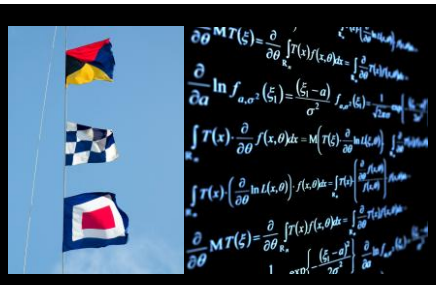


Domain Specific Languages

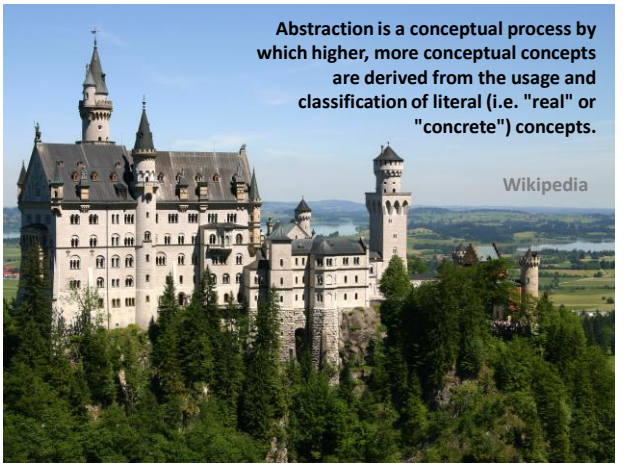
Examples, Tools, Practices

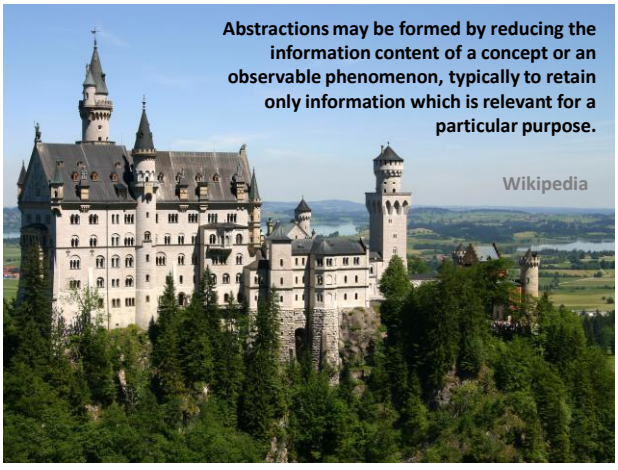
Markus Voelter
independent/itemis

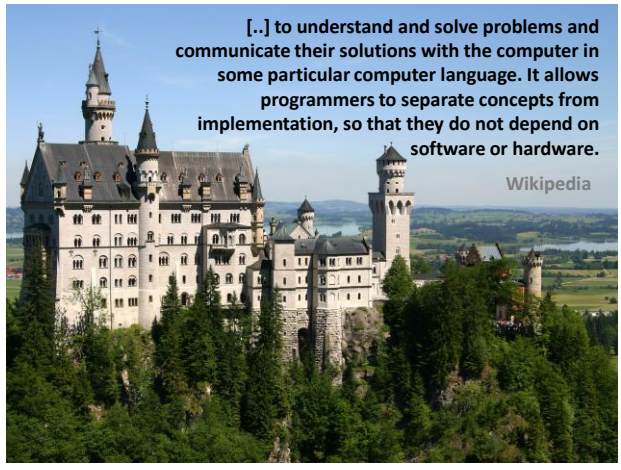
Domain Specific Languages

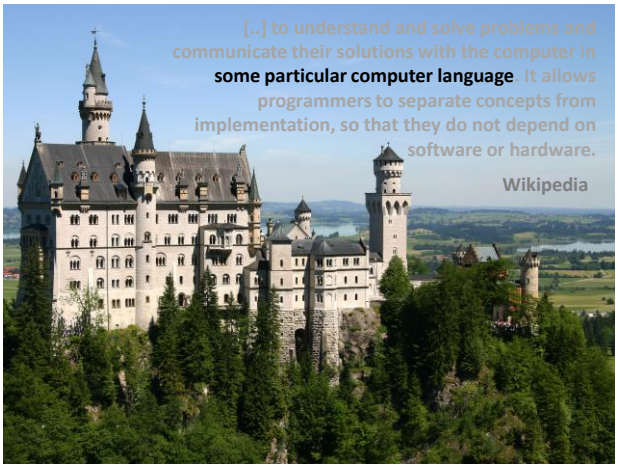
Examples, Tools, Practices

Introduction









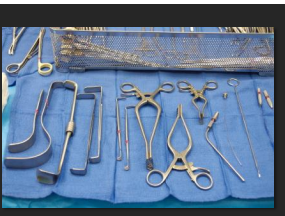
One Language to Rule Them All?

What is a good Language?

What is a good Language?
… for a given purpose?

general purpose

domain specific



tailor made

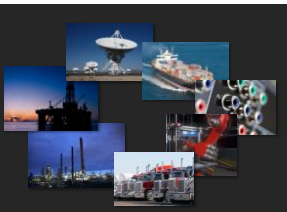
effective++

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.



execute?

map

DSL Program
(aka Model)

map

automated!

GPL Program

map

Generation
Transformation
Compilation

Interpretation

LanguageOrientedProgramming
DomainSpecificModeling
GenerativeProgramming
ModelDrivenDevelopment
ModelIntegratedComputing
ModelBasedDevelopment
ModelDrivenEngineering
ModelBasedEngineering
ModelDrivenArchitecture
ModelDrivenSoftwareDevelopment

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

customization
vs.
configuration

graphical
vs.
textual

Domain Specific Languages

Examples, Tools, Practices

Examples

Example 1:

Embedded Protocol Handler

Component Specification

```
processing DigitalIn "BI" moduletype 0x08 hal = DigitalInHAL  {

    datatypes {
        SinglePointIndicationWithoutTime;
        SinglePointIndicationWithTime;
        DoublePointIndicationWithoutTime;
        DoublePointIndicationWithTime;
        BitStringType18BitWithoutTime;
        BitStringType18BitWithTime;
    }

    parametertypes {
        DataType default {
            subattr db0 # intendedDataType == pdt SinglePointIndicationWithTime;
        };
        DebounceFilterTime default {
            attr filterTimeInMs == 0x02;
            subattr db1 # SP == 0x00;
            subattr db1 # IN == 0x00;
        } ;
        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 writeRegisterNumber2 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 writeRegisterNumber2 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];
    } ;
}
```

tests

```
test writeRegisterNumber2 for dip writeRegisterNumber2 {
    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;
    };
}
```

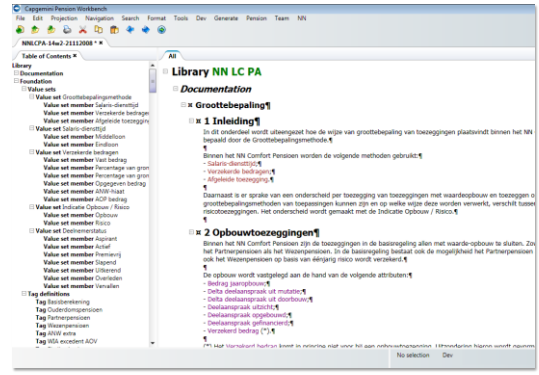
refines

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

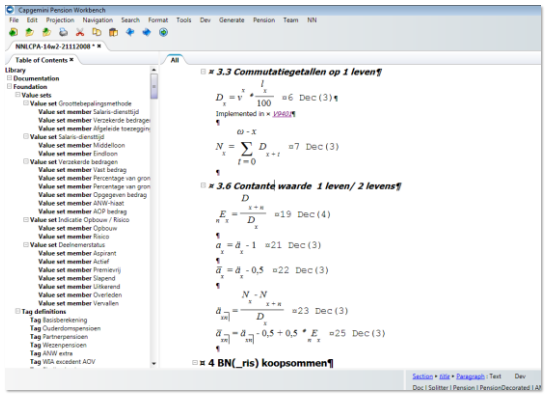
Example 2:

Pension Fund Specification

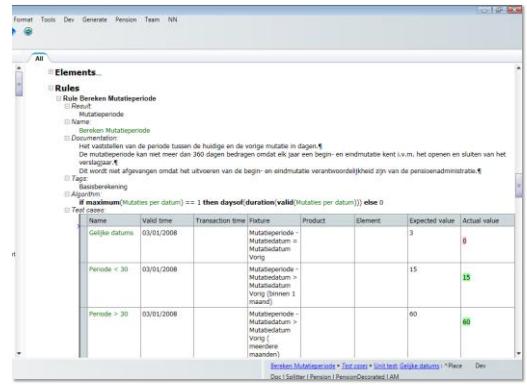
Textual Documentation



Insurance Mathematics



Calculation Rules and Tests



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 "Standardmenue"
        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
}
```

Example 5:
OSGi-based System

Component Specification

```
subsystem the.world.scenarios {

    public:

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

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

        interface Press {
            broadcast( m: 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 onceUponS
            requires home: Radio [0..1]
        }

}
```

Test Case Modeling

```
subsystem the.world.scenarios {

    public:

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

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

        interface Press {
            broadcast( m: 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 onceUponS
            requires home: Radio [0..1]
        }

}
```

```
import "classpath:/twoworlds.compds1"

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"
    }

}
```

647x651

Deploymnet

```
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 onceUponS
            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
    }
}
```

Example 8:

Miscellaneous

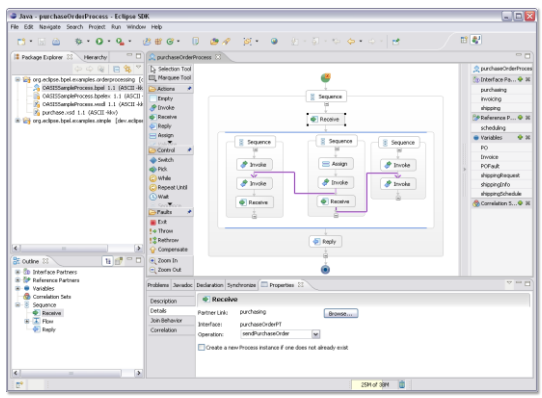
SOLUTION

A DSL Hearing Aid Configuration

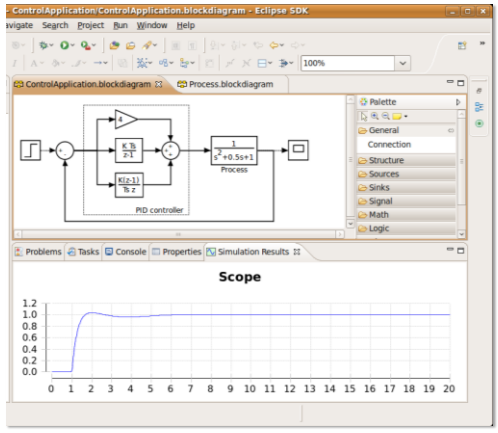
SOLUTION

A DSL for Refridgerator Configuration

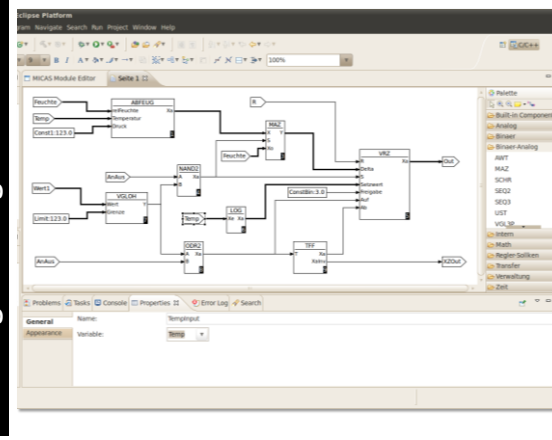
BPEL Designer



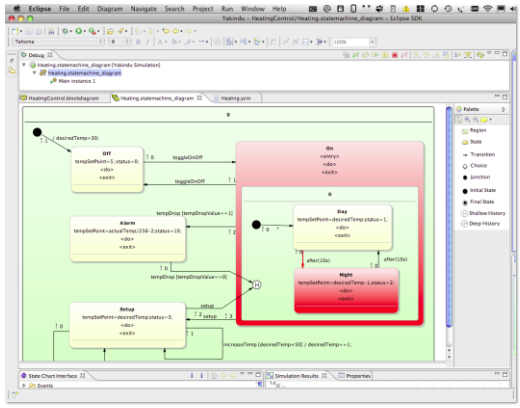
Block Diagrams



PLC Programming



State Charts



Domain Specific Languages

Examples, Tools, Practices

Concepts & Terms

Metamodel
(Abstract Syntax)

Metamodel
(Abstract Syntax)
Concrete Syntax
(Notation)

Metamodel
(Abstract Syntax)
Concrete Syntax
(Notation)
Static Semantics
(Constraints)

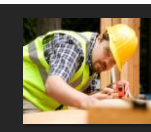
Metamodel
(Abstract Syntax)
Concrete Syntax
(Notation)
Static Semantics
(Constraints)
Dynamic Semantics

Metamodel
(Abstract Syntax)
Concrete Syntax
(Notation)
Static Semantics
(Constraints)
Dynamic Semantics
Model vs. Program

Domain Specific Languages
Examples, Tools, Practices

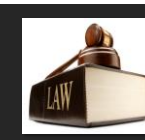
Benefits

Automation
faster, deterministic

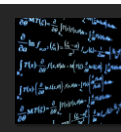


Increased Quality
well defined structures allthrough the system

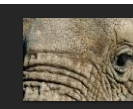
Meaningful Validation
more semantics in the model



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

Domain Specific Languages

Examples, Tools, Practices

Tooling


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

Domain Specific Languages

Examples, Tools, Practices

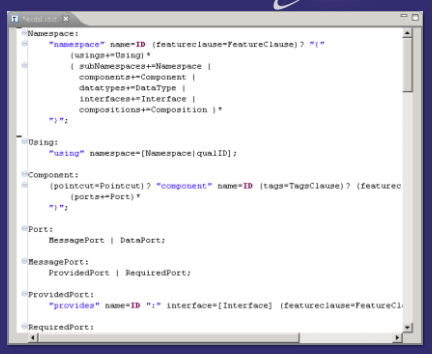
Tooling: Eclipse Xtext

eclipse modeling PROJECT

Xtext

eclipse modeling PROJECT

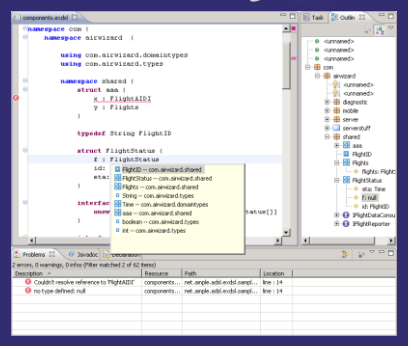
Building Textual Editors



Xtext

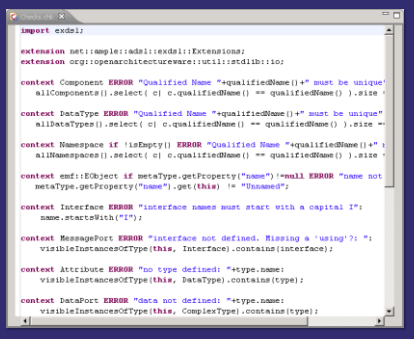
eclipse modeling PROJECT

Building Textual Editors

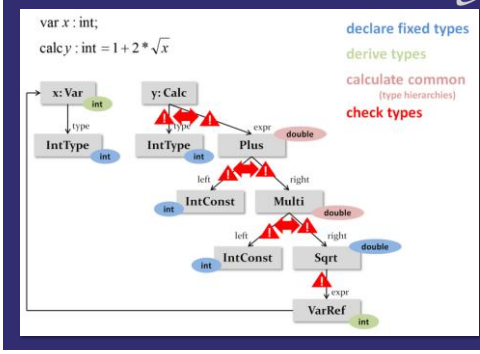


Constraints

expressed
in **Java**
-or-
with **OCL**
and dialects



Xtext Typesystem F/W

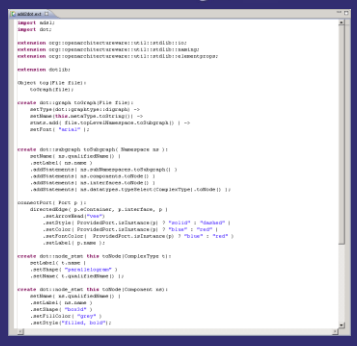


Typing
Rules
For
Expr's

M2M

Model-to-Model Transformations

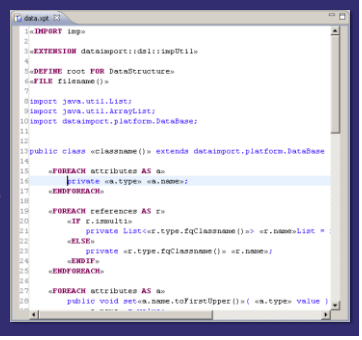
- INRIA's ATL
- QVT
- Xtend



M2T

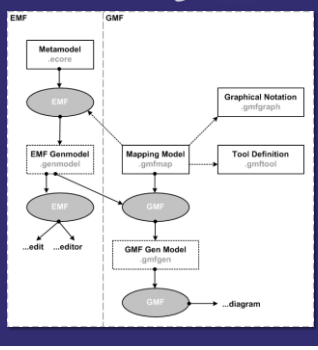
Model-to-Text Transformations

- JET: Java Emitter Templates
- Xpand: oAW's template engine



GMF

Graphical Box/Line editors based on EMF



EMF

Ecore meta meta model
+
Editing
Transactions
Validation
Query
Distribution/Persistence

DEMO

Building DSLs with
Eclipse Xtext

Domain Specific Languages

Examples, Tools, Practices

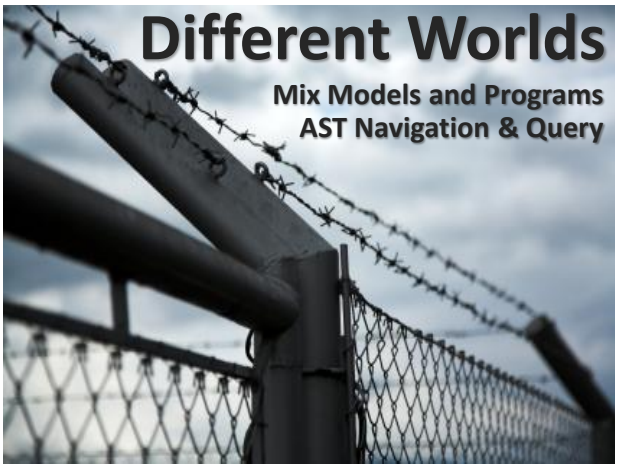
Programming vs. Modeling

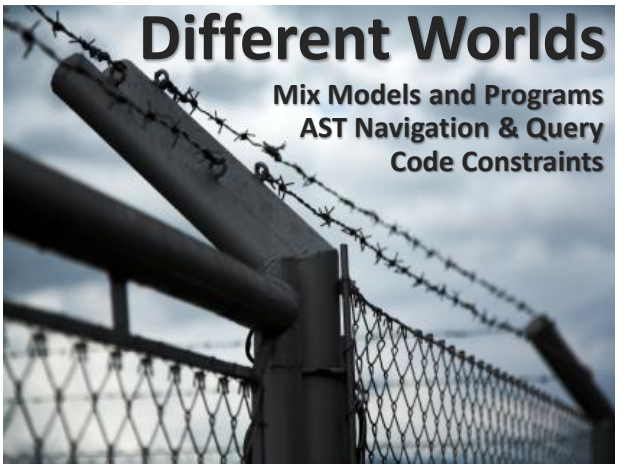
programming?

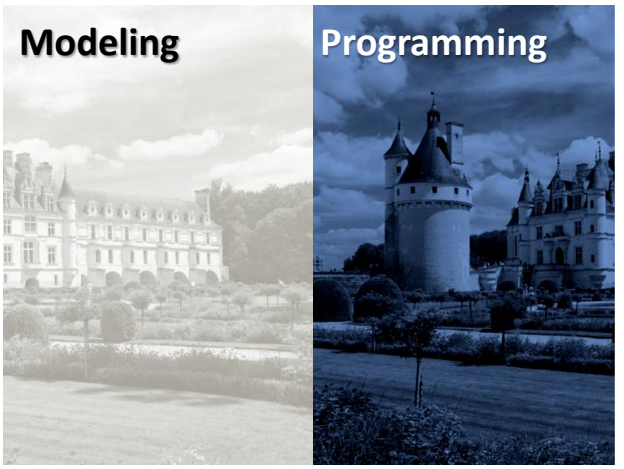
modelling?

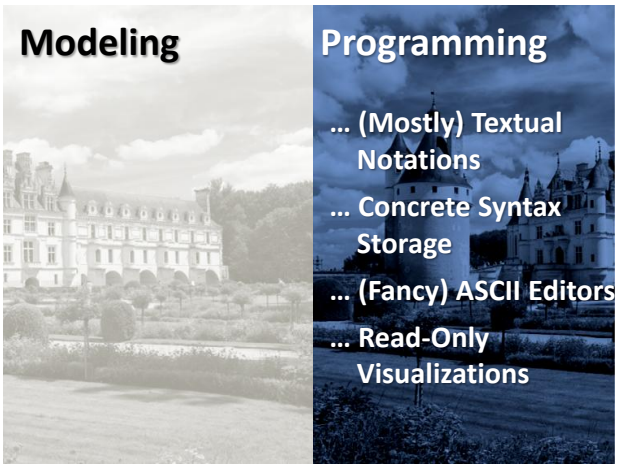
programming?
vs.
modelling?

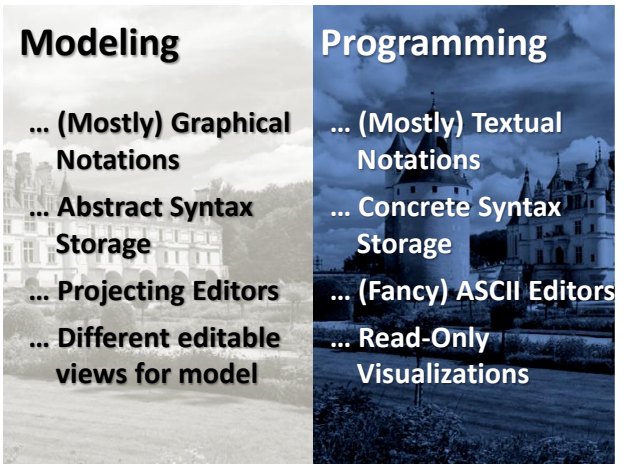
different?
the same?

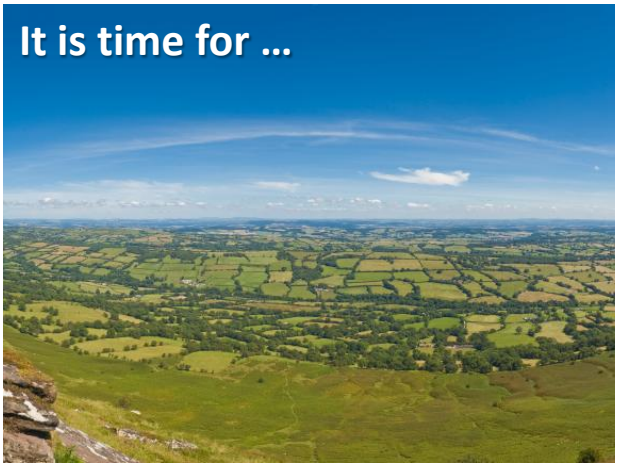

Different Worlds
Mix Models and Programs
AST Navigation & Query


Different Worlds
Mix Models and Programs
AST Navigation & Query
Code Constraints


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



We don't want to
model,
we want to
program!

We don't want to
model,
we want to
program!

… at different levels of abstaction
… 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

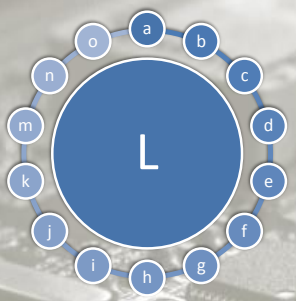
We don't want to
model,
we want to
program!

And always:
precise and tool processable

Domain Specific Languages
Examples, Tools, Practices

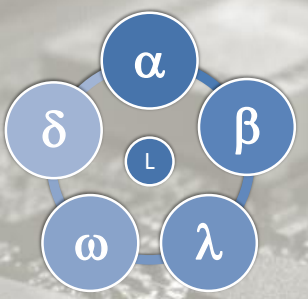
Modular Languages

Big Language?



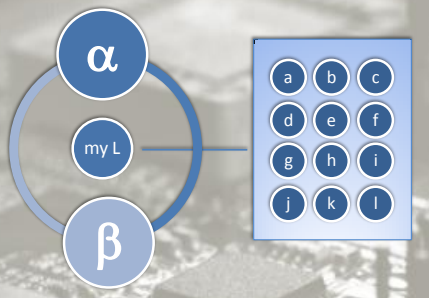
with many first class concepts!

Small Language?



with a few, orthogonal
and poweful concepts

Modular Language



with many optional, composable concepts

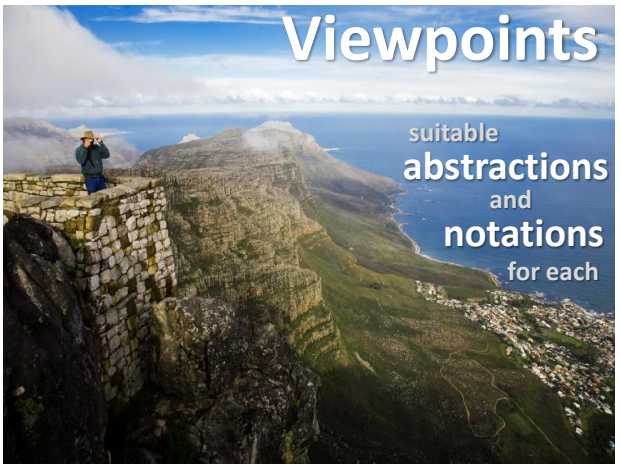
Modular Language

Like frameworks and libraries,

Modular Language

Like frameworks and libraries,

but with syntax and IDE support



Viewpoints
General Purpose

predefined library

configure

Viewpoints
Domain Specific

custom purpose-built

create/include

Viewpoints
Domain Specific

Custom Notations

real business expert integration

Viewpoints

General Purpose

C

Domain Specific

LEGO Robot Control

Viewpoints

General Purpose

C

Components

State Machines

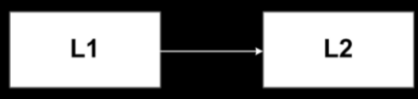
Sensor Access

Domain Specific

LEGO Robot Control

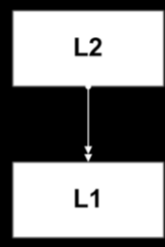
Language Integration by…

Referencing



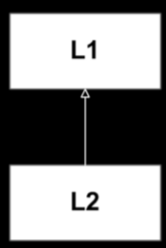
Language Integration by…

Cascading



Language Integration by…

Extension

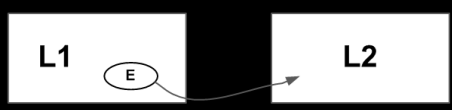


Language Integration by…

Embedding

Language Integration by…

Contributing




Not a new idea…


Growing A Language
(Guy L Steele)


Growing A Language
(Guy L Steele)
stoneage-OOPSLA talk,
get it on Google Video!


Language
Workbench
(Martin Fowler)


Language
Workbench
(Martin Fowler)
Freely
define
languages and
integrate
them

Language Workbench
(Martin Fowler)

use
persistent
abstract
representation

Language Workbench
(Martin Fowler)

language ::=
schema
+ editors
+ generators

Language Workbench
(Martin Fowler)

projectional
editing

Language Workbench
(Martin Fowler)

persist
incomplete
or
contradictory
information

Language Workbench
(Martin Fowler)

powerful
editing
testing
refactoring
debugging
groupware

language definition
implies
IDE definition

Language Workbench
(Martin Fowler)

support for
„classical“
programming
„classical“ and
modeling

Not a new idea…

Not a new idea…

But NOW! a practical reality :-)

Domain Specific Languages

Examples, Tools, Practices

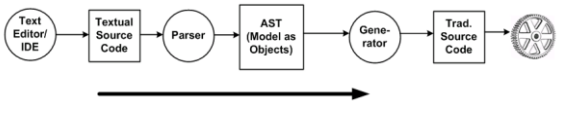
Projectional Editing

Parser-based

text

… to tree

… to text



Projectional

tree

… to text-lookalike (editor)

… to other trees … [*]

… to text

Programming as Modeling

- … (Mostly) Graphical Notations
- … Abstract Syntax Storage
- … Projecting Editors
- … Different editable views for model

Programming as Modeling

- … ~~(Mostly) Graphical~~ Any kind of Notations
- … Abstract Syntax Storage
- … Projecting Editors
- … Different editable views for model

Language Composition

There's no parsing.

Unique Language Element Identity.

Unlimted language composition.

Flexible Notations

Textual
like ASCII

Graphical
box & line

Semi-Graphical
mathematical

} treated the same
can be mixed

Automatic IDE Extension

tool support is inherent
for languages build with
projectional tools

language definition
implies
IDE definition

Multiple Notations

… for the same concepts

e.g. in different contexts
or for different tasks

Partial
Projections

... different views
... for different roles/people
... only a particular variant

Storage
!= Schema

... store arbitraty meta data
- change log
- conflicting information
- variability annotations

... independent of language schema!
... „aspects“, overlay

Live
Programs

think: spreadsheet

a change to one part of program can lead to (dependent) changes in other parts

Tree Editing

... is different from editing text

... try to make it feel like text

... takes some getting used to

but: for more flexible notations a more general editing paradigm is needed

Infrastructure
Integration

... storage is not text

... diff/merge must be in tool

... existing text tools don‘t work

Proprietary
Tools

... no standards

... no interop

[redacted]

Domain Specific Languages

Examples, Tools, Practices

Tooling: MPS

also do...

IntelliJ IDEA
Resharper

released in

Q3 2009

currently

1.5

April 2011

2.0

Open Source

licensed under

Apache 2.0

& me

xtext

INTENTIONAL SOFTWARE

MetaEdit+

& JetBRAINS

& itemis

Build new standalone DSLs

Build new standalone DSLs
Build DSLs that reuse parts of other languages

Build new standalone DSLs
Build DSLs that reuse parts of other languages

(MPS comes with BaseLanguage) Java++

extend base language

Build new standalone DSLs
Build DSLs that reuse parts of other languages

(MPS comes with BaseLanguage) Java++

extend base language
build DSLs that reuse parts of BaseLanguage

Domain Specific Languages

Examples, Tools, Practices

Demo Scenario

C is good

- fast
- flexible
- wide-spread
- multi-platform

C is bad

- limited abstractions
- not domain-specific
- „dangerous“ preprocessor
- not extensible

Modeling

compensates for C‘s problems

Modeling

compensates for C‘s problems, specifically

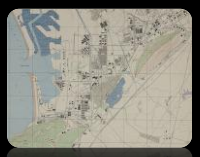
- good abstractions
- domain-specific
- extensible

Modeling

compensates for C‘s problems, but:

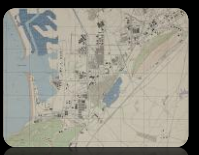
- not extensible
- tool/tool integration
- tool/code integration
- process integration

Two Classes




(yes, this is a slight simplification)

Plan Oriented
Top Down
Big Systems
Big Companies

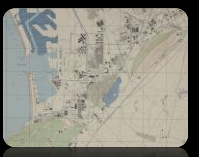


Modeling

Grown
Bottom Up
Small/Medium Systems
Smaller Companies

C Code

Two Classes




… and good luck with mixing the two!

mbeddr.com

Modular Embedded Language

Incremental Extension of C

mbeddr
0110101101011
rethinking embedded software development

Incremental Extension of C

- Components
- Tasks
- State Machines
- Physical Units
- Special Data Types

mbeddr
0110101101011
rethinking embedded software development

Incremental Extension of C

syntactically and semantically integrated

mbeddr
0110101101011
rethinking embedded software development

Incremental Extension of C

extensible with domain-specific constructs (DSLs)

SOLUTION

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

  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] -> runnin
    }
    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 contr
task run cyclic prio = 2 every = 2 {
  statemachine 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);
}
```

SOLUTION

```
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);
  }
}
```

SOLUTION

```
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

}
```

SOLUTION

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
OptionalOutputlkjlkjlkj	Display output should be optional
TwoPhases	Initialization should be separate from operation
ConsistentSetting	Motor settings have to be updated consistently

SOLUTION

Program Code with Annotations (green)

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>;
}
```

SOLUTION

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
  Calibration: The black/white values should                  *requirements ([fmain.null]
  ConsistentSetting: Motor settings have to be upda           *requirements ([fmain.null]
  Cyclic: the actual control of the devi                      *requirements ([fmain.null]
  Efficient: The program should be as small                   *requirements ([fmain.null]
  Init: The system should start operat                        *requirements ([fmain.null]
  MaxSpeed: Speeds per motor can only be u                    *requirements ([fmain.null]
  OptionalOutputlkjlkjlkj: Display output should be optio     *requirements ([fmain.null]
  TwoPhases: Initialization should be separ                   *requirements ([fmain.null]
}
trace Init
int32 light = 0;
```

Find Usages of Requirements

Usages: DummyRequirement

- Searched nodes
  - DummyRequirementsCollection
    - DummyRequirement(role: requirements; in: DummyRequirementsCollection)
- 4 usages found
  - LineFollower (4)
    - lfmain (4)
      - LineFollower (4)
        - DummyRequirement(role: links; in: RequirementsTrace)
        - DummyRequirement(role: links; in: RequirementsTrace)
        - DummyRequirement(role: links; in: RequirementsTrace)
        - DummyRequirement(role: links; in: RequirementsTrace)

SOLUTION

```
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
```

SOLUTION

```
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
```

SOLUTION

```
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>>
}
```

DEMO

Extending Languages
With MPS

Domain Specific Languages

Examples, Tools, Practices

Tool: Spoofax

Traditionally:

Language ::=
Token* | Rule*;

Traditionally:

Language ::=
Token* | Rule*;

Token ::=
Character*;

Traditionally:

Language ::=
Token* | Rule*;

Token ::=
Character*;

Rule ::=
Token* | Rule;

Traditionally:

Language ::=
Token* | Rule*;

2 Phases:

1. Global Tokenization (aka Scanning)
2. Grammar Recognition (aka Parsing)

What if!

combine languages

same seq<Char> as different tokens?

Ambiguous!

Not Parsable.

Bad Luck.

Modern:

Language ::=
Rule*;

Rule ::=
Char* | Rule*;

1 Phase:
Grammar Recognition
directly on char*

Scannerless Parsing

No Ambiguity
when combining languages

Always Parsable.
(well, almost. There are some corner cases...)

Custom Syntax

Graphical
Textual
Symbolic++

IDE Support

Teamwork
Debugging
Custom Editors

Complete Symbolic Integration

Goto Def
Find Refs
Refactoring

Limited to Unicode

how to handle
non-character symbols

Graphics != Text

two worlds…

separate editors

- … per syntax/viewpoint
- … models can still be ref integrated

TU Delft Delft University of Technology

Eelco Visser's group

SDF: Declarative Syntax Definition

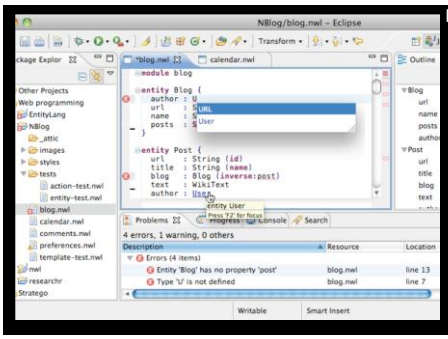
TU Delft Delft University of Technology

Eelco Visser's group

Stratego: Rewriting Language

TU Delft Delft University of Technology

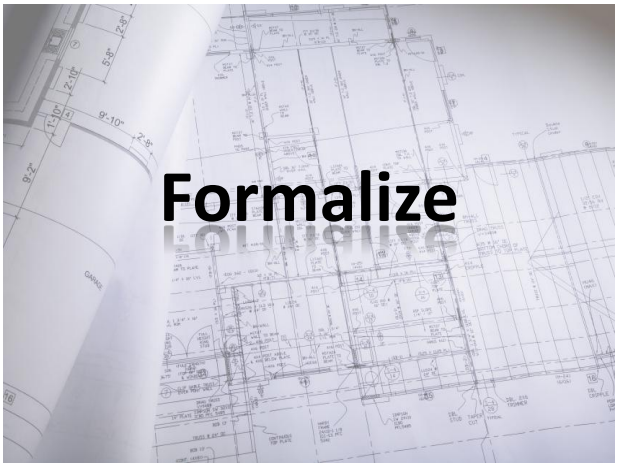
Eelco Visser's group

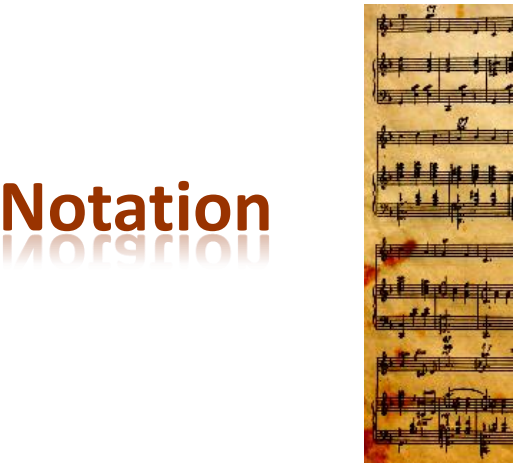


Domain Specific Languages

Examples, Tools, Practices

Recap: What did we do?





Isolate Technology

Automate

$\frac{\partial}{\partial\theta}\mathrm{MT}(\xi)=\frac{\partial}{\partial\xi}$

$\frac{\partial}{\partial a}\ln f_{a,\sigma^2}(\xi_1$

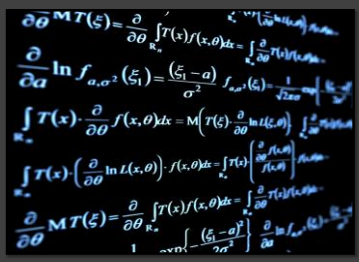
Domain Specific Languages

Examples, Tools, Practices

A few Best Practices

Limit Expressiveness

Notation, Notation, Notation



Graphical vs. Textual

DSL Semantics

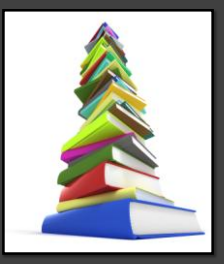
Viewpoints

Partitioning

Evolution

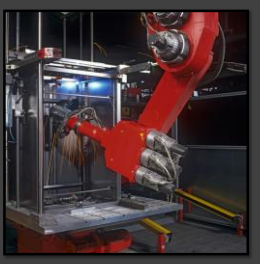
The Fallacy of Generic Languages

Learn from 3GLs



Tooling Matters!

Interpretation vs. Generation



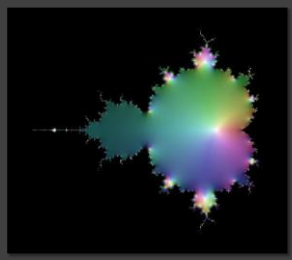
Rich Domain Specific Platform

Integrating Generated and Manually Written Code

Model-2-Model
To Simplify Generators

Don‘t forget
Testing

Iterate!



Documentation
is still necessary

Domain Users
Programming?


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


.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