A couple of thoughts on PLE and models and languages …

MAPLE/SCALE 2011

Markus Voelter
Independent/itemis
voelter@acm.org

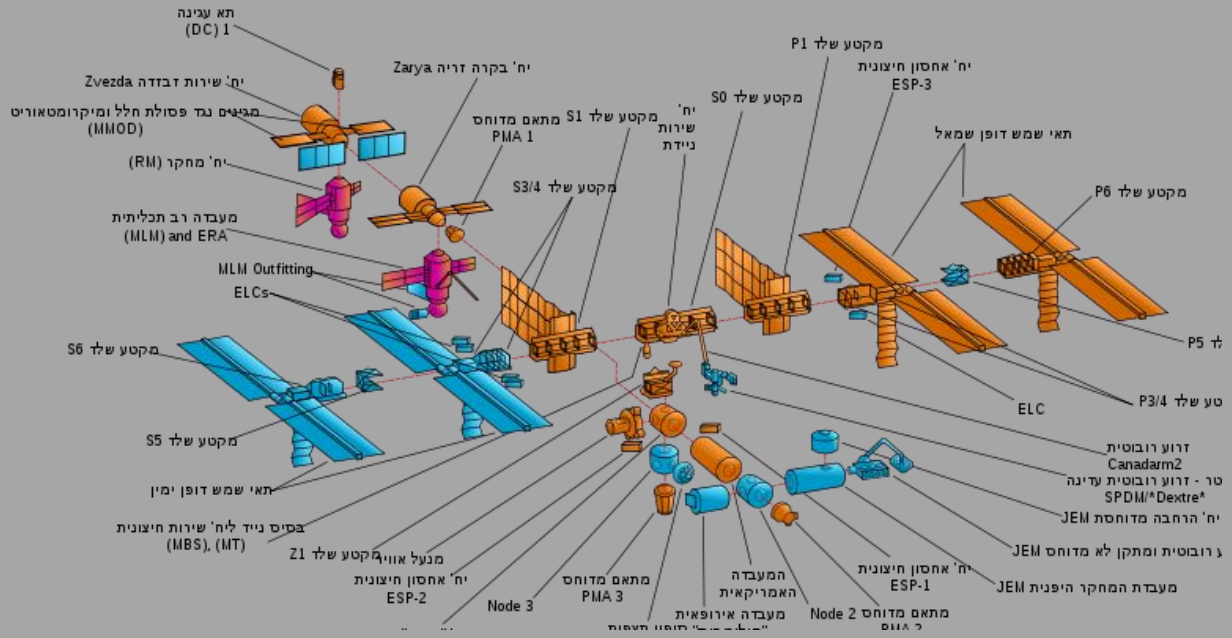
Do you build those?

PLE seems heavyweight.

Lightweight?

Agile?

Can we reduce scope?
Less Cross-Cutting?



Configuration Management vs. Variant Management

Is PLE Like IKEA?

i.e. selection and assembly?

Is PLE Like IKEA?

i.e. selection and assembly?

just Feature Models?

Is PLE Like IKEA?

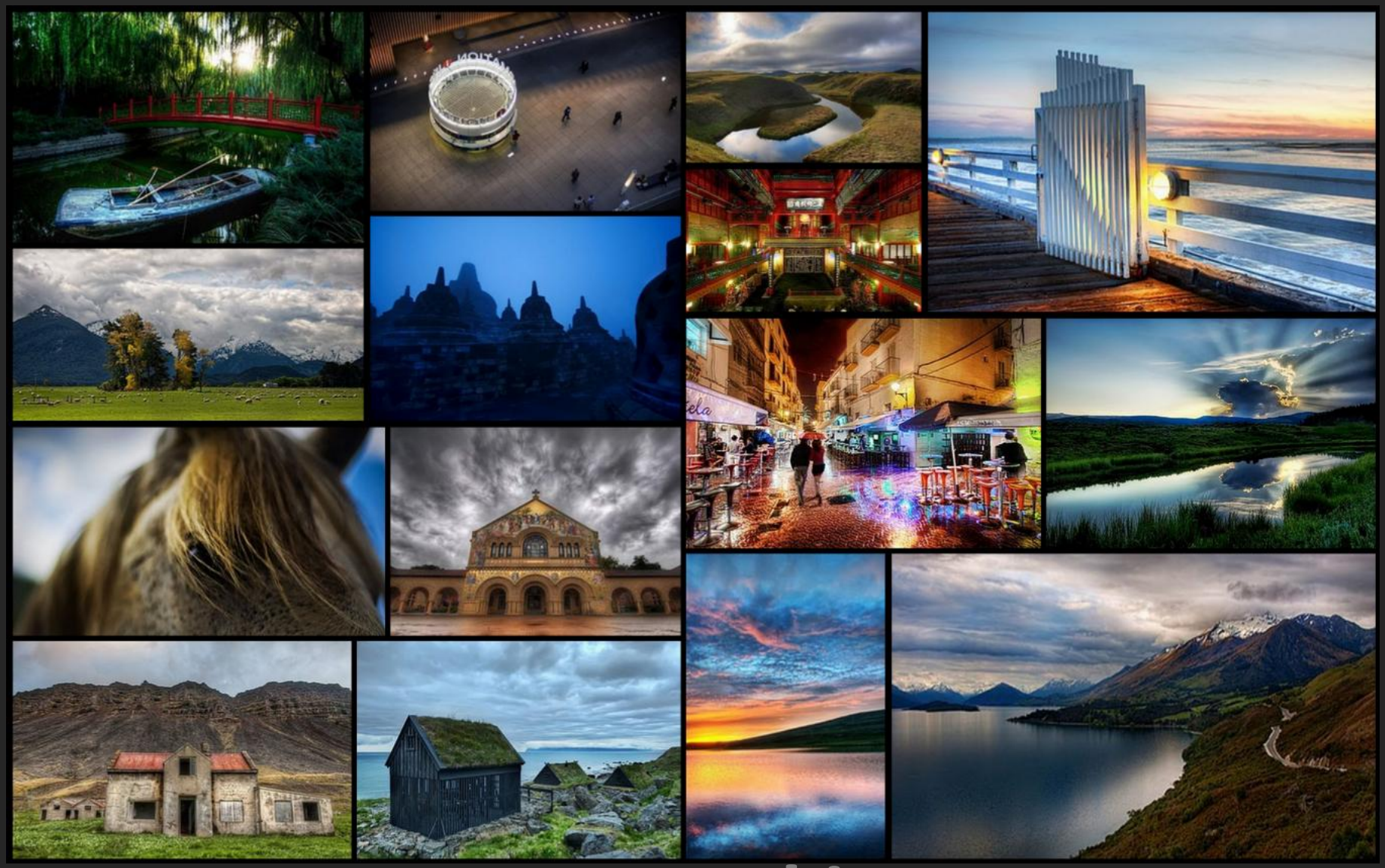
i.e. selection and assembly?

just Feature Models?

and otherwise raw code?

Configuration vs. Customization

Trey Ratcliff

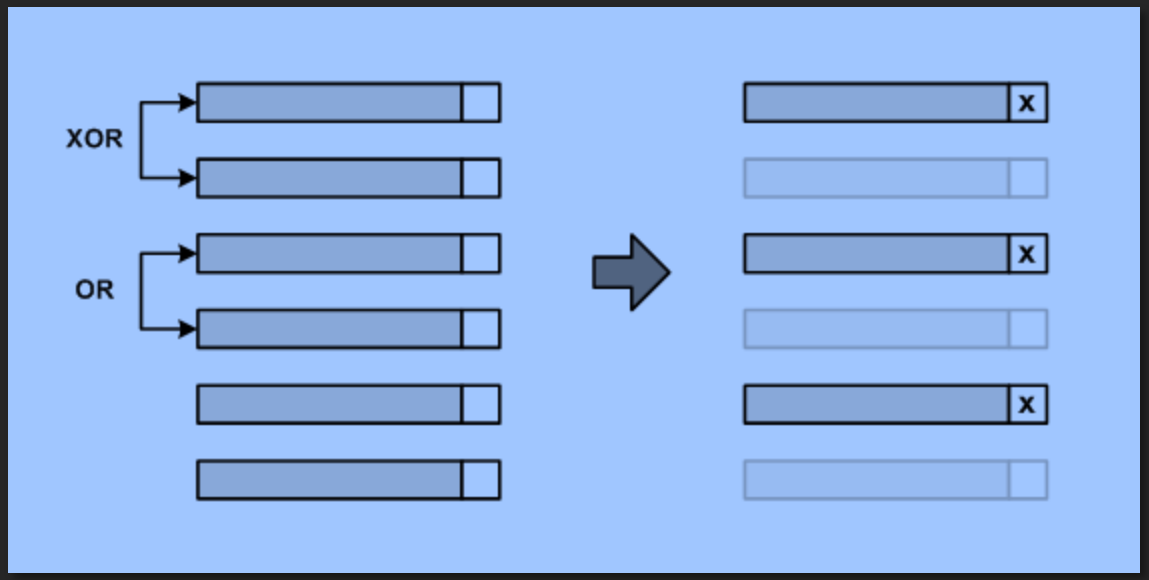


stuckincustoms.com

Selection???

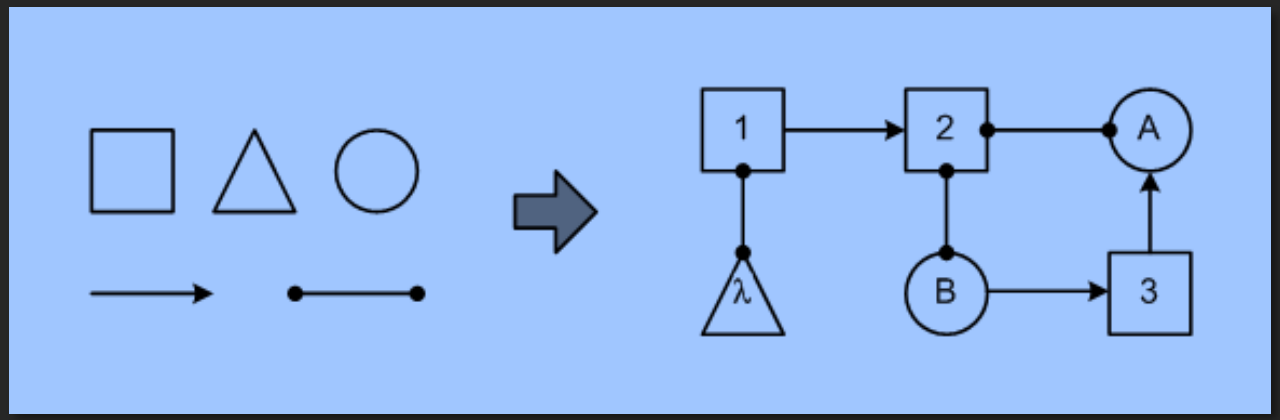
Configuration

… selecting options
… setting param values



Customization

… DSLs
… instantiation
… connections



Domain Specific Languages (DSLs)

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.

Testing

tests

```
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;
    };
};
```

Pension Fund Specification

Capgemini Pension Workbench

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

NNLCPA-14w2-21112008 * ×

Table of Contents ×

- Library
  - Documentation
  - Foundation
    - Value sets
      - Value set Groottebepalingsmethode
        - Value set member Salaris-diensttijd
        - Value set member Verzekerde bedrage
        - Value set member Afgeleide toezeggin
      - 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

All

¤ 3.3 Commutatiegetallen op 1 leven¶

$$D_x = v^x * \frac{l_x}{100}$$ ¤6 Dec(3)¶

Implemented in ¤ V9401¶

¶

$$N_x = \sum_{t=0}^{\omega - x} D_{x+t}$$ ¤7 Dec(3)

¶

¤ 3.6 Contante waarde 1 leven/ 2 levens¶

$$_{n}E_x = \frac{D_{x+n}}{D_x}$$ ¤19 Dec(4)

¶

$$a_x = \ddot{a}_x - 1$$ ¤21 Dec(3)

¶

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

¶

$$\ddot{a}_{x\overline{n}|} = \frac{N_x - N_{x+n}}{D_x}$$ ¤23 Dec(3)

$$\bar{a}_{x\overline{n}|} = \ddot{a}_{x\overline{n}|} - 0{,}5 + 0{,}5 * {}_{n}E_x$$ ¤25 Dec(3)

¶

¤ 4 BN(_ris) koopsommen¶

Section ▸ title ▸ Paragraph : Text Dev

Doc | Splitter | Pension | PensionDecorated | AM

Component Architecture

```
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"
    }

}
```

647x551

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

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

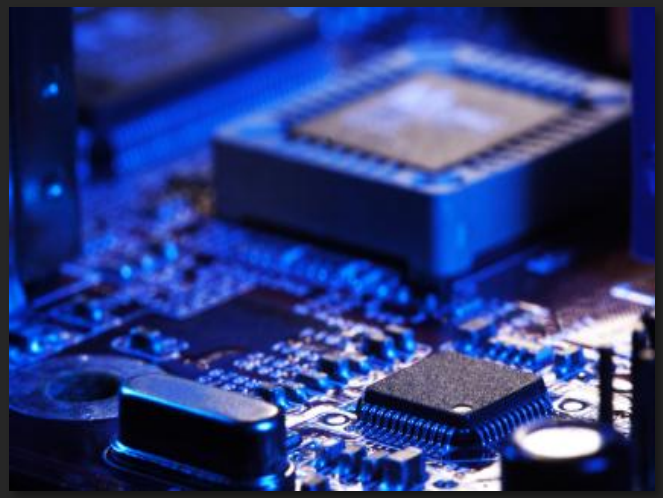
Fountain Behaviour

```
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
        }
}
```

Automation Is Key

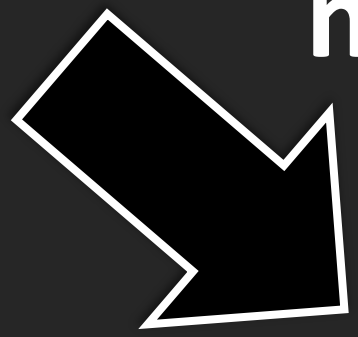
execute?

map



DSL Program
(aka Model)

map



automated!

GPL Program

Problem Space

map

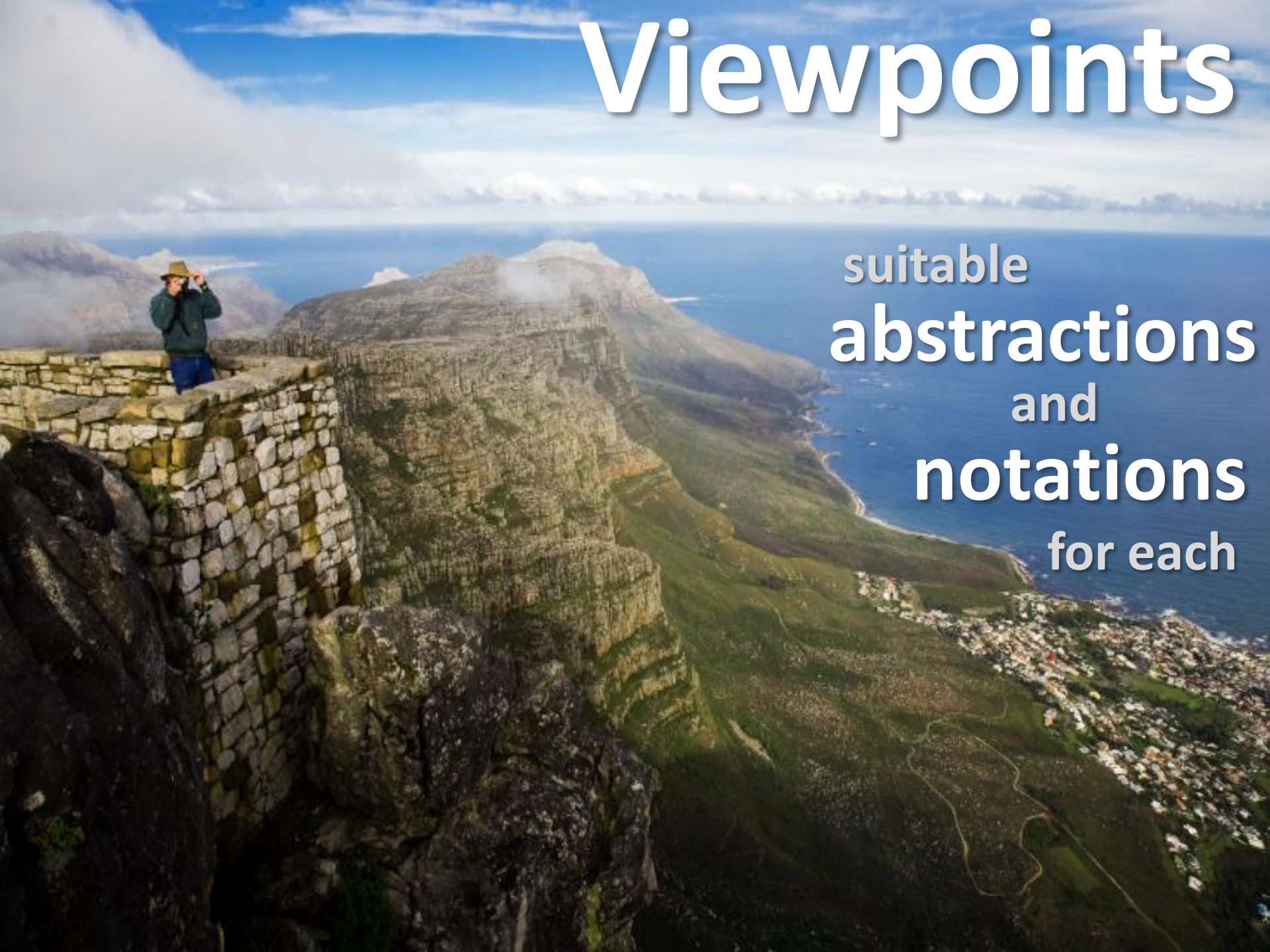
automated!

Solution Sp.

Domain :=

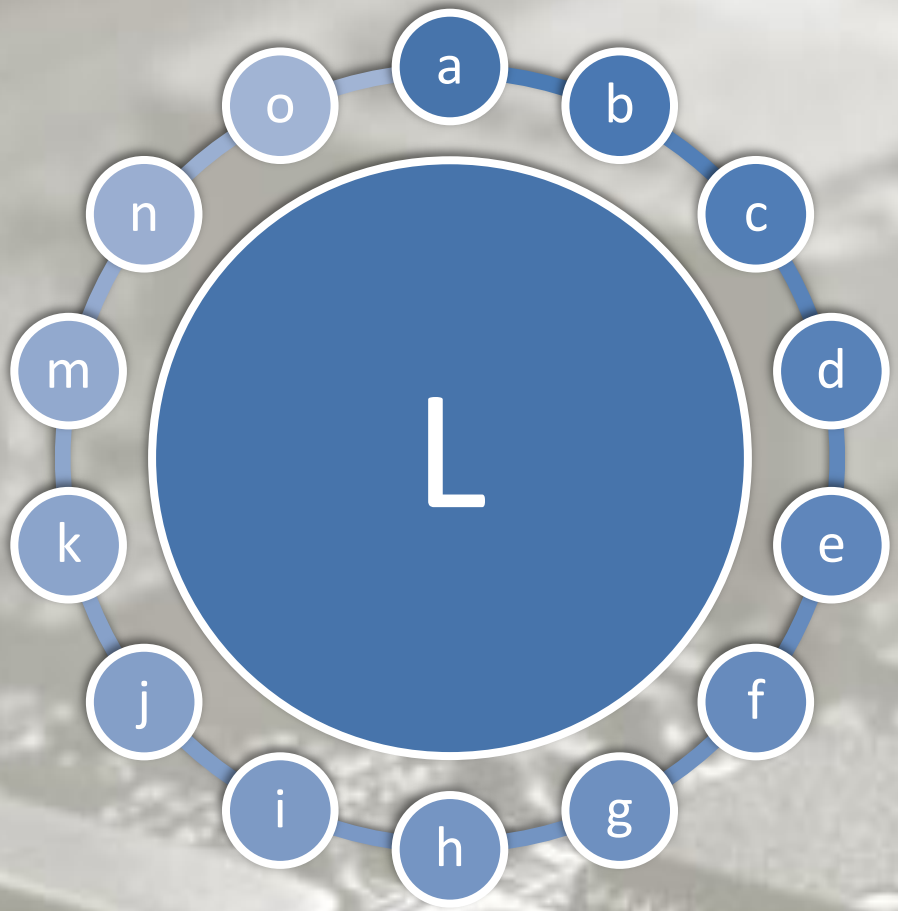
$\sum$ Viewpoints

Viewpoints


Viewpoints
suitable
abstractions
and
notations
for each

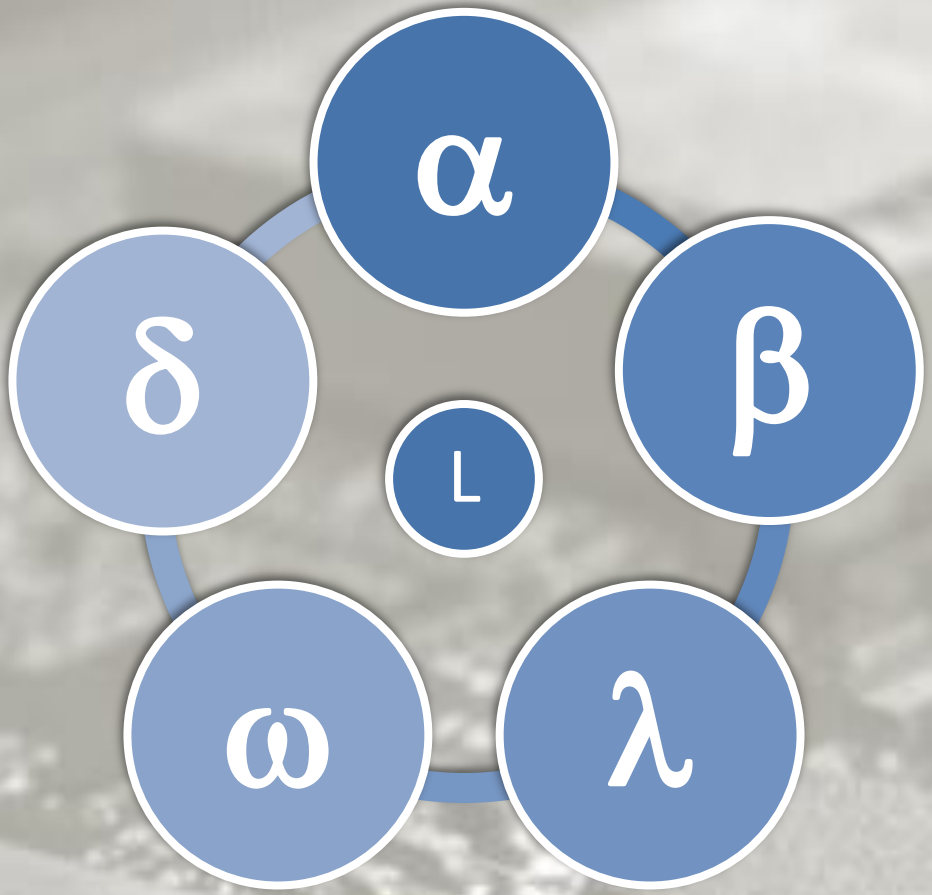
Viewpoints
Integrated
via
symbolic
references
and
seamless
transitions

Big Language?



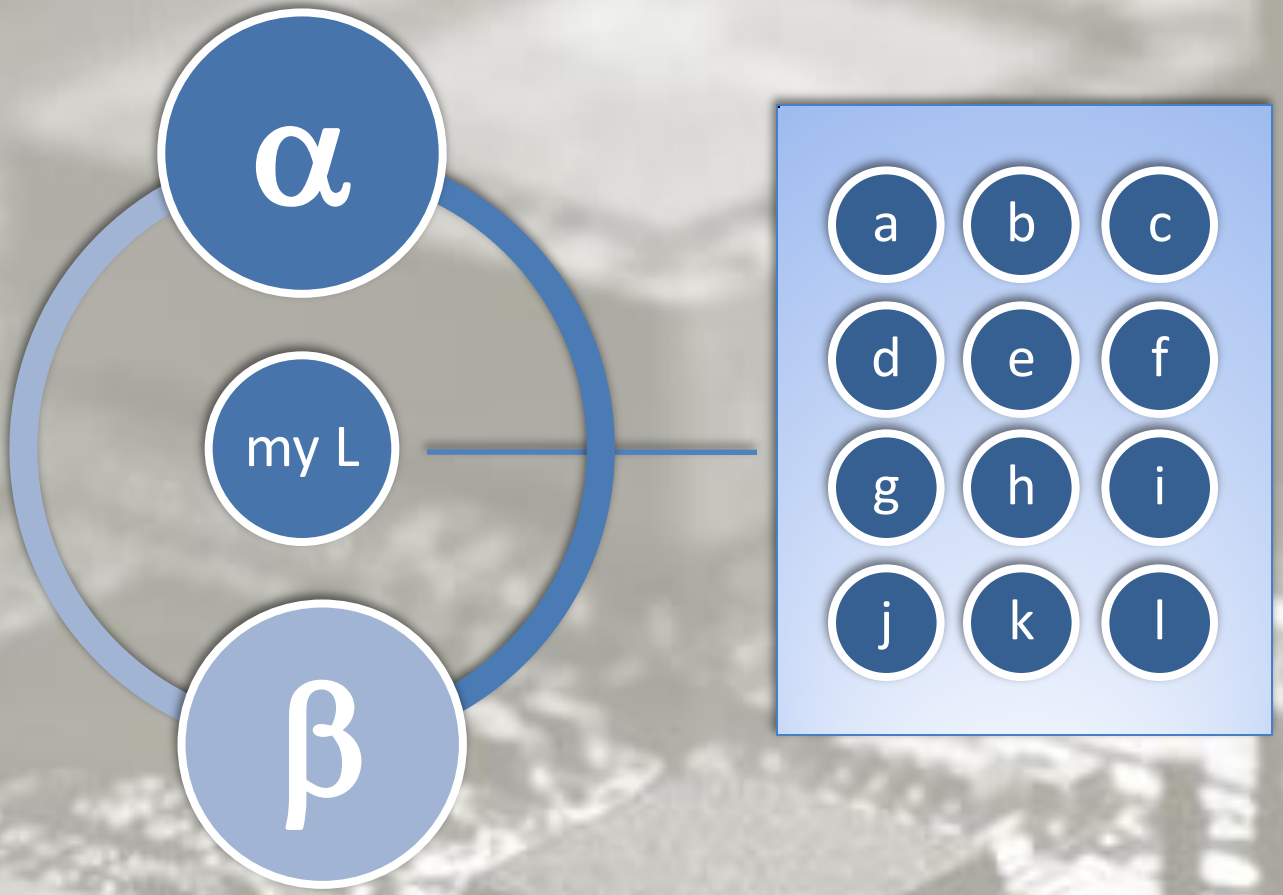
with many first class concepts!

Small Language?



with a few, orthogonal and powerful concepts

Modular Language



with many optional, composable concepts

Viewpoints

General Purpose

predefined library

configure

Viewpoints

Domain Specific

custom
purpose-built

create/include

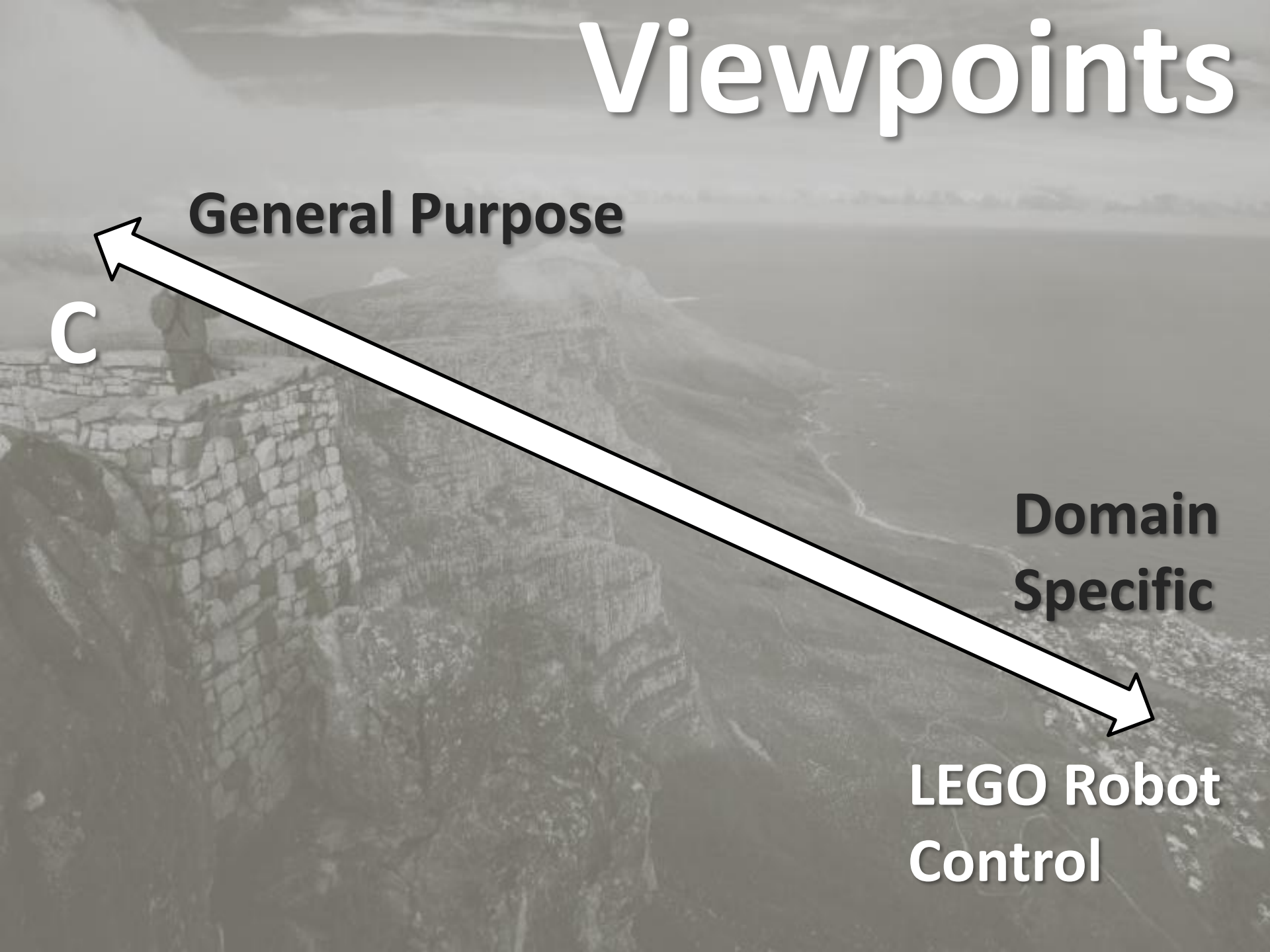
Viewpoints

Domain Specific

Custom Notations

real business expert integration

Viewpoints



Viewpoints

General Purpose

C

Components

State Machines

Sensor Access

Domain Specific

LEGO Robot Control

Modular Language for Emb-Sys

```
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);
}
```

Robot Control DSL in C Program

```
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

}
```

Robot DSL: Translation

```
module impl imports EcAPI {

  var int stateLocalTime  =  0;

  var int currentLeftDelta  =  0;

  var int currentRightDelta  =  0;

  var int currentLeftSpeed  =  0;

  var int currentRightSpeed  =  0;

  statemachine stopAndGoStateMachine {
    event timer;
    state step_0 {
      entry stateLocalTime = 0;
      entry currentLeftDelta = calcStep(currentLeftSpeed, 50, 3000);
            currentRightDelta = calcStep(currentRightSpeed, 50, 3000);
      timer [stateLocalTime > 3000] -> step_1
    }
    state step_1 {
      entry stateLocalTime = 0;
      entry currentLeftDelta = calcStep(
            currentRightDelta = calcStep
      timer [stateLocalTime > 3000] -> s
    }
    state step_2 {
      entry stateLocalTime = 0;
      entry currentLeftDelta = 0;
            currentRightDelta = 0;
            currentLeftSpeed = 0;
            currentRightSpeed = 0;
      <<transitions>>
    }
    initial state initial {
      timer [true] -> step_0
    }
  }
```

```
module impl imports <<imports>> {

  robot script stopAndGo
    accelerate to 50  within 3000
    decelerate to 0  within 3000
    stop

}
```

```
task stopAndGoTask cyclic prio = 1 every = 100 {
  stateLocalTime = stateLocalTime + 100;
  currentLeftSpeed = currentLeftSpeed + currentLeftDelta;
  currentRightSpeed = currentRightSpeed + currentRightDelta;
  nxt_motor_set_speed(MOTOR_PORT_T::NXT_PORT_B, currentLeftSpeed, 1)   (import EcAPI);
  nxt_motor_set_speed(MOTOR_PORT_T::NXT_PORT_C, currentRightSpeed, 1);
  event stopAndGoStateMachine:timer
  # TerminateTask(); // automatically added by platform.osek:addTermianateTask
}

int calcStep( int currSpeed, int newSpeed, int time ) {
  int ds = newSpeed - currSpeed;
  int steps = time / 100;
  return ds / steps;
}
```

Modular Language for Emb-Sys

```
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);
  }
}
```

Mathematics ...

```
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
```

Mathematics ... and Java Code

```
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>>
}
```

Configuration + Customization

Dummy Feature Model

```
feature runtimeCalibration : false
feature bumper : true
feature sonar : false
feature debugOutput : true
```

```
{sonar} task sonartask cyclic prio = 2 every = 100 {
    int s = ecrobot_get_sonar_sensor(SENSOR_PORT_T::NXT_PORT_S2);
    sonarHistory[sonarIndex] = s;
    sonarIndex = sonarIndex + 1;
    if ( sonarIndex == 10 ) {
      sonarIndex = 0;
    }
    int ss = 0;
    for ( int i = 0; i < 10; i = i + 1; ) {
      ss = ss + sonarHistory[i];
    }
    currentSonar = ss / 10;
    { {debugOutput} { debugInt(2, "sonar:", currentSonar); } }
  }
```

```
doc This is the cyclic task that is called every 1ms to do the actual control of the
task run cyclic prio = 2 every = 2 {
  stateswitch linefollower
    state running
      {bumper} int bump = ecrobot_get_touch_sensor(SENSOR_PORT_T::NXT_PORT_S3);
      {bumper} if ( bump == 1 ) {
        {debugOutput} { debugString(3, "bump:", "BUMP!"); }
        event linefollower:bumped
        terminate;
      }
      {sonar} if ( currentSonar < 150 ) {
        event linefollower:blocked
        terminate;
      }
      int light = ecrobot_get_light_sensor(SENSOR_PORT_T::NXT_PORT_S1);
      if ( light < ( WHITE + BLACK ) / 2 ) {
        updateMotorSettings(SLOW, FAST);
      } else {
        updateMotorSettings(FAST, SLOW);
      }
      {debugOutput} { debugInt(4, "light:", light); }
    {sonar} state paused
              updateMotorSettings(0, 0);
              if ( currentSonar < 255 ) {
                event linefollower:unblocked
              }
    {bumper} state crash
               updateMotorSettings(0, 0);
    default
      <noop>;
}
```

Configuring Models and Code

Dummy Feature Model

```
feature runtimeCalibration : false
feature bumper : true
feature sonar : false
feature debugOutput : true
```

```
{sonar} task sonartask cyclic prio = 2 every = 100 {
    int s = ecrobot_get_sonar_sensor(SENSOR_PORT_T::NXT_PORT_S2);
    sonarHistory[sonarIndex] = s;
    sonarIndex = sonarIndex + 1;
    if ( sonarIndex == 10 ) {
      sonarIndex = 0;
    }
    int ss = 0;
    for ( int i = 0; i < 10; i = i + 1; ) {
      ss = ss + sonarHistory[i];
    }
    currentSonar = ss / 10;
    { {debugOutput} { debugInt(2, "sonar:", currentSonar); } }
  }
```

```
doc This is the cyclic task that is called every 1ms to do the actual co
task run cyclic prio = 2 every = 2 {
  stateswitch linefollower
    state running
      if ( currentSonar < 150 ) {
        event linefollower:blocked
        terminate;
      }
      int light = ecrobot_get_light_sensor(SENSOR_PORT_T::NXT_PORT_S1);
      if ( light < ( WHITE + BLACK ) / 2 ) {
        updateMotorSettings(SLOW, FAST);
      } else {
        updateMotorSettings(FAST, SLOW);
      }
    state paused
      updateMotorSettings(0, 0);
      if ( currentSonar < 255 ) {
        event linefollower:unblocked
      }
    default
      <noop>;
}
```

Configuring Models and Code

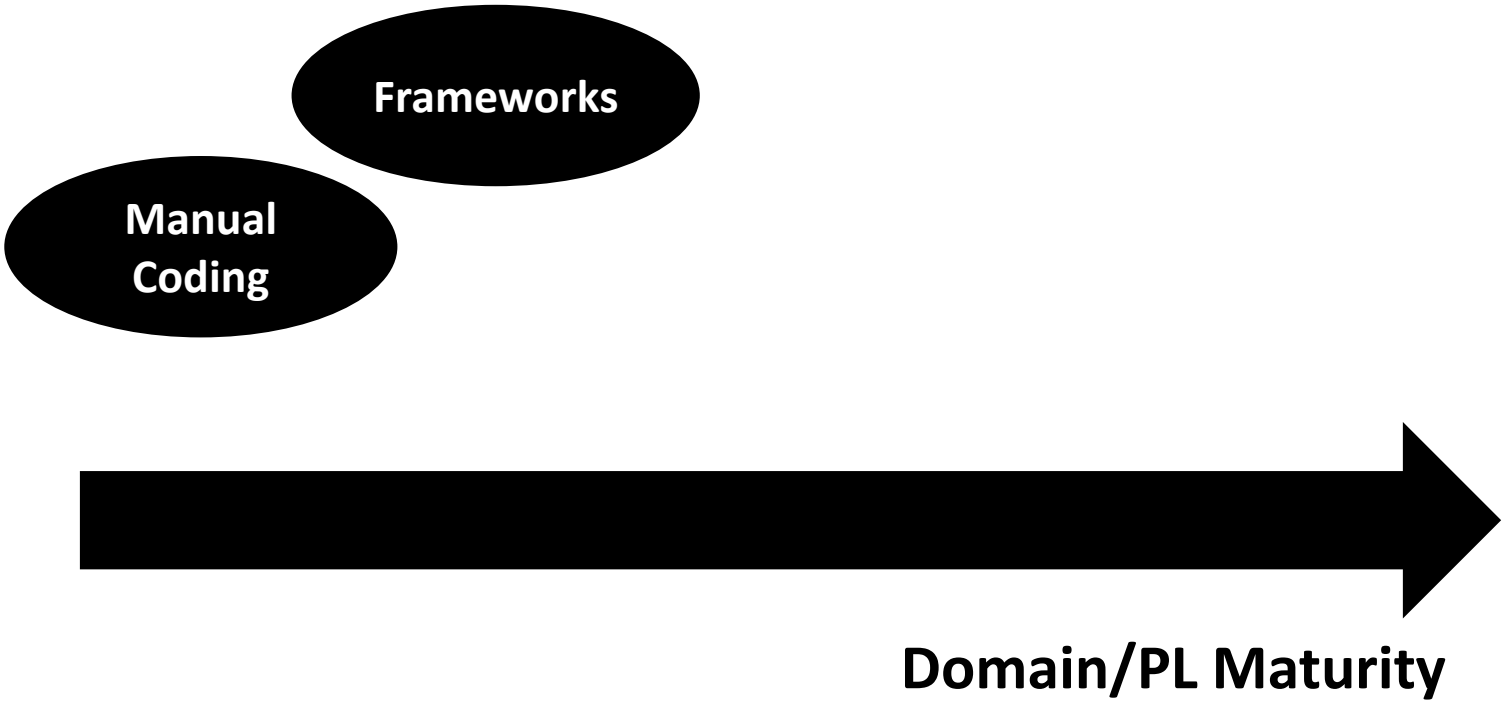
A DSL script in a feature model

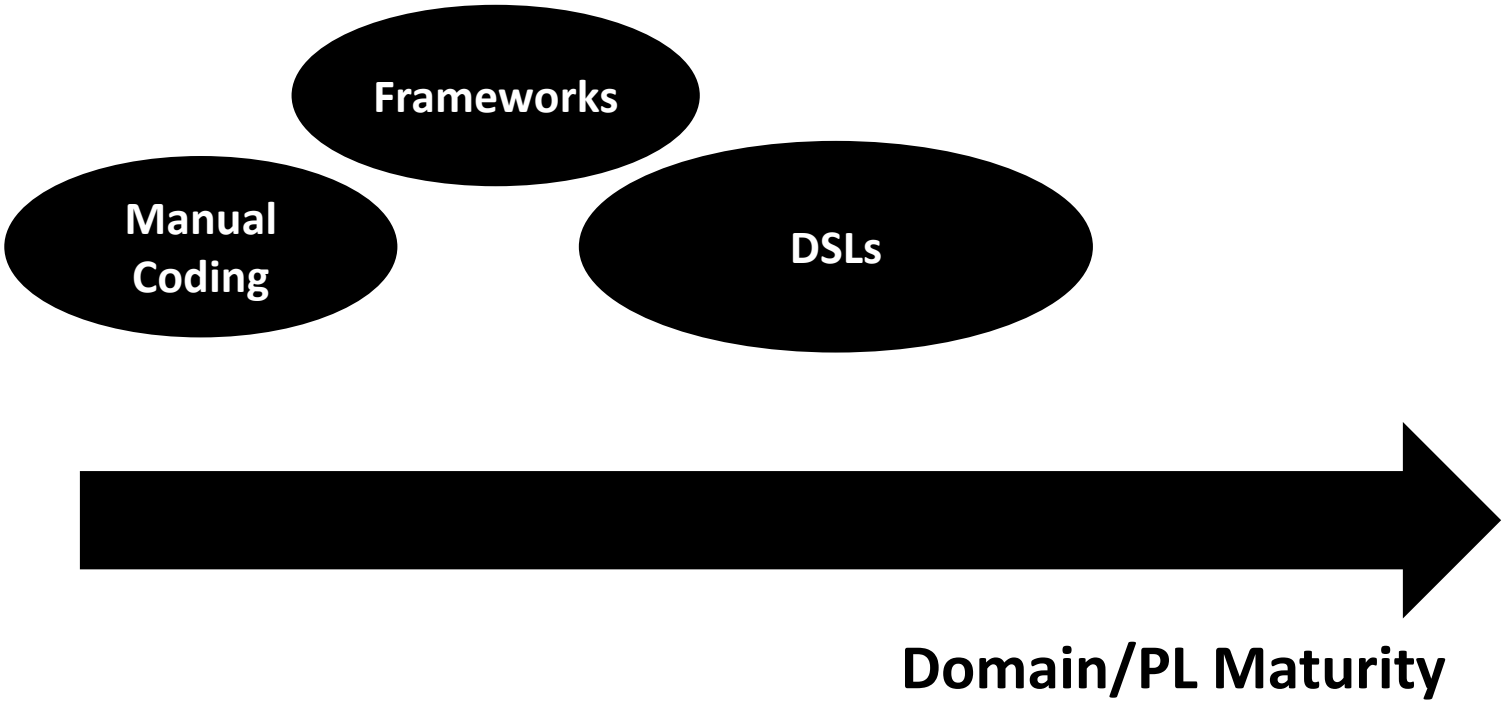
```
mandatory communication
  xor async
    or polling [ timeout : int
                 retry : custom retry script
                                 wait 10 ms
                                 try again
                                 give up ]
    or callback
    or oneway
      xor bestEffort [noOfTries : int]
      xor reliable
  xor sync
    optional exception
```

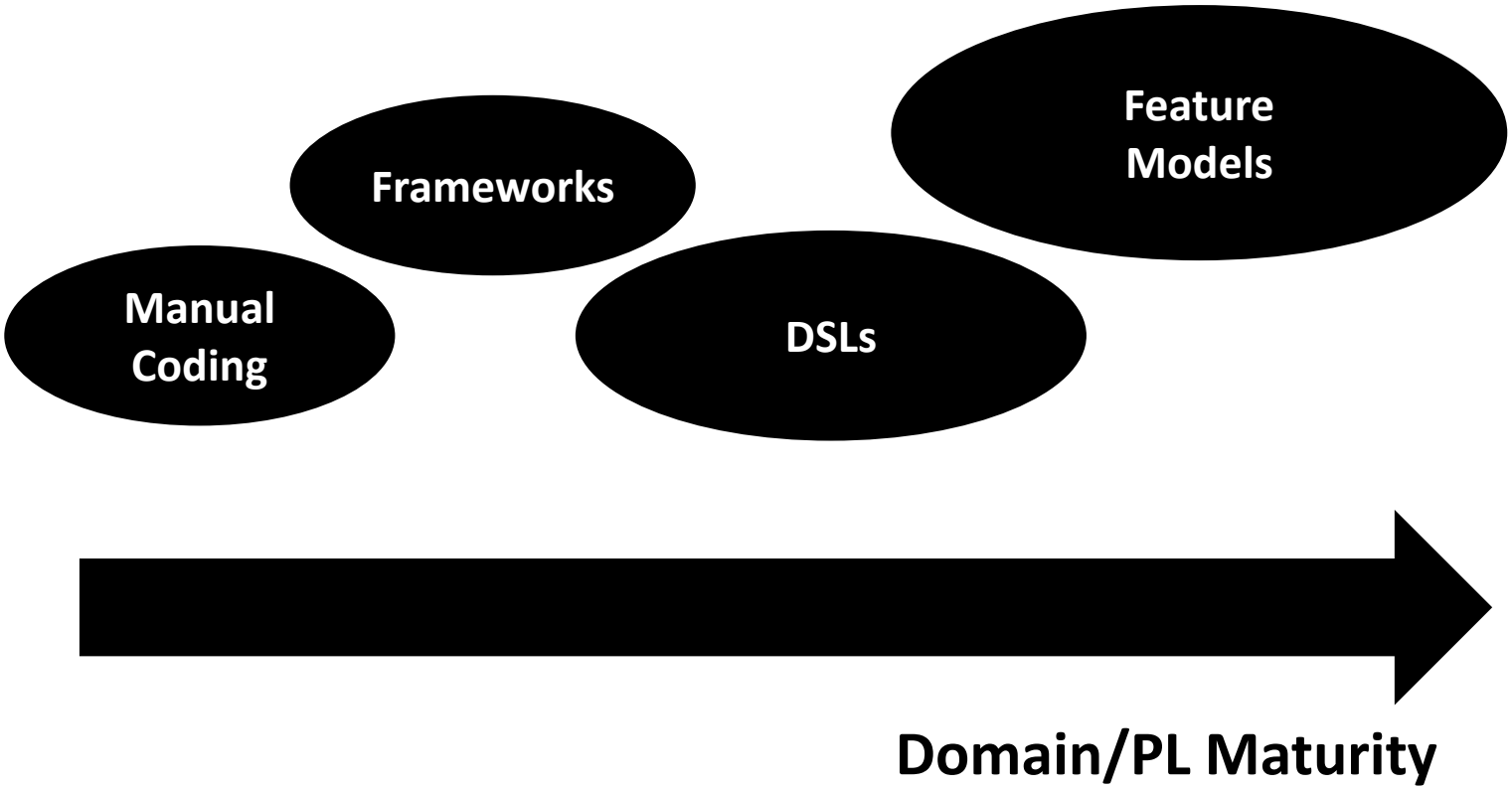
When to use Which?

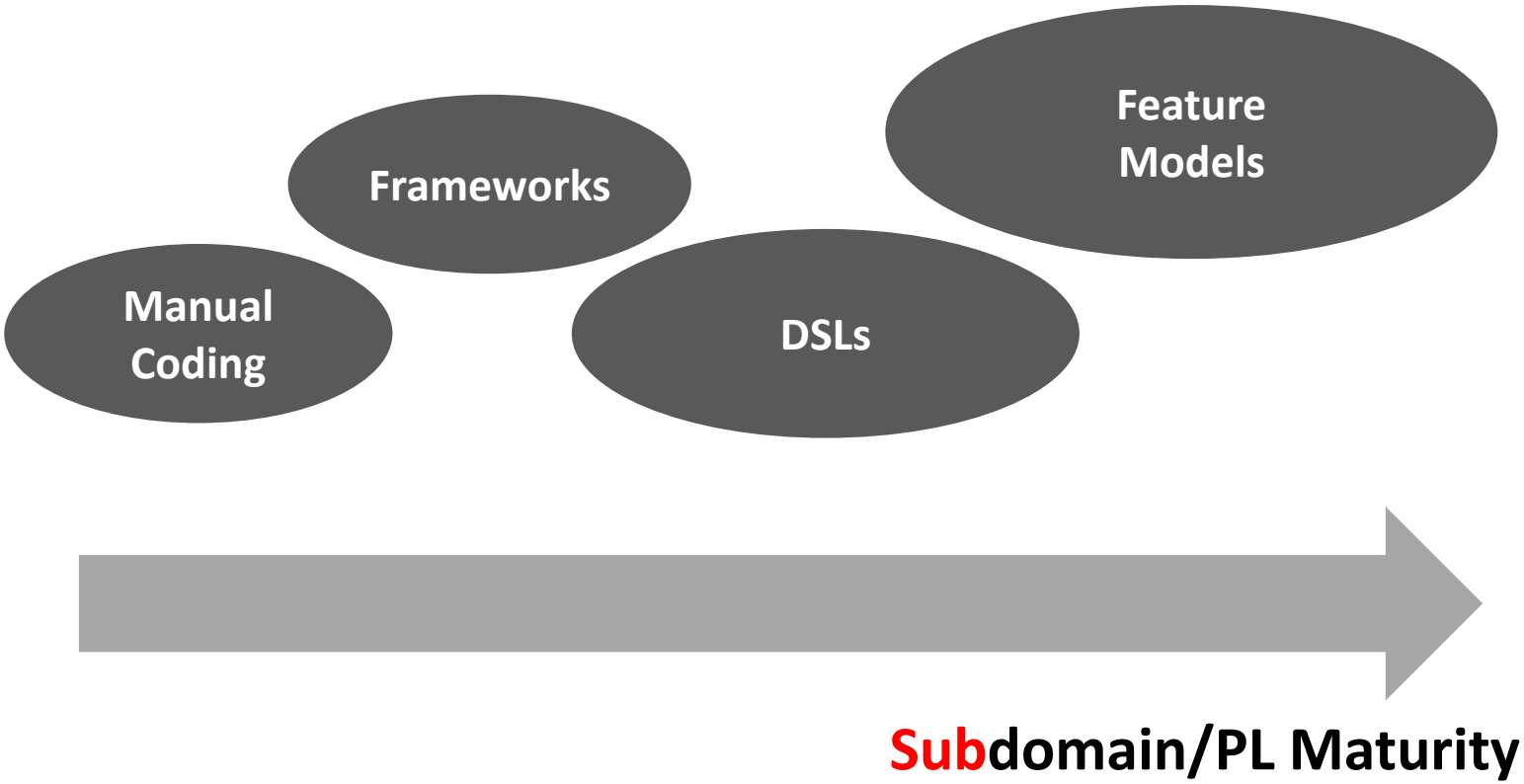
Domain/PL Maturity

Manual
Coding
Domain/PL Maturity


Frameworks
Manual Coding
Domain/PL Maturity


Frameworks
Manual Coding
DSLs
Domain/PL Maturity


Feature Models
Frameworks
Manual Coding
DSLs
Domain/PL Maturity


Feature Models
Frameworks
Manual Coding
DSLs
Subdomain/PL Maturity

DSL Tools

xtext

MPS
Meta Programming System

INTENTIONAL®
SOFTWARE

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®
SOFTWARE

Commercial Tool.
Projectional Editor
Very flexible notations
Version 1.8 is current

Way More:

Spoofax

Rascal

oomega

The Whole Platform

see also

http://languageworkbenches.net

THE END.

.coordinates

web www.voelter.de
email voelter@acm.org
skype schogglad
twitter markusvoelter

xing http://www.xing.com/profile/Markus_Voelter
linkedin http://www.linkedin.com/pub/0/377/a31

THE END.

.coordinates

web	www.voelter.de
email	voelter@acm.org
skype	schogglad
twitter	markusvoelter

xing	http://www.xing.com/profile/Markus_Voelter
linkedin	http://www.linkedin.com/pub/0/377/a31

THE END.

.coordinates

web	www.voelter.de
email	voelter@acm.org
skype	schogglad
twitter	markusvoelter

xing	http://www.xing.com/profile/Markus_Voelter
linkedin	http://www.linkedin.com/pub/0/377/a31

THE END.

.coordinates

web	www.voelter.de
email	voelter@acm.org
skype	schogglad
twitter	markusvoelter

xing	http://www.xing.com/profile/Markus_Voelter
linkedin	http://www.linkedin.com/pub/0/377/a31

THE END.

.coordinates

web	www.voelter.de
email	voelter@acm.org
skype	schogglad
twitter	markusvoelter

xing	http://www.xing.com/profile/Markus_Voelter
linkedin	http://www.linkedin.com/pub/0/377/a31