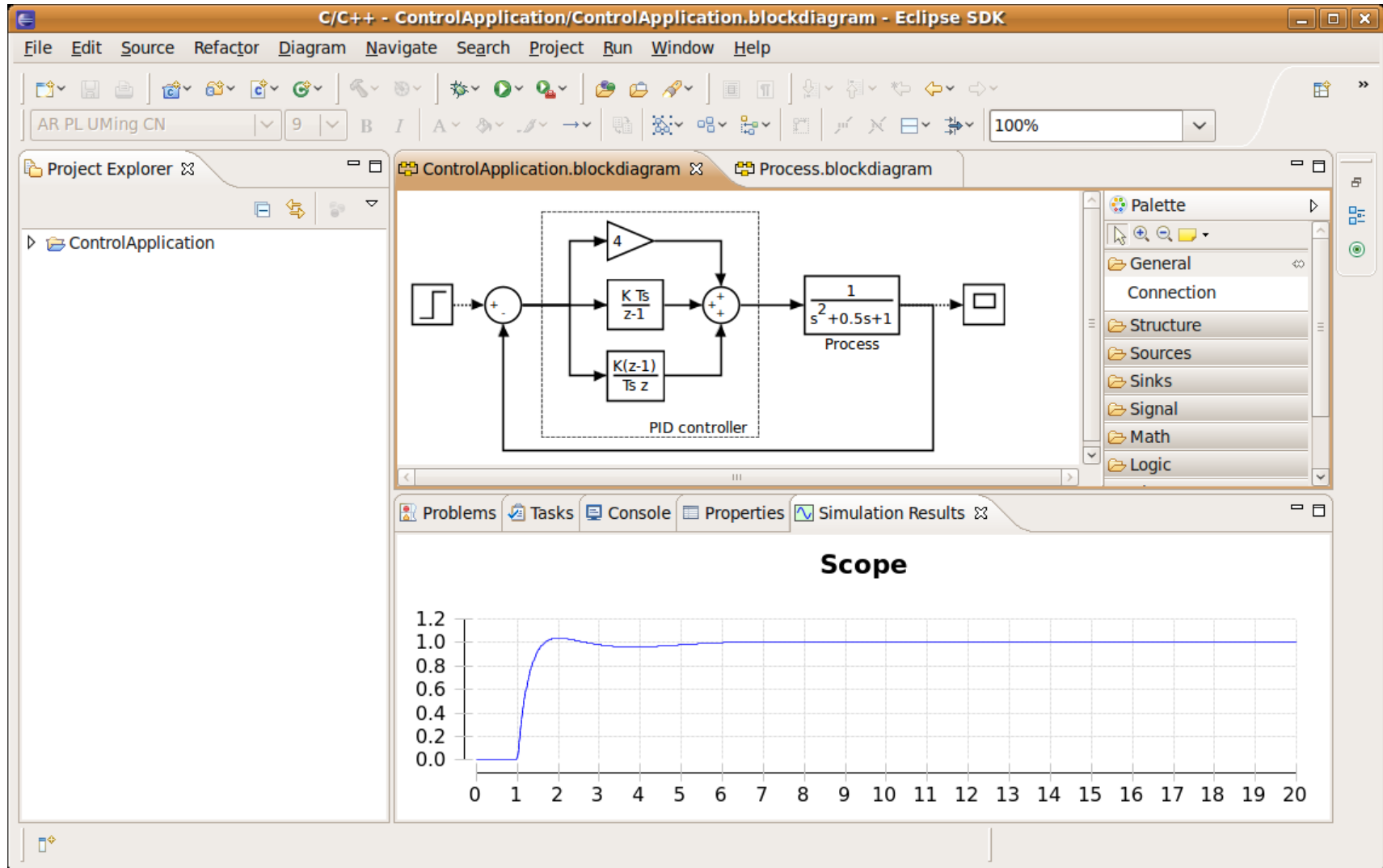
Operation: sendPurchaseOrder

Create a new Process instance if one does not already exist

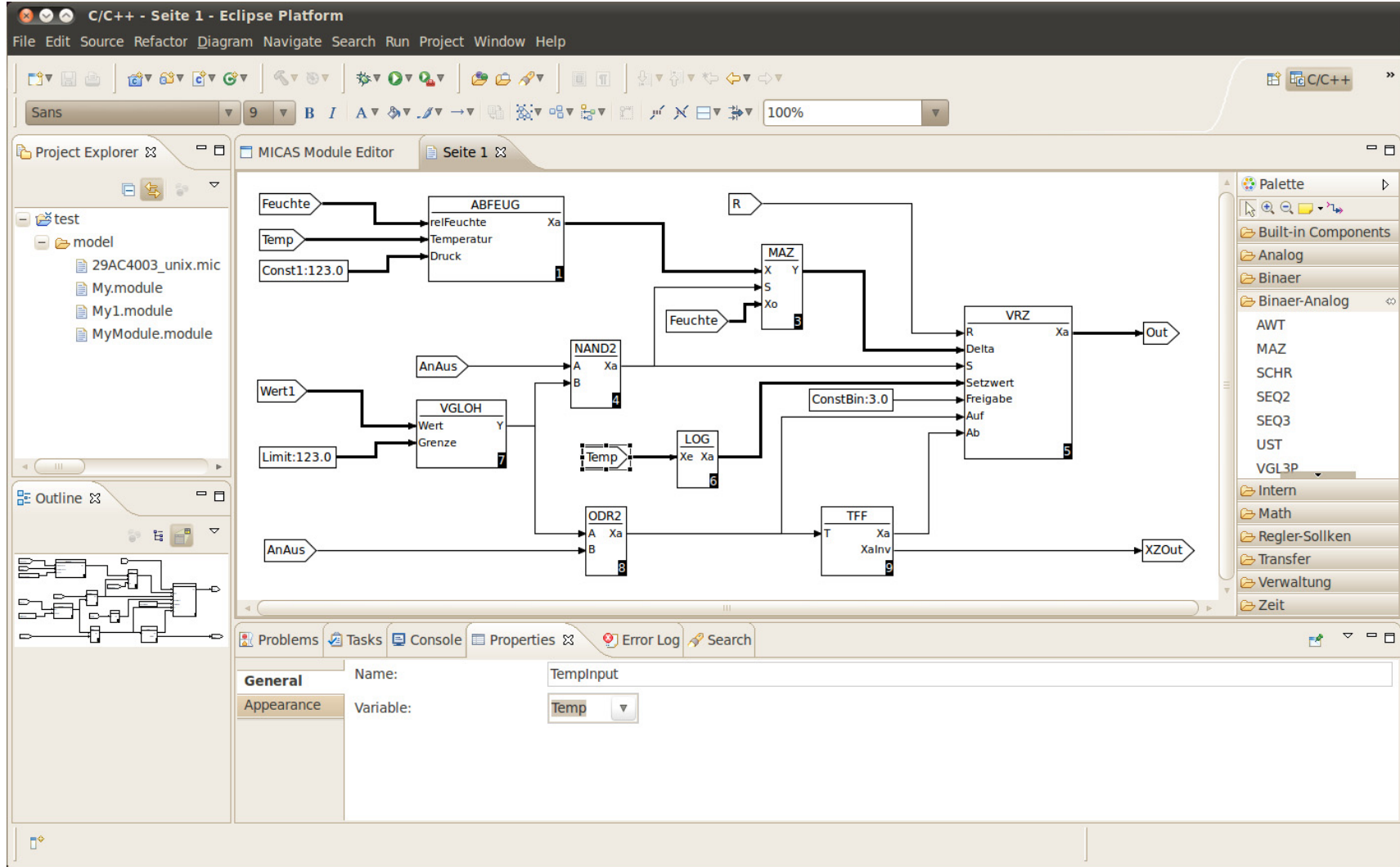
25M of 38M

# Block Diagrams

SOLUTION



# PLC Programming



SOLUTION

# State Charts

SOLUTION

The screenshot displays the Eclipse IDE interface for a Yakindu State Machine Diagram (SMD) titled "HeatingControl.blockdiagram". The main diagram shows a state machine with the following states and transitions:

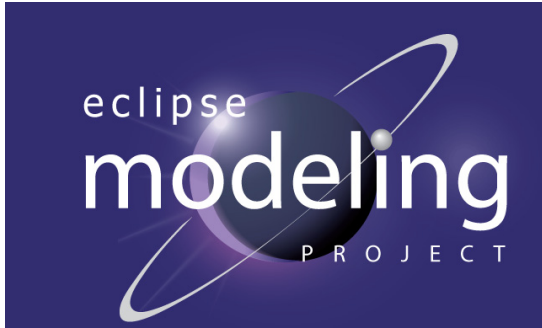
- Off** (yellow):
  - Initial state: `desiredTemp=30;`
  - Transition to **On**: `toggleOnOff` (event 0, action 1)
  - Transition from **On**: `toggleOnOff` (event 1, action 1)
- Alarm** (yellow):
  - Transition from **On**: `tempDrop [tempDropValue==1]` (event 0, action 2)
  - Transition to **On**: `tempDrop [tempDropValue==0]` (event 0, action 1)
- Setup** (yellow):
  - Transition from **Off**: `increaseTemp [desiredTemp<50] / desiredTemp+=1;` (event 0, action 1)
  - Transition to **On**: `setup` (event 2, action 3)
  - Transition from **On**: `setup` (event 1, action 2)
- On** (red):
  - Entry state: `<entry>`, `<do>`, `<exit>`
  - Composite state containing **Day** and **Night** states.
- Day** (yellow):
  - Initial state: `tempSetPoint=desiredTemp,status=1;`, `<do>`, `<exit>`
  - Transition to **Night**: `after(10s)` (event 0, action 1)
- Night** (red):
  - Initial state: `tempSetPoint=desiredTemp-1,status=2;`, `<do>`, `<exit>`
  - Transition to **Day**: `after(10s)` (event 0, action 1)

On the right, the "Yakindu State Machine View" shows the state variables and events:

Kategorie	Variable/Ereignis	
Eingang	actualTemp	
Eingang	dayTime	
Eingang	tempDropValue	
Ereignis	decreaseTemp	Auslösen
Ereignis	increaseTemp	Auslösen
Ereignis	powerOff	Auslösen
Ereignis	setup	Auslösen
Ereignis	tempDrop	Auslösen
Ereignis	toggleOnOff	Auslösen
Variable	desiredTemp	
Ausgang	status	
Ausgang	tempSetPoint	

The bottom of the IDE shows the "State Chart Interface" and "Simulation Results" tabs.

Launching Heating.sta...hine\_diagram



xtext



# Tools



# Tooling!





# Tooling!

## Editor



# Tooling!

## Editor, Debugger



# Tooling!

## Editor, Debugger, Testing



# Tooling!

Editor, Debugger, Testing, Groupware



# Tooling!

Editor, Debugger, Testing, Groupware,  
Scalable



# 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



INTENTIONAL<sup>®</sup>  
SOFTWARE

# xtext

**Open Source (EPL)**

**Eclipse-based, Eclipse Project**

**Very flexible, very popular!**

**Current Version 2.0:**

- improved performance**

- Xbase: expressions for reuse**

- Xtend2: „Better Java“, with support  
for Xpand-like templates**



**Open Source (Apache 2.0)**

**Projectional Editor**

**Very good at lang. Composition**

**Current Version 2.0:**

**Improved performance**

**Unified generate/compile/build**

**Debug MPS in MPS**

**Tables in the editor**

**(Diagrams planned for 2.1)**



**INTENTIONAL<sup>®</sup>**  
S O F T W A R E

**Commercial Tool.**

**Projectional Editor**

**Very flexible notations**

**Version 1.8 is current**

# Way More:

Spooifax

Rascal

oomega

The Whole Platform

see also

<http://languageworkbenches.net>

5

# 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

$$\frac{\partial}{\partial \theta} \mathbb{M}T(\xi) = \frac{\partial}{\partial \theta} \int_{\mathcal{R}_x} T(x) f(x, \theta) dx = \int_{\mathcal{R}_x} \frac{\partial}{\partial \theta} T(x) f(x, \theta) dx$$

$$\frac{\partial}{\partial a} \ln f_{a, \sigma^2}(\xi_1) = \frac{(\xi_1 - a)}{\sigma^2} f_{a, \sigma^2}(\xi_1) = \frac{1}{\sqrt{2\pi\sigma^2}} \left\{ \frac{\xi_1 - a}{\sigma^2} \right\} e^{-\frac{(\xi_1 - a)^2}{2\sigma^2}}$$

$$\int_{\mathcal{R}_x} T(x) \cdot \frac{\partial}{\partial \theta} f(x, \theta) dx = \mathbb{M} \left( T(\xi) \cdot \frac{\partial}{\partial \theta} \ln L(\xi, \theta) \right) = \int_{\mathcal{R}_x} T(x) \cdot \left( \frac{\partial}{\partial \theta} \ln L(x, \theta) \right) \cdot f(x, \theta) dx = \int_{\mathcal{R}_x} T(x) \cdot \left( \frac{\partial}{\partial \theta} \ln L(x, \theta) \right) \cdot f(x, \theta) dx$$

$$\frac{\partial}{\partial \theta} \mathbb{M}T(\xi) = \frac{\partial}{\partial \theta} \int_{\mathcal{R}_x} T(x) f(x, \theta) dx = \int_{\mathcal{R}_x} \frac{\partial}{\partial \theta} T(x) f(x, \theta) dx = \int_{\mathcal{R}_x} T(x) \cdot \left( \frac{\partial}{\partial \theta} f(x, \theta) \right) dx = \int_{\mathcal{R}_x} T(x) \cdot \left( \frac{\partial}{\partial \theta} \ln f_{a, \sigma^2}(\xi_1) \right) \cdot f_{a, \sigma^2}(\xi_1) dx$$

# 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

6

# A more theoretical View





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.

## 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] );
}
```

## 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 construct and treat it “correctly”.**

# 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;
            }
        }
    }
}
```

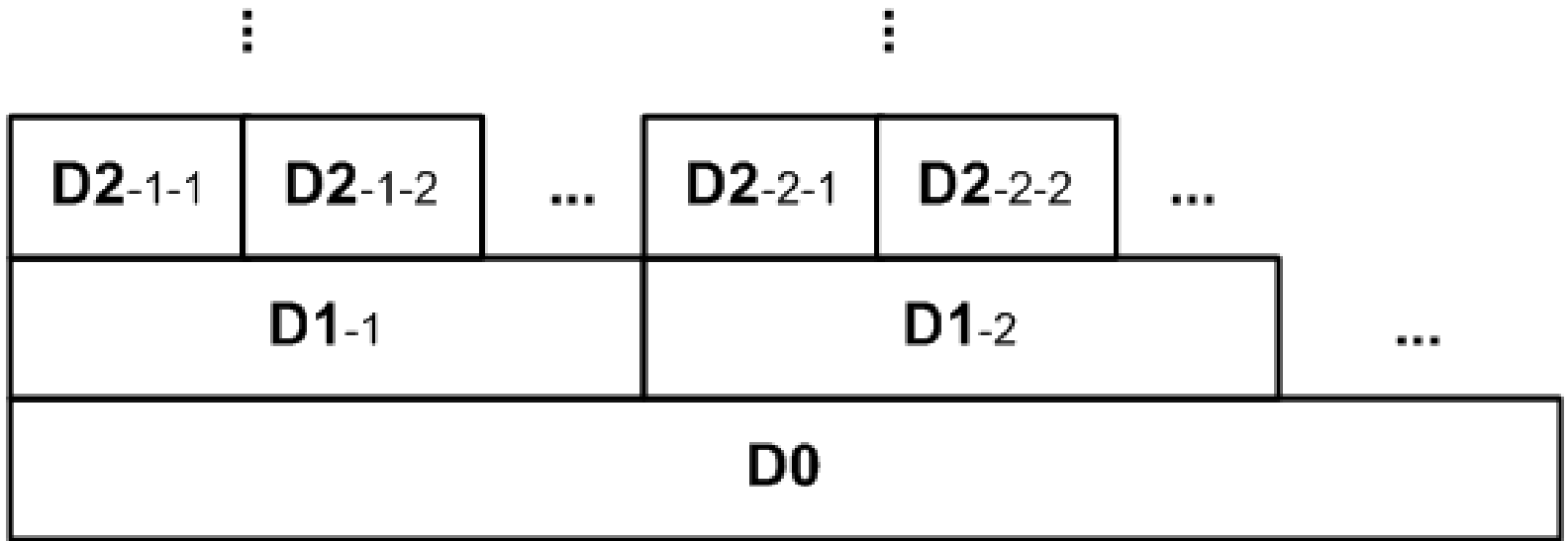
## Much better!

```
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>>
  }
}
```

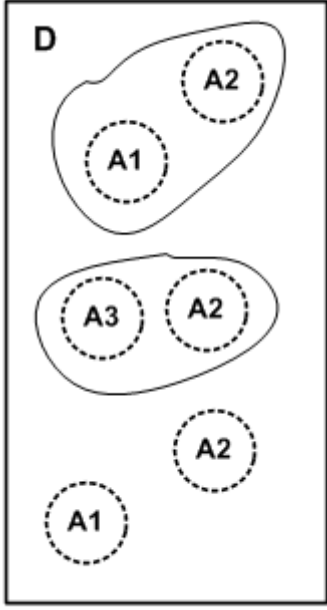
# Much better!

<b>linefollower</b>	<i>initializing</i>	<i>paused</i>	<i>running</i>	<i>crash</i>
<i>initialized</i>	true	running		
<i>bumped</i>			true	crash
<i>blocked</i>			true	paused
<i>unblocked</i>		true	running	

# Domains are Hierarchical

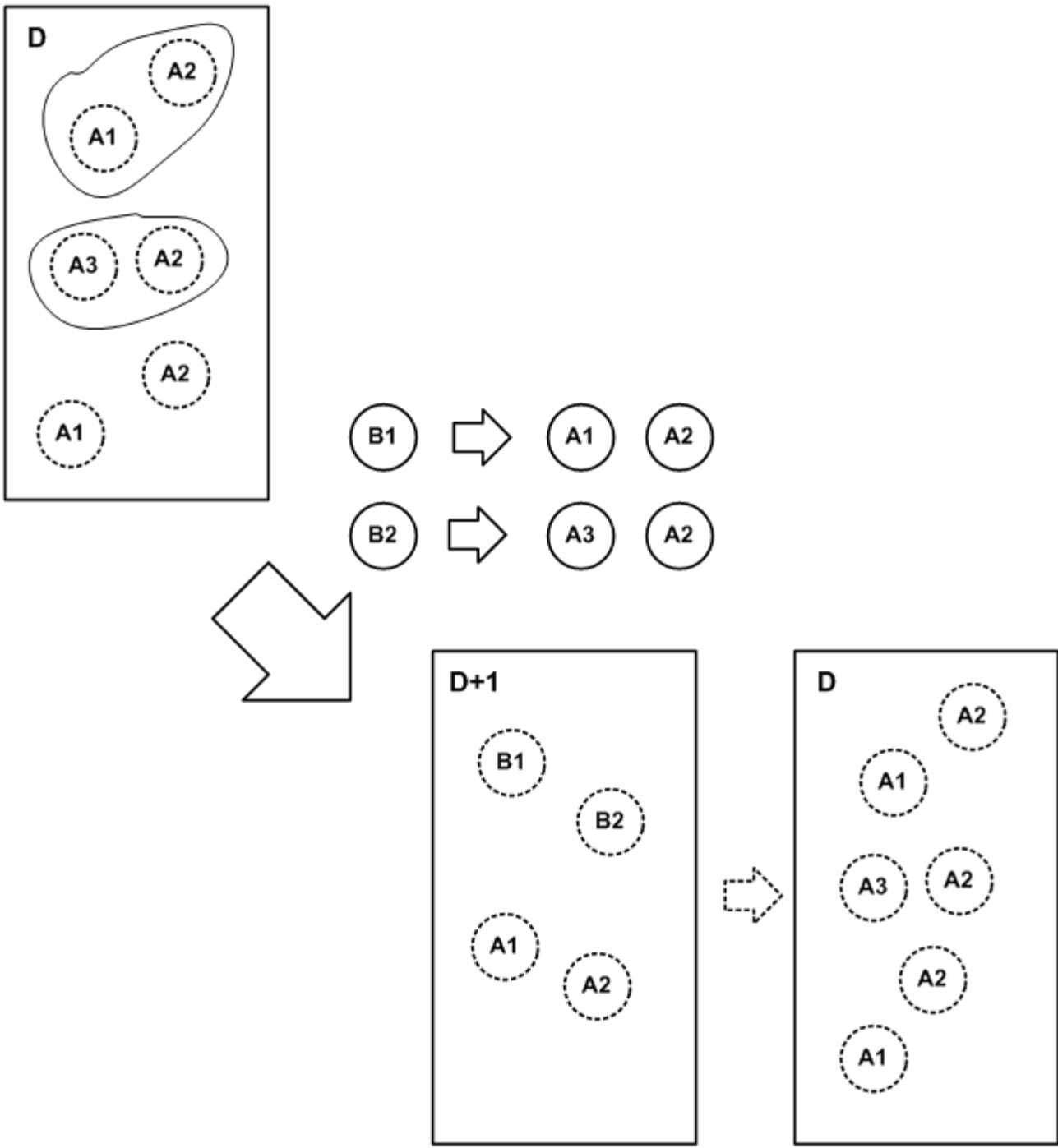


# Creating Linguistic Abstractions





# Creating Linguistic Abstractions

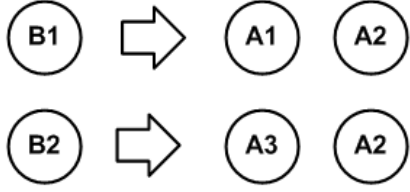


A DSL is a **language** at D that provides **linguistic abstractions** for **common patterns and idioms** of a language at D-1 when used within the domain D.

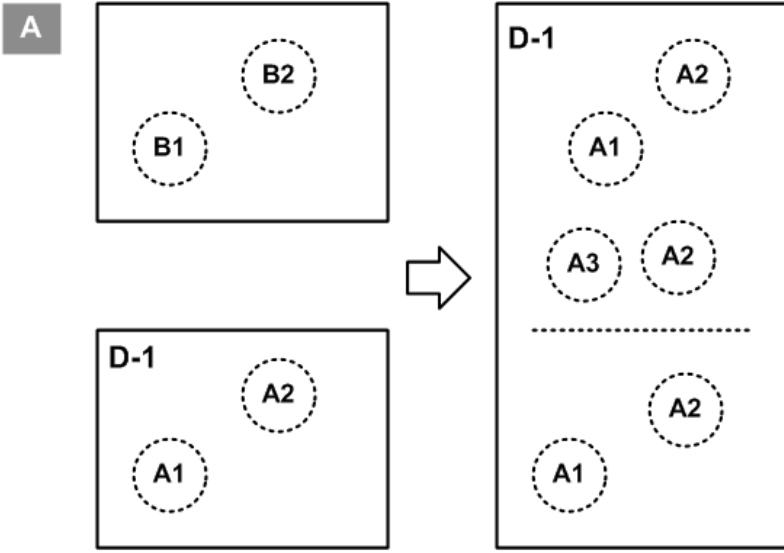
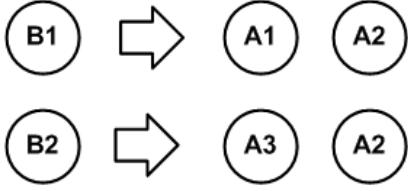
A DSL is a **language** at D that provides **linguistic abstractions** for **common patterns and idioms** of a language at D-1 when used within the domain D.

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.

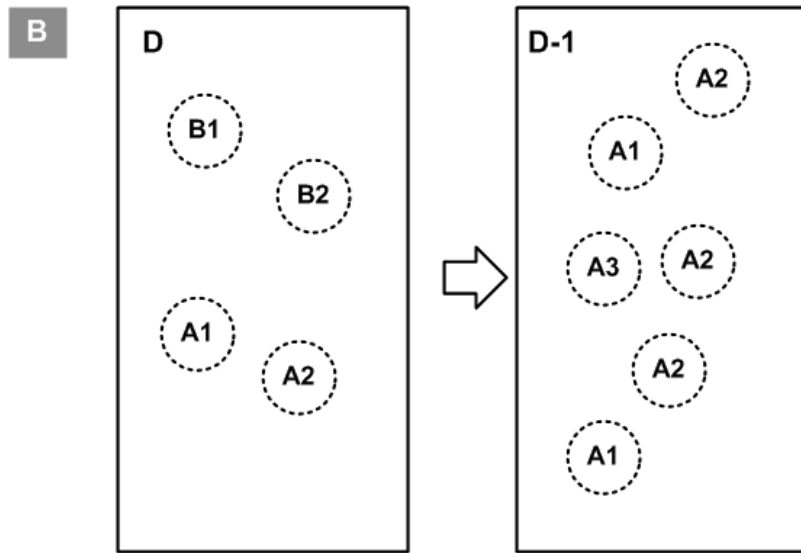
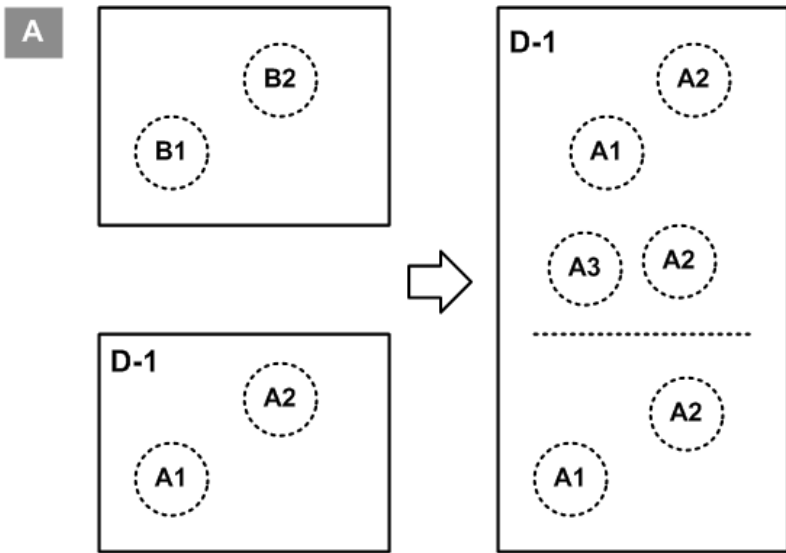
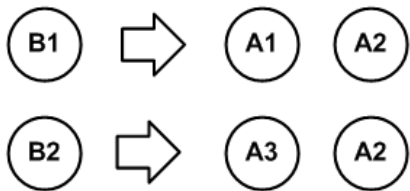
# Extension vs. Cascading



# Extension vs. Cascading



# Extension vs. Cascading



7

# Programming and Modeling



# Modeling



# Programming





# 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**

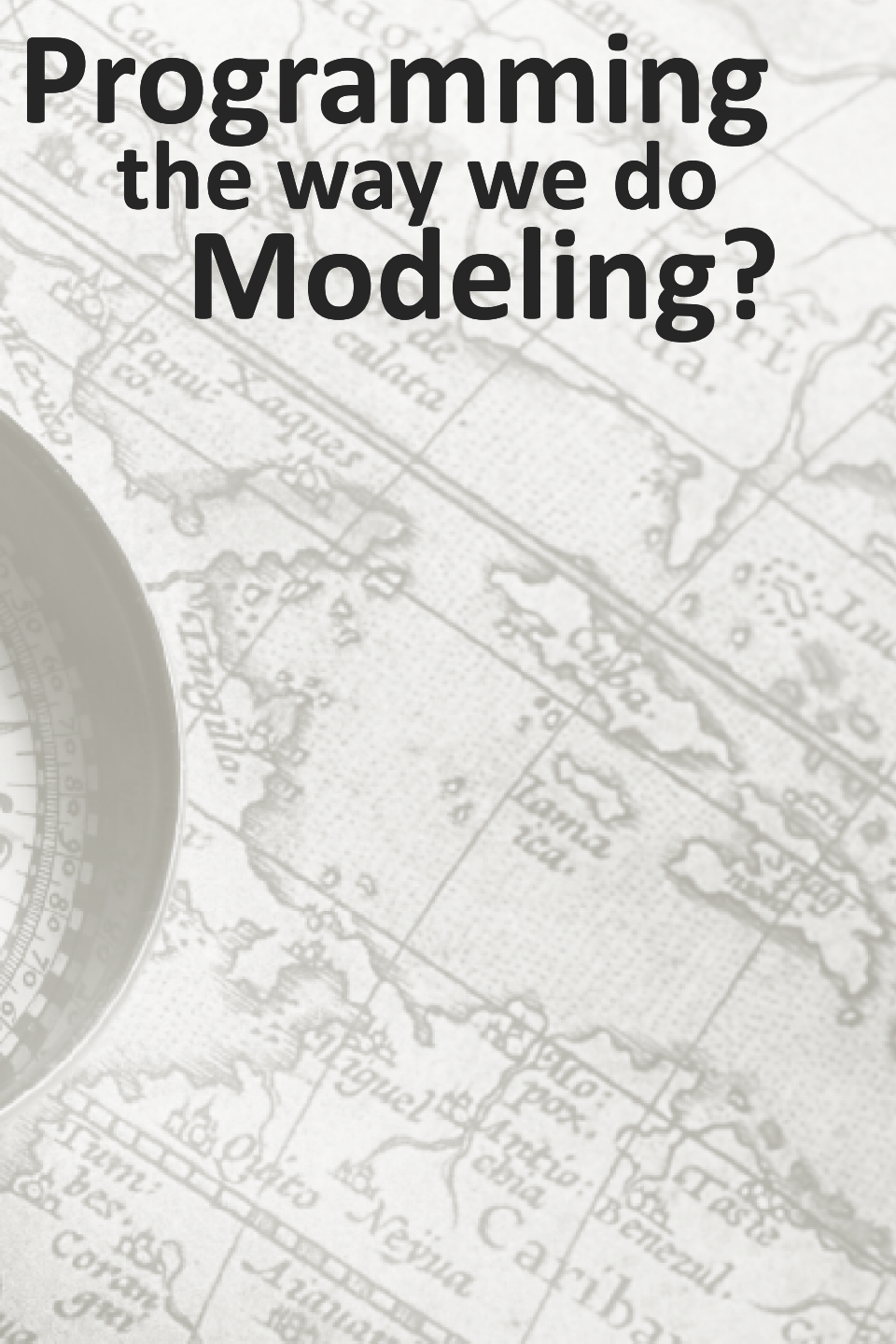
**Why**  
**the difference?**

It is time for ...



... a Different Perspective

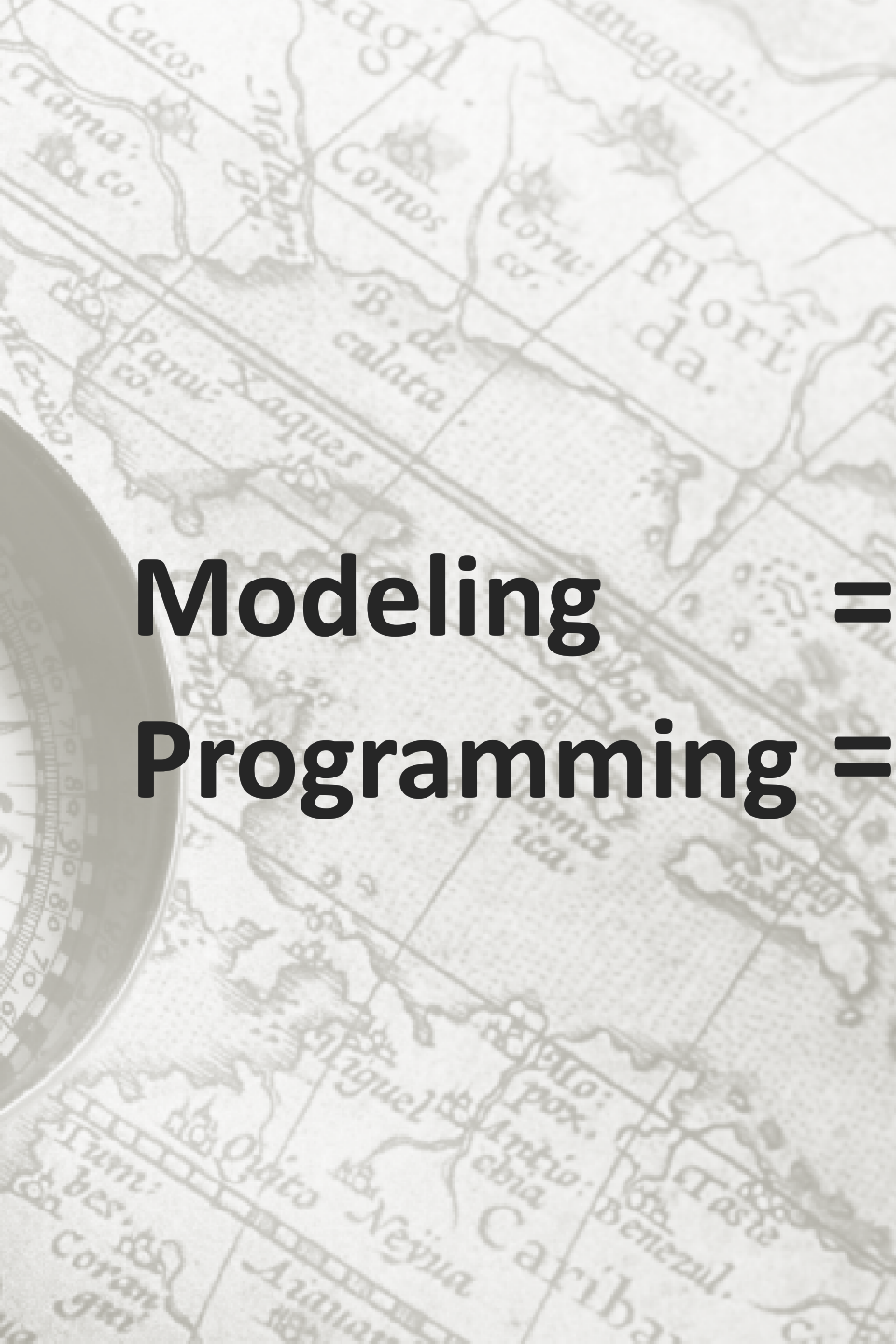




**Programming  
the way we do  
Modeling?**



**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**





Programming  
Languages  
are not

**MODULAR**  
enough.



Programming  
Languages  
are not

**COMPOSABLE**  
enough.



Programming  
Languages  
are not

**CONFIGURABLE**  
enough.





Programming  
Languages  
are not

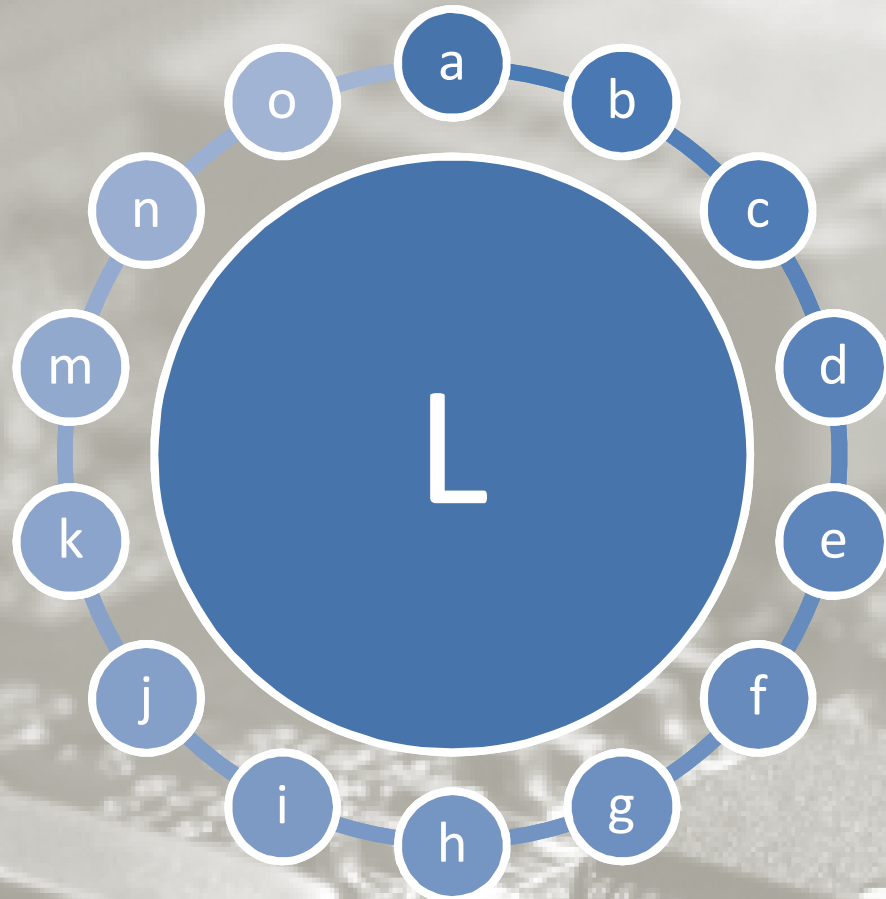
**ADAPTABLE**  
enough.



Programming  
Language Syntax  
is not

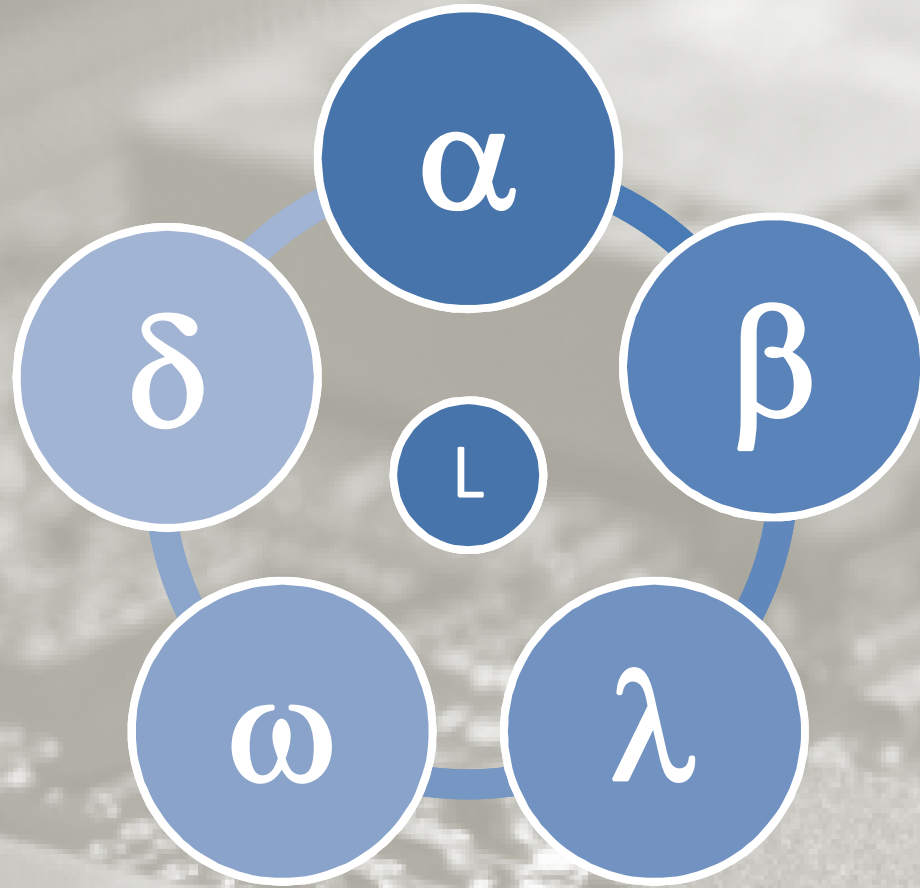
**FLEXIBLE**  
enough.

# Big Language?



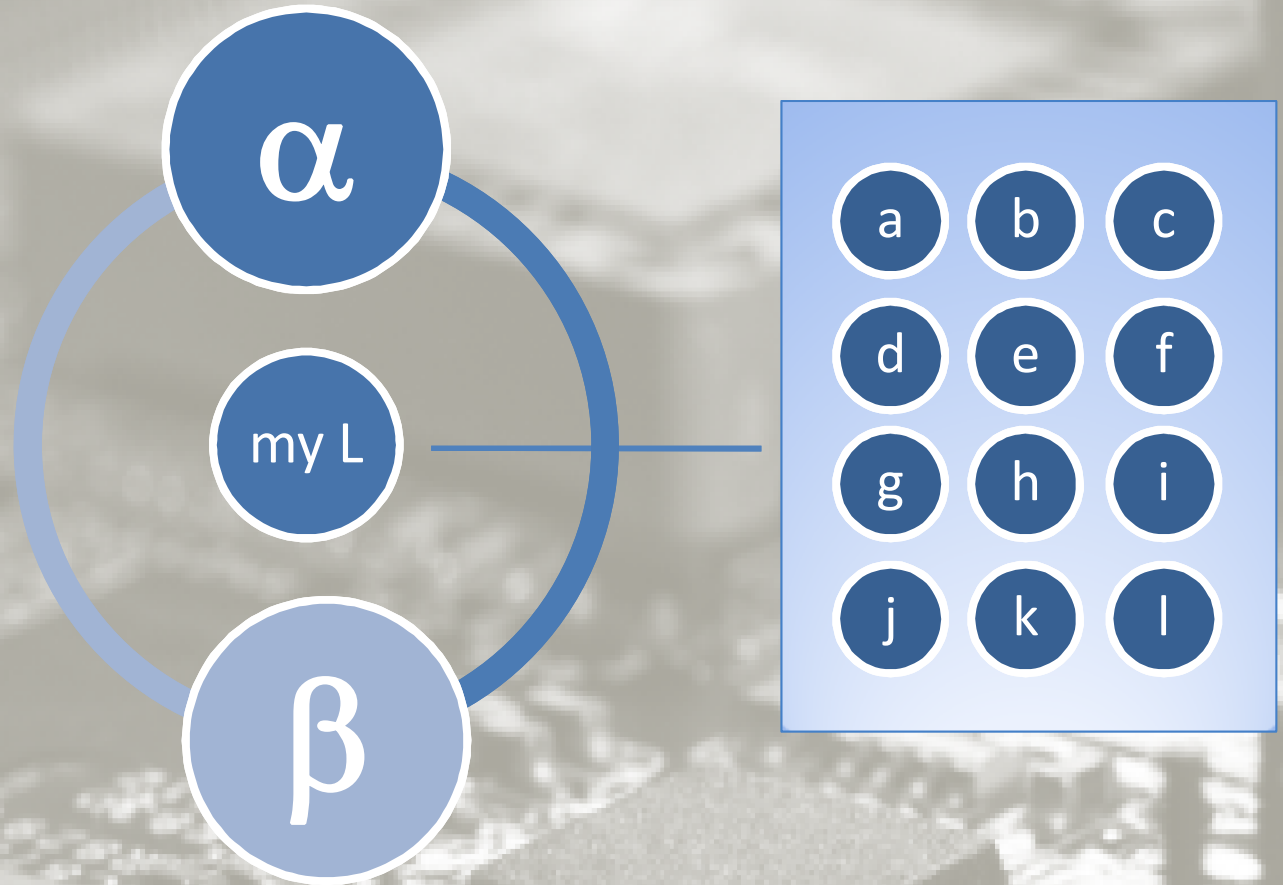
with **many** first class concepts!

# Small Language?



with a few, orthogonal  
and powerful concepts

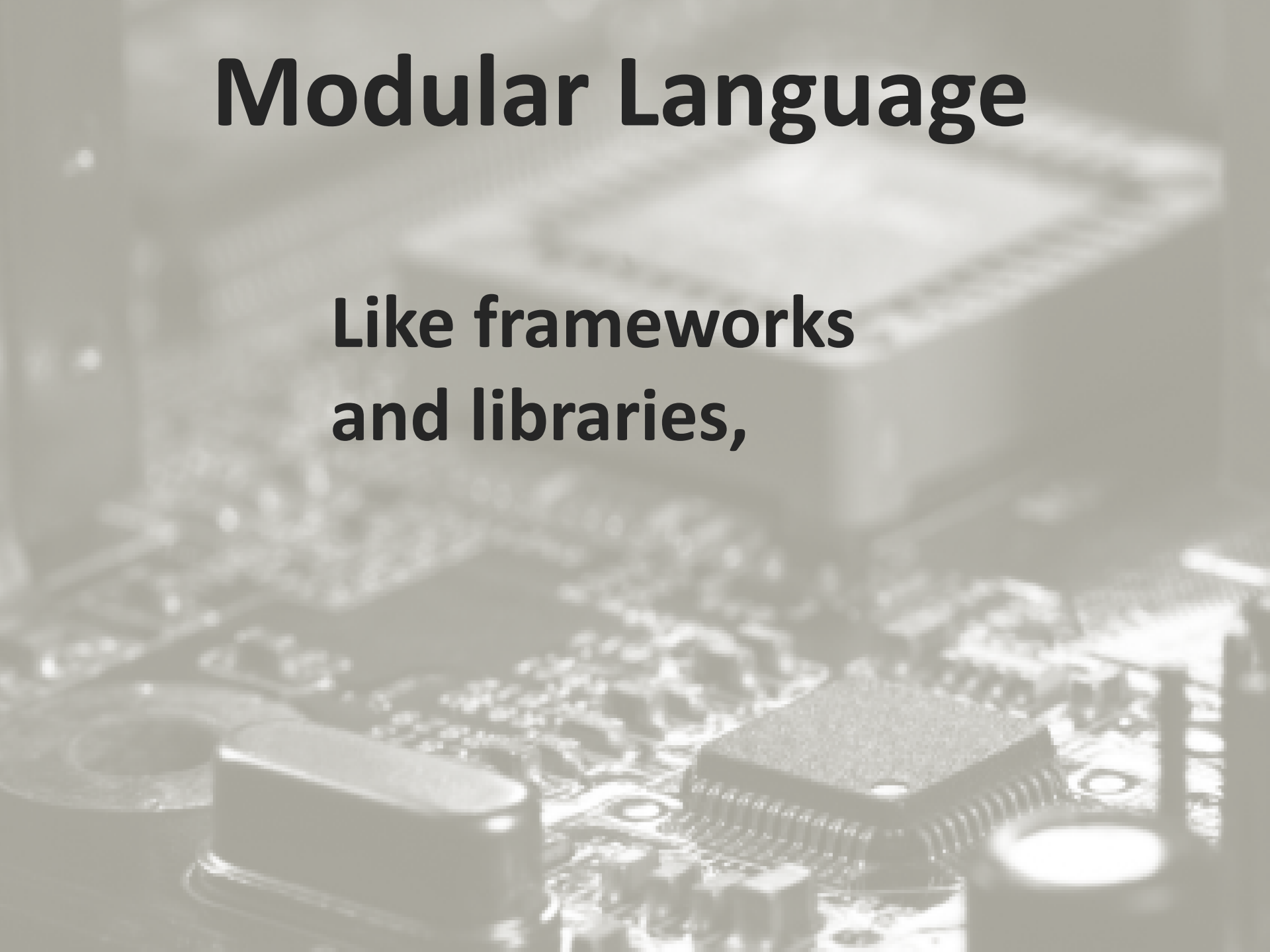
# Modular Language



with many **optional,**  
**composable** concepts

# Modular Language

Like frameworks  
and libraries,

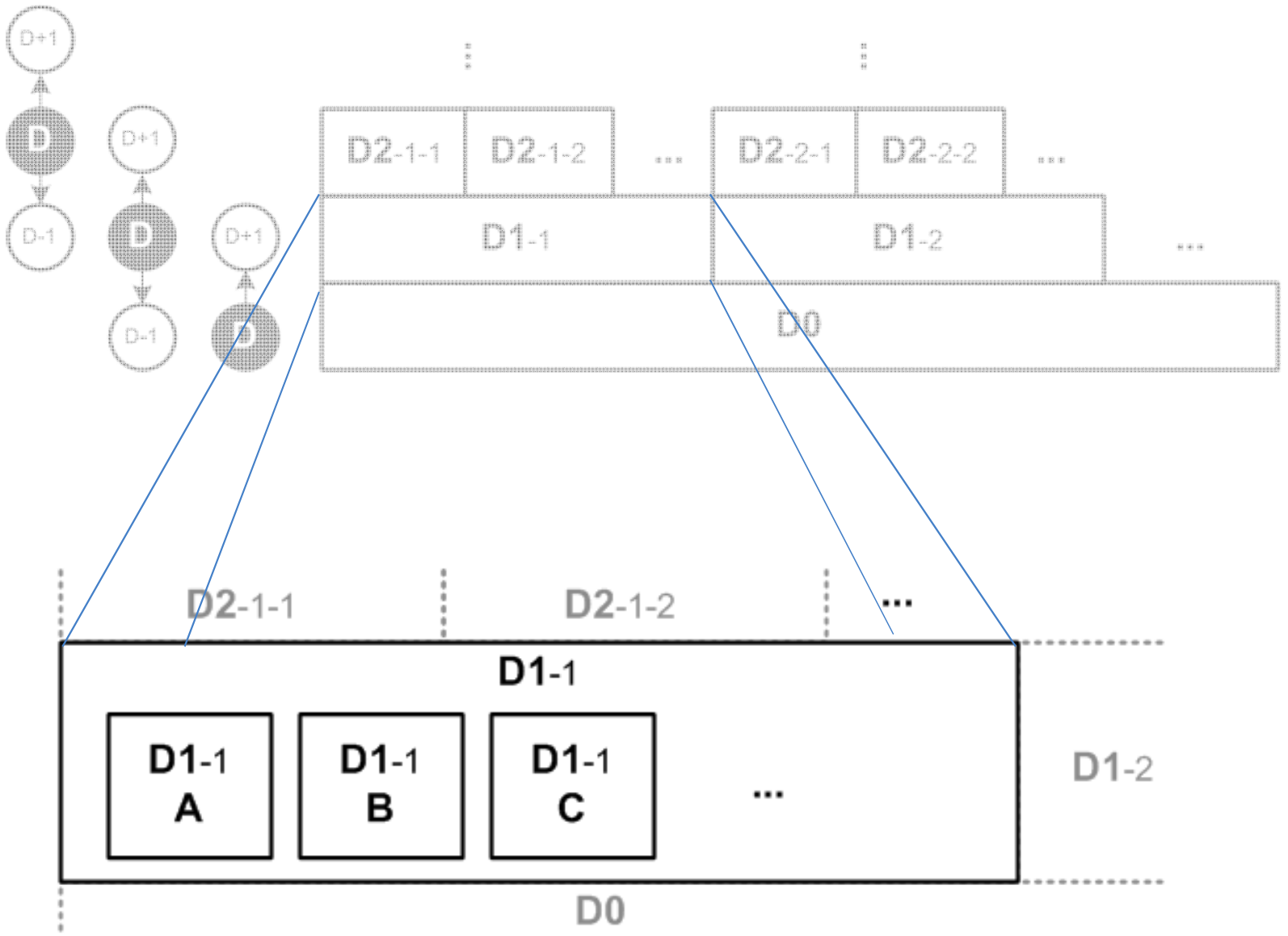


# Modular Language

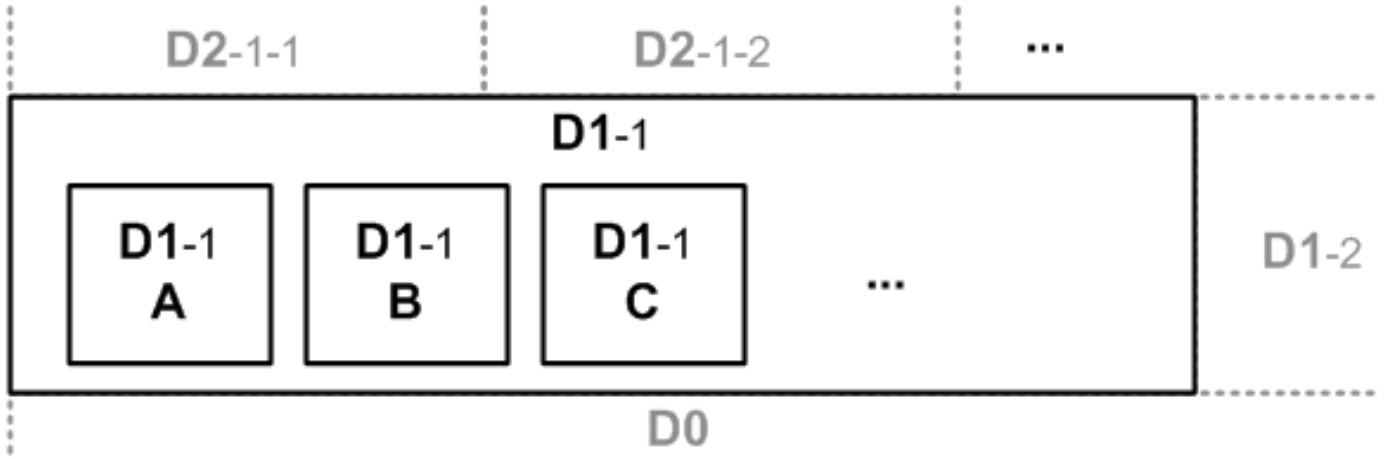
Like frameworks  
and libraries,

but with syntax  
and IDE support

# Several Concerns in a Domain

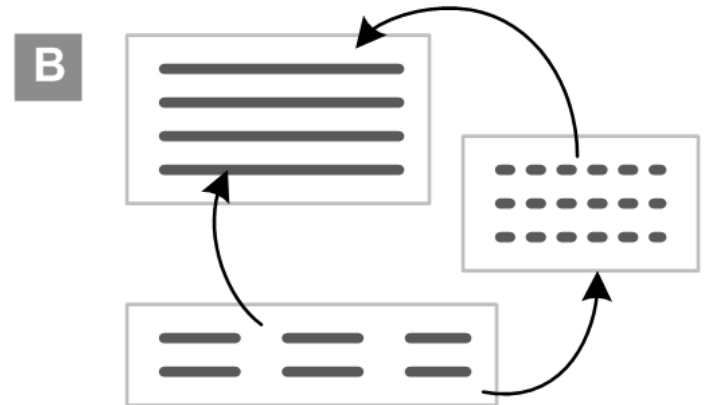
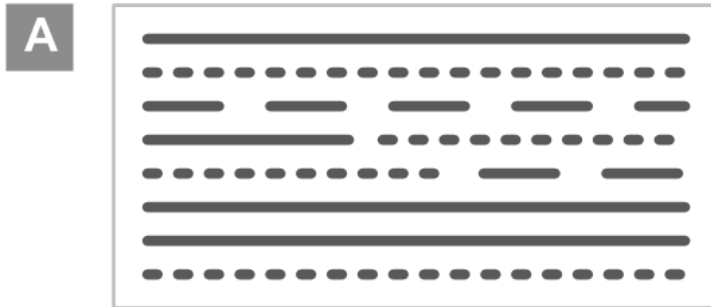
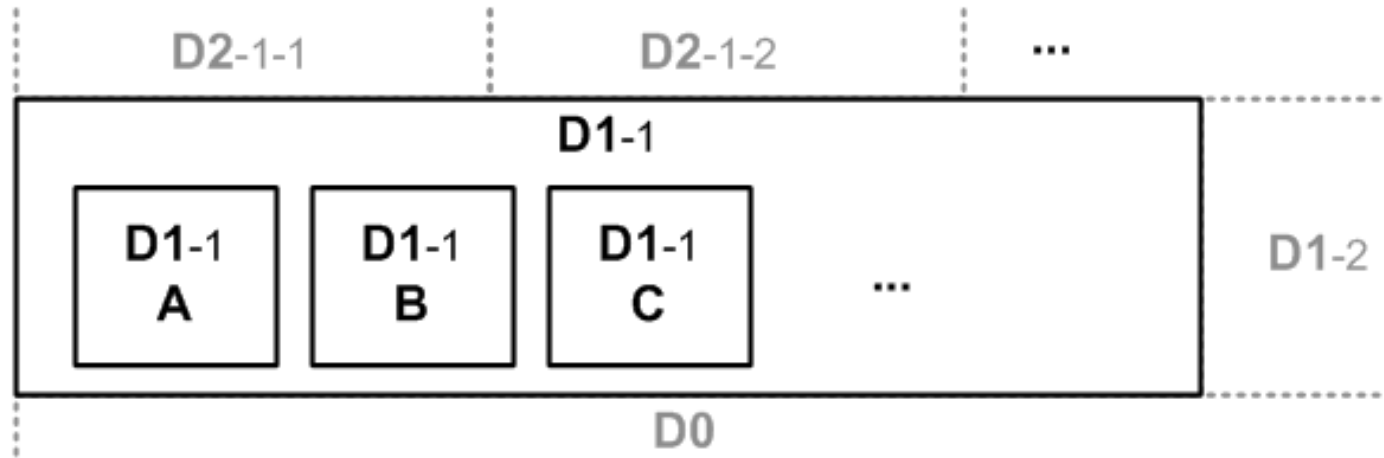






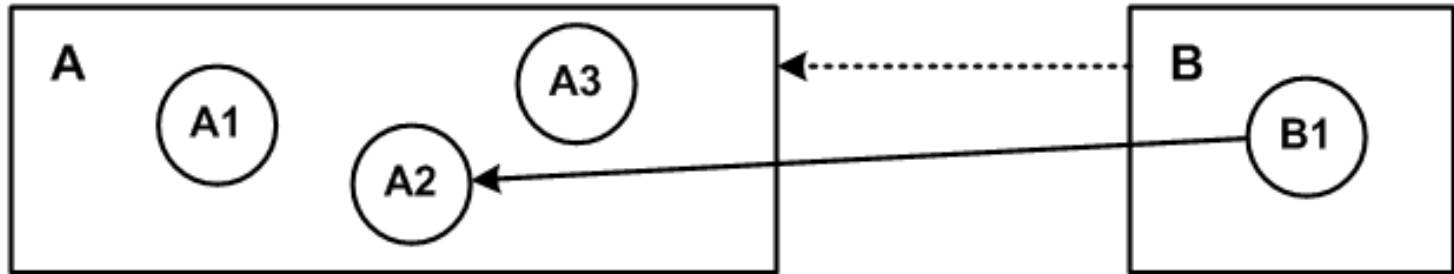
# Viewpoints

# Viewpoints

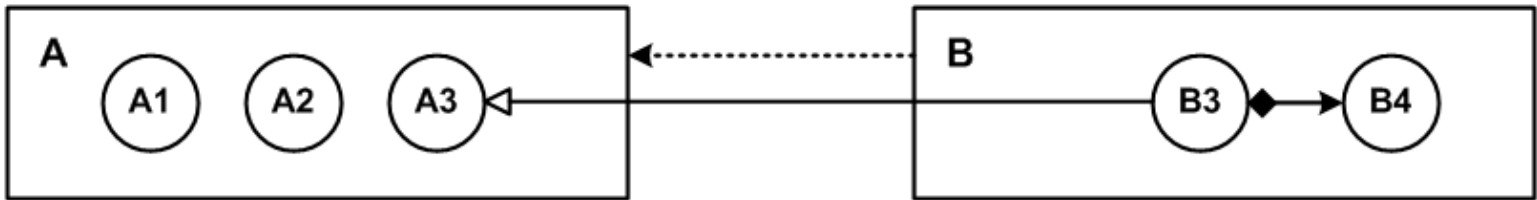


**A:** mixed

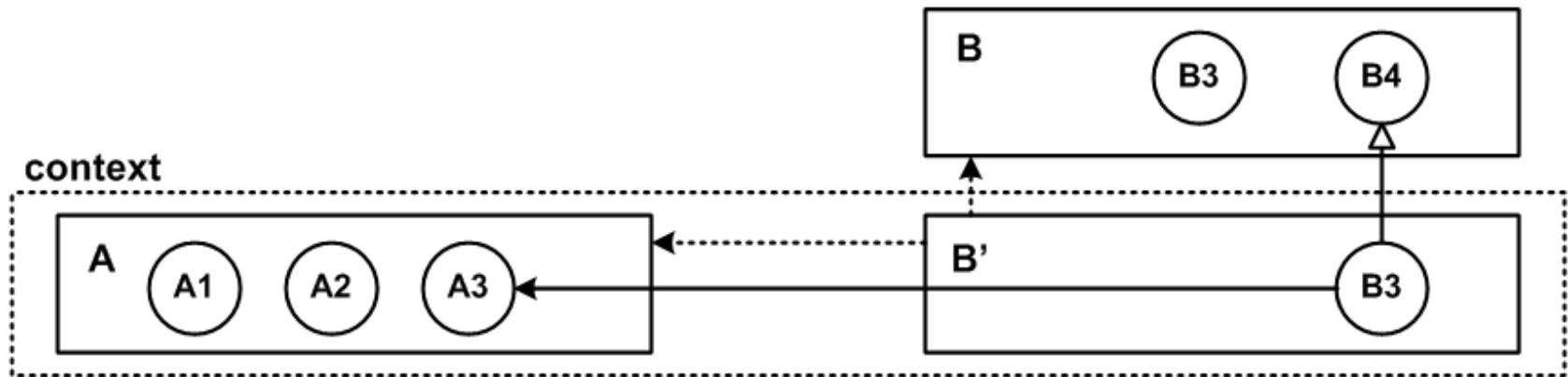
**B:** separate Viewpoints



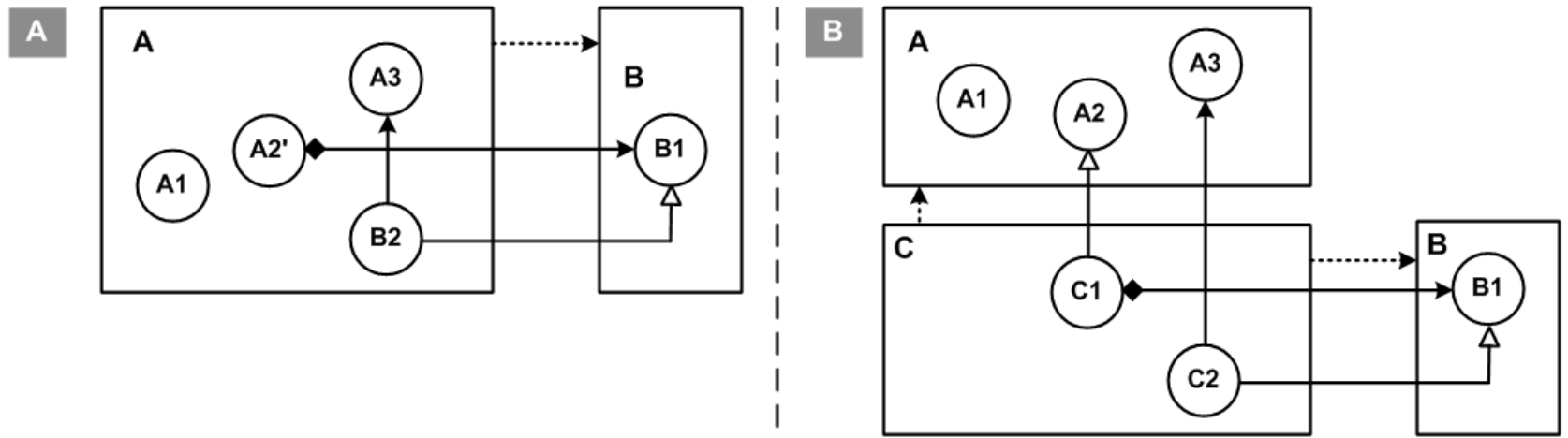
When DSL (for several concerns) are developed from scratch, as a group, then dependencies between the concerns can be materialized as dependencies between the languages and the language concepts



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.



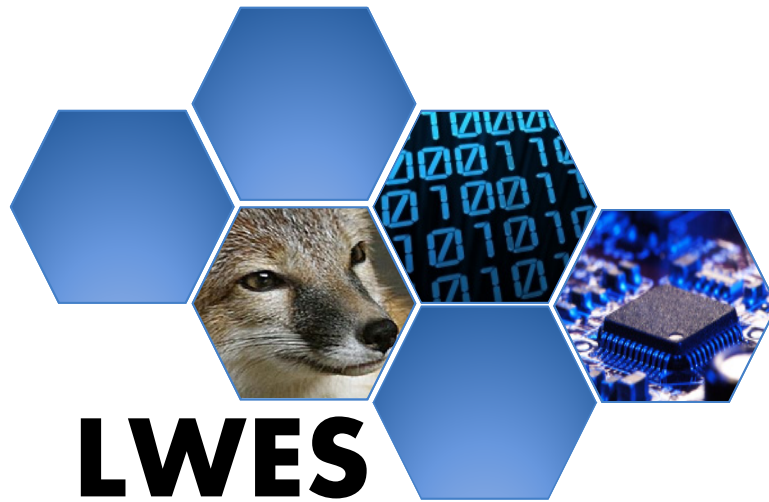
A language has been developed to be used in contexts not known at the time of development. No dependencies allowed! The reusable language has to be extended so it can reference concepts from languages in that context.



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

# The LWES Project





# **LWES**

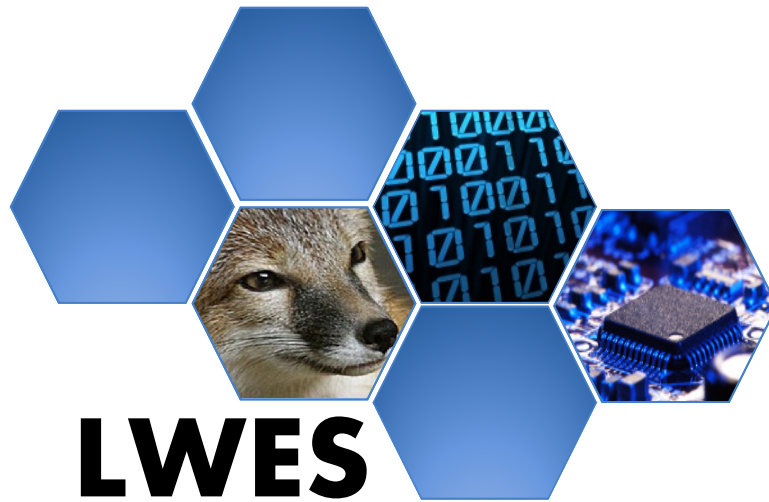
**Language Workbenches**  
*for* **Embedded Systems**



---

<http://mbeddr.com>





**LWES**

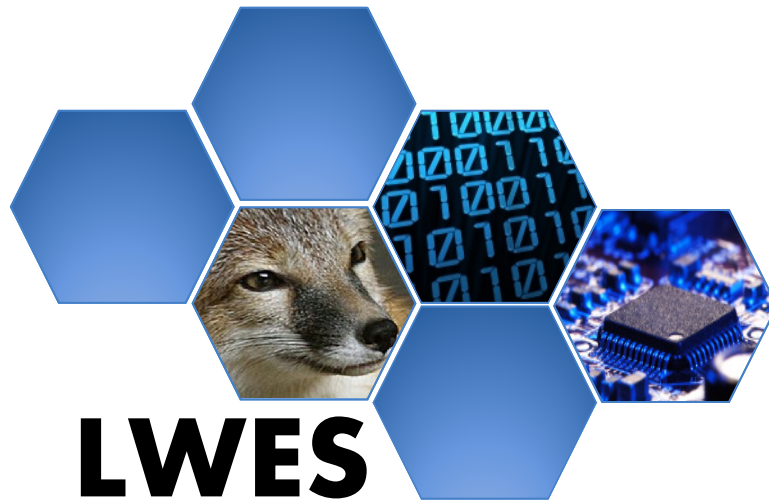
**Language Workbenches  
for Embedded Systems**



<http://mbeddr.com>

---

# **Incremental Extension of C with DSLs for Embedded Systems, integrated with Formal Methods and support for PLE and Requirements Tracing**



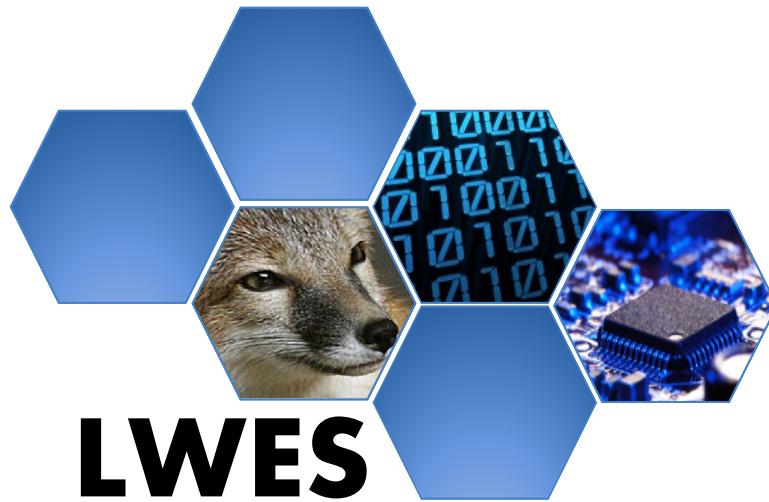
# LWES

Language Workbenches  
*for* Embedded Systems



<http://mbeddr.com>





# LWES

Language Workbenches  
*for* Embedded Systems



<http://mbeddr.com>

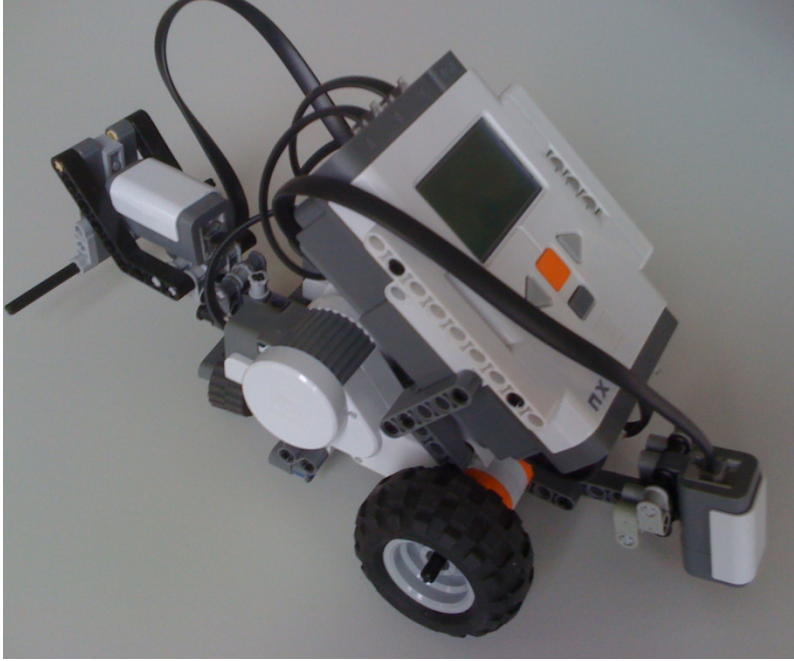
itemis

**fortiss**  
innovation in software and systems

**SICK**  
Sensor Intelligence.

**LEAR**  
CORPORATION

# EXAMPLE CASE



*doc This module represents the code for the line follower lego robot. It has a couple*

```
module main imports OsekKernel, EcAPI, BitLevelUtilities {
```

```
    constant int WHITE = 500;
```

```
    constant int BLACK = 700;
```

```
    constant int SLOW = 20;
```

```
    constant int FAST = 40;
```

*doc State machine to manage the t*

```
state machine linefollower {
```

```
    event initialized;
```

```
    initial state initializing {
```

```
        initialized [true] -> running
```

```
    }
```

```
    state running {
```

```
    }
```

```
}
```

```
initialize {
```

```
    ecrobot_set_light_sensor_active
```

```
    event linefollower:initialized
```

```
}
```

*doc This is the cyclic task that is called every 1ms to do the actual control*

```
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);
```

```
}
```

```
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);  
    }  
}
```

```

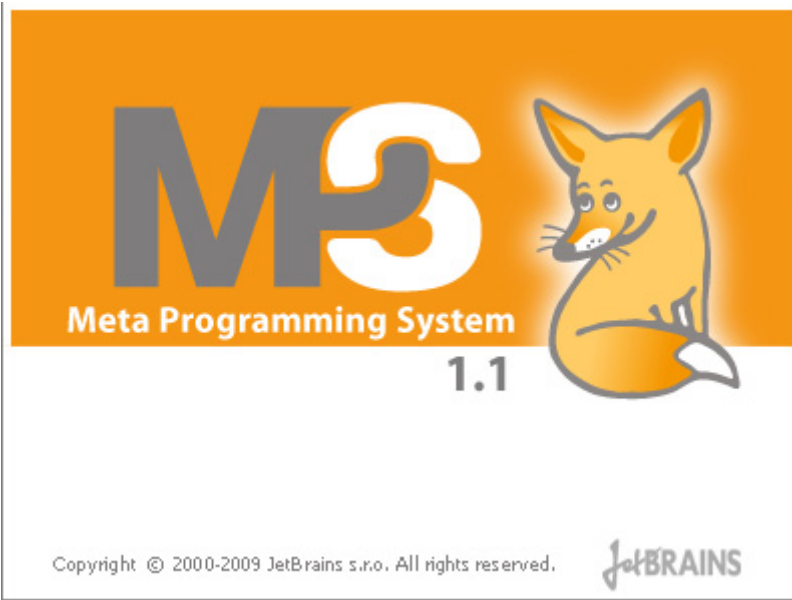
module impl imports <<imports>> {

    int speed( int val ) {
        return 2 * val;
    }

    robot script stopAndGo
        block main on bump block retreat on bump <no bumpReaction>
            stop
            accelerate to 0 - 30 within 2000
            drive on for 2000
            decelerate to 0 within 1000
            stop
            accelerate to speed(25) within 3000
            drive on for 2000
            turn left for 2000
            block driveMore on bump <no bumpReaction>
                accelerate to 80 within 2000
                turn right for 3000
            decelerate to 0 within 3000
            stop

}

```



# JetBrains MPS





## First C Code working

July 17, 2011 by mpscmcd

As you may know, our project relies on the idea of extending the C programming language with domain specific extensions. For that to work, we first have to make C available in MPS. While we had done this to some extent in our proof of concept, we are now implementing C much more thoroughly. As you can see in the screenshot below, some essential things are already working.

```
TestModule -  
Module TestModule {  
  void test(int a, int theThird, int f)  
  {  
    int x = a + 2;  
    int a1;  
    int a2 = 2 + 3;  
    int a3 = a2 + a2 * a1;  
    { ... }  
  }  
  int c1 = a1;  
  { ... }  
  for ( int i = 0; i < 20; i++)  
  {  
    int x = 1;  
    int y = x;  
    ...  
  }  
}
```

## ARCHIVES

- [July 2011 \(3\)](#)
- [June 2011 \(2\)](#)
- [January 2011 \(2\)](#)
- [July 2010 \(1\)](#)
- [June 2010 \(2\)](#)

## CATEGORIES

- [code \(3\)](#)
- [demos'n'stuff \(4\)](#)
- [dev progress \(1\)](#)
- [news \(4\)](#)
- [Uncategorized \(2\)](#)

## PAGES

# Bonus: Best Practices



# Limit Expressiveness



# Notation, Notation, Notation

$$\frac{\partial}{\partial \theta} \mathbf{M}T(\xi) = \frac{\partial}{\partial \theta} \int_{\mathcal{R}_n} T(x) f(x, \theta) dx = \int_{\mathcal{R}_n} \frac{\partial}{\partial \theta} T(x) f(x, \theta) dx$$

$$\frac{\partial}{\partial a} \ln f_{a, \sigma^2}(\xi_1) = \frac{(\xi_1 - a)}{\sigma^2} f_{a, \sigma^2}(\xi_1) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left\{-\frac{(\xi_1 - a)^2}{2\sigma^2}\right\}$$

$$\int_{\mathcal{R}_n} T(x) \cdot \frac{\partial}{\partial \theta} f(x, \theta) dx = \mathbf{M}\left(T(\xi) \cdot \frac{\partial}{\partial \theta} \ln L(\xi, \theta)\right) = \int_{\mathcal{R}_n} \frac{\partial}{\partial \theta} T(x) f(x, \theta) dx$$

$$\int_{\mathcal{R}_n} T(x) \cdot \left(\frac{\partial}{\partial \theta} \ln L(x, \theta)\right) \cdot f(x, \theta) dx = \int_{\mathcal{R}_n} T(x) \cdot \left(\frac{\partial}{\partial \theta} \frac{f(x, \theta)}{f(x, \theta)}\right) \cdot f(x, \theta) dx$$

$$\frac{\partial}{\partial \theta} \mathbf{M}T(\xi) = \frac{\partial}{\partial \theta} \int_{\mathcal{R}_n} T(x) f(x, \theta) dx = \int_{\mathcal{R}_n} \frac{\partial}{\partial \theta} T(x) f(x, \theta) dx = \int_{\mathcal{R}_n} T(x) \cdot \frac{\partial}{\partial \theta} f(x, \theta) dx = \int_{\mathcal{R}_n} T(x) \cdot \frac{\partial}{\partial \theta} \ln f_{a, \sigma^2}(\xi_1) \cdot f_{a, \sigma^2}(\xi_1) dx$$



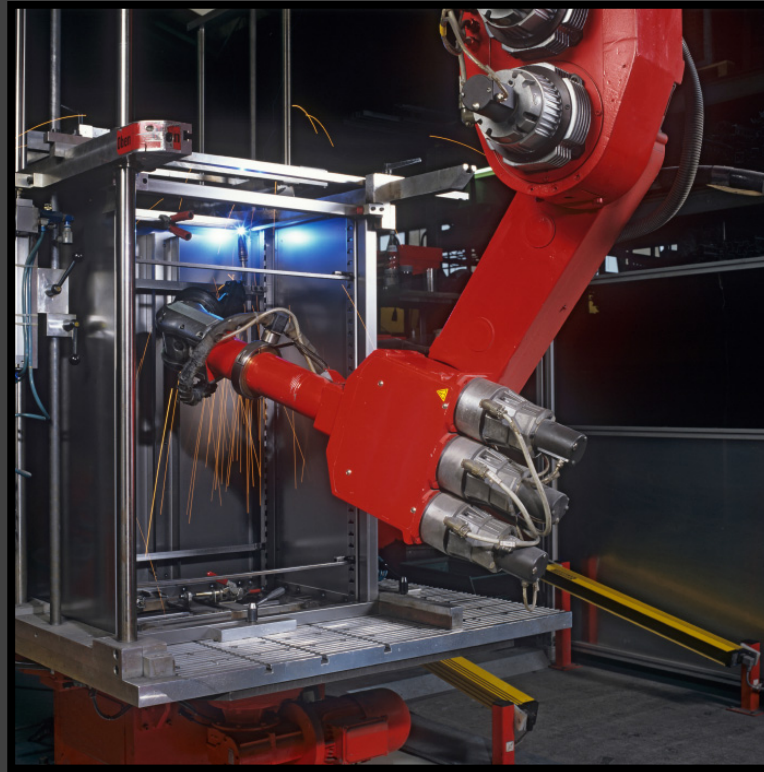
# Viewpoints



# Teamwork Support



# Interpretation vs. Generation





# Rich Domain Specific Platform



# Checks First and Separate



# Cascading



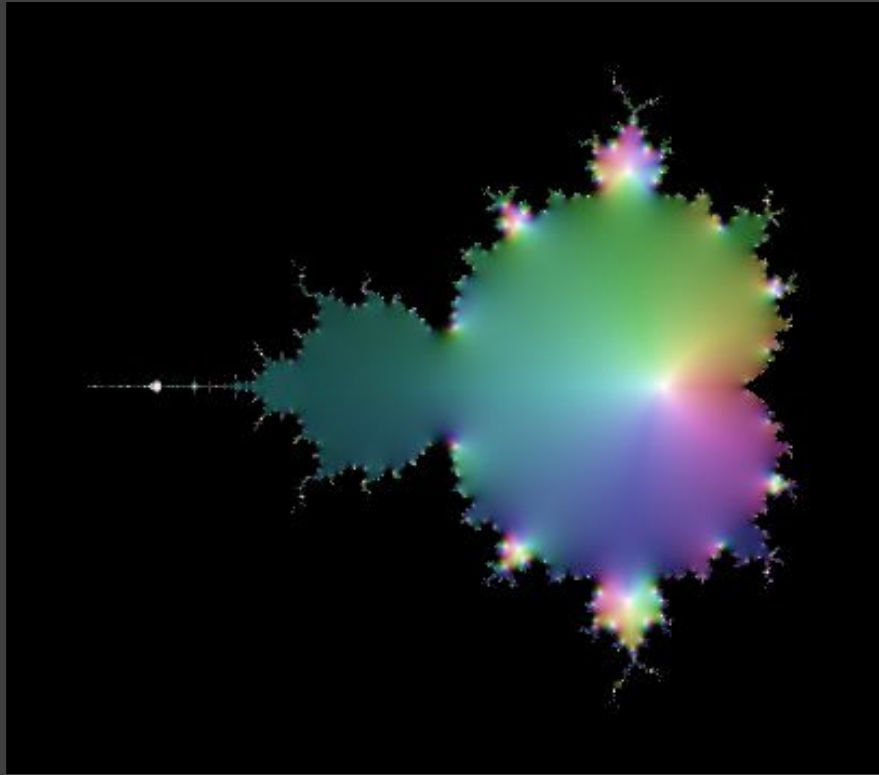
# Annotation Models



# Don't forget Testing



# Iterate!



# Co-Evolve Language and Concepts



# Domain Users Programming?





# Compatible Organization



# THE END.

**.coordinates**

**web** [www.voelter.de](http://www.voelter.de)

**email** [voelter@acm.org](mailto:voelter@acm.org)

**twitter** [markusvoelter](https://twitter.com/markusvoelter)

**xing** [http://www.xing.com/profile/Markus\\_Voelter](http://www.xing.com/profile/Markus_Voelter)

**linkedin** <http://www.linkedin.com/pub/0/377/a31>

