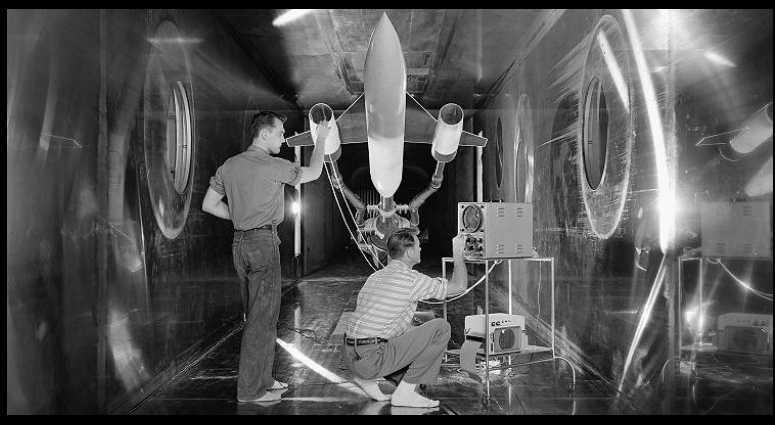
DSLs, Models and Architecture

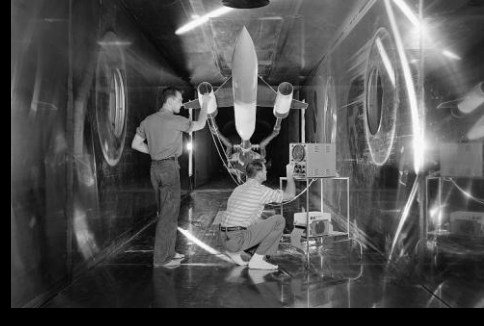


itemis

Markus Voelter
www.voelter.de
voelter@acm.org

- What is Architecture?
- What is Architecting?
- Domain Specific Languages
- Examples
- What did we do?
- Why Textual?
- DSL Tools
- Validation
- Generating Code
- Analysis Tools
- Standards and UML
- Documentation
- Summary

What is Architecture?



… the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships between them.

Wikipedia

… the set of design decisions which, if made incorrectly, may cause your project to be cancelled.

Eoin Woods

… its "style and method of design and construction".

Hayes-Roth

… comprises

- A collection of software and system components, connections, and constraints.
- A collection of system stakeholders' need statements.
- A rationale which demonstrates that the components, connections, and constraints define a system that, if implemented, would satisfy the collection of system stakeholders' need statements.

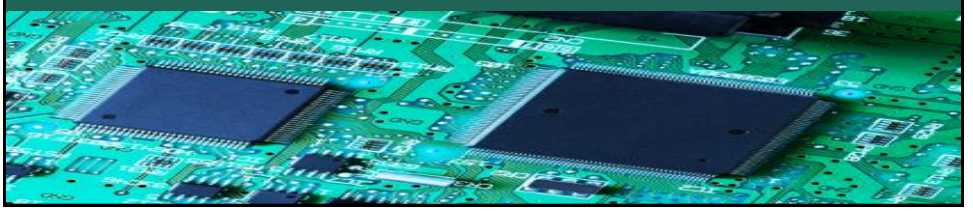
Barry Boehm

… everything that needs to be consistent throughout a software system

My Own

Modularize

Procedures, Classes,
Components, Services



Encapsulate

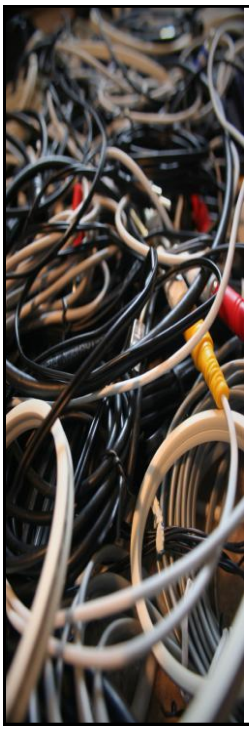
Private Members
Frameworks
Facade Pattern
Components
Layers/Rings/Levels
Packed Data Wrapper

Contracts

Interfaces
Pre/Post Conditions
Protocol State Machines
Message Exchange Patterns
Published APIs

Decoupling

Message Queues
Deferred Consistency
Compensating TX



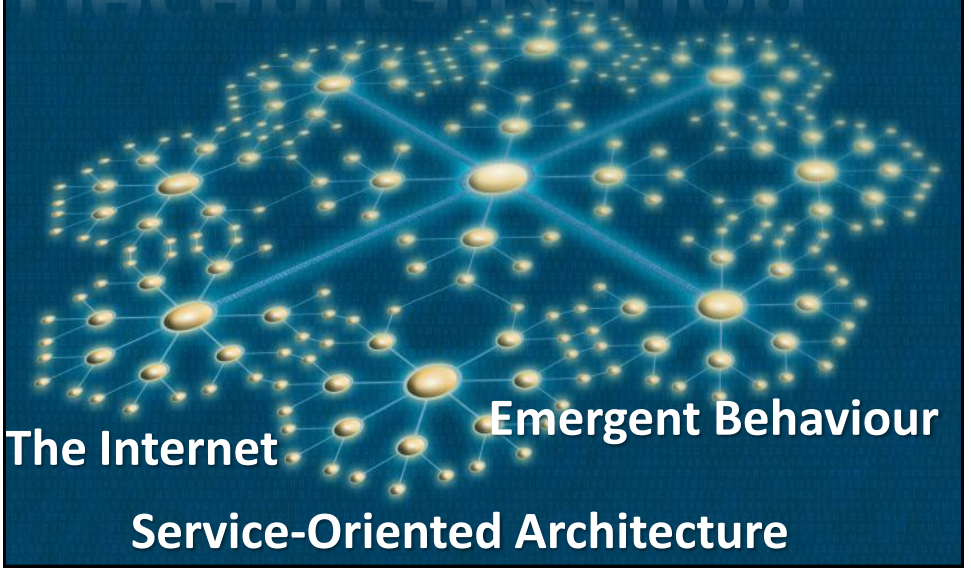
Handle Crosscuts

Aspect Orientation
Interceptors
Application Servers
Exception Handling

Isolate

Pure functional vs. Impure
Safety Critical Parts
OS Processes
Real-Time Kernel

Decentralization



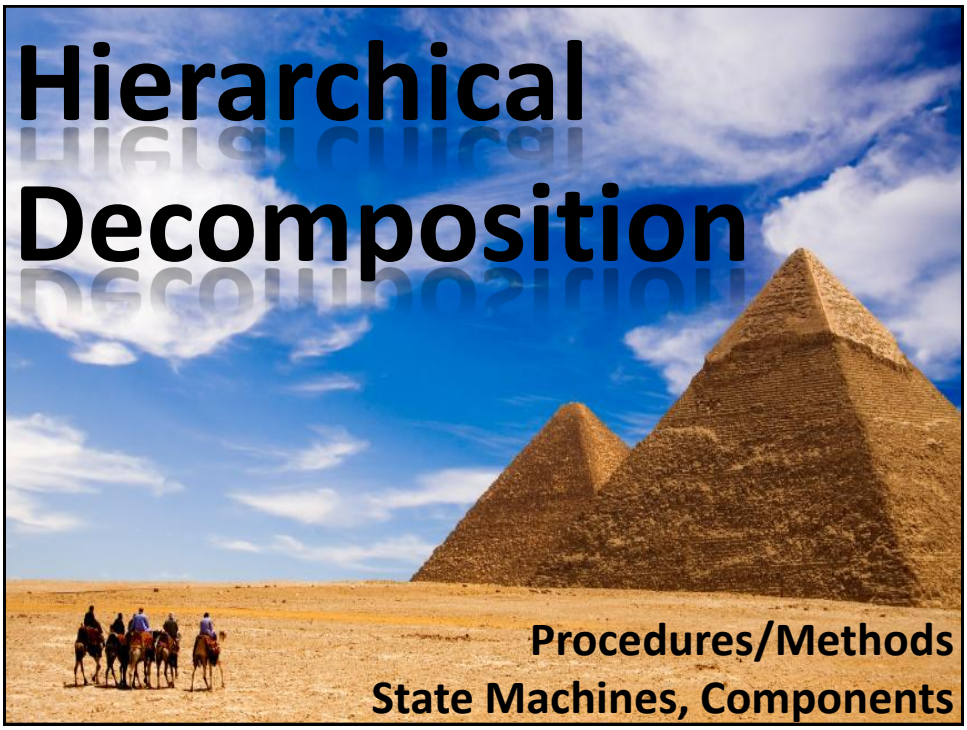
The Internet

Emergent Behaviour

Service-Oriented Architecture

Orthogonality

Closures, Program As Data, Macros, Higher-Order Functions



Protocols

Transctions
Locking/Synchronization
Resource Access

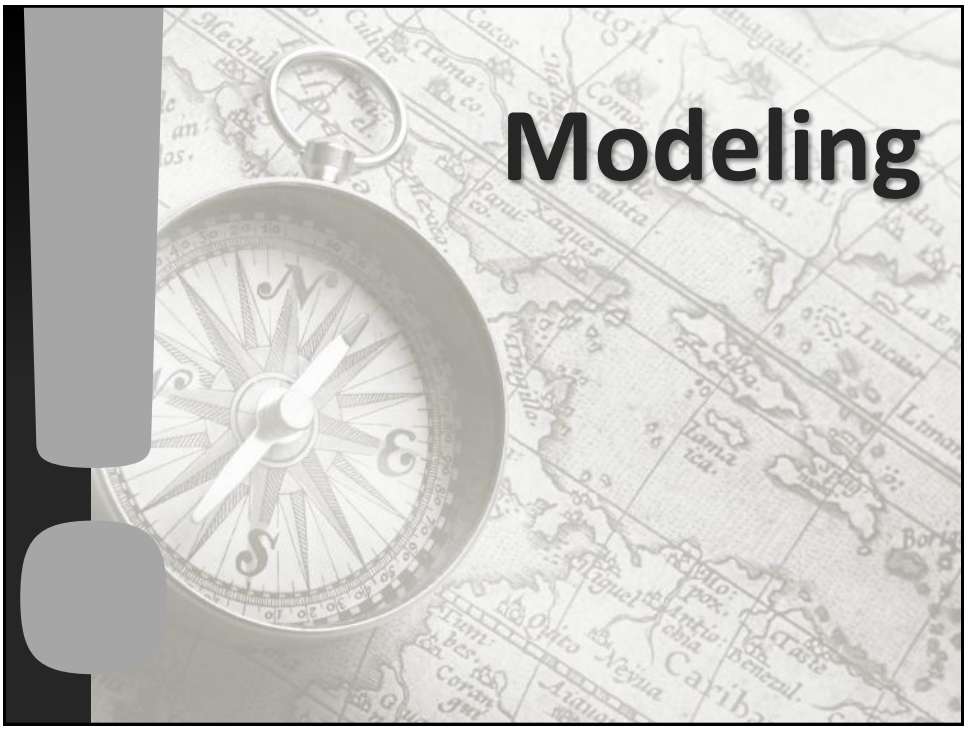

DOC Middleware
Orthogonal Persistence
(OR Mappers)
Make
Transparent

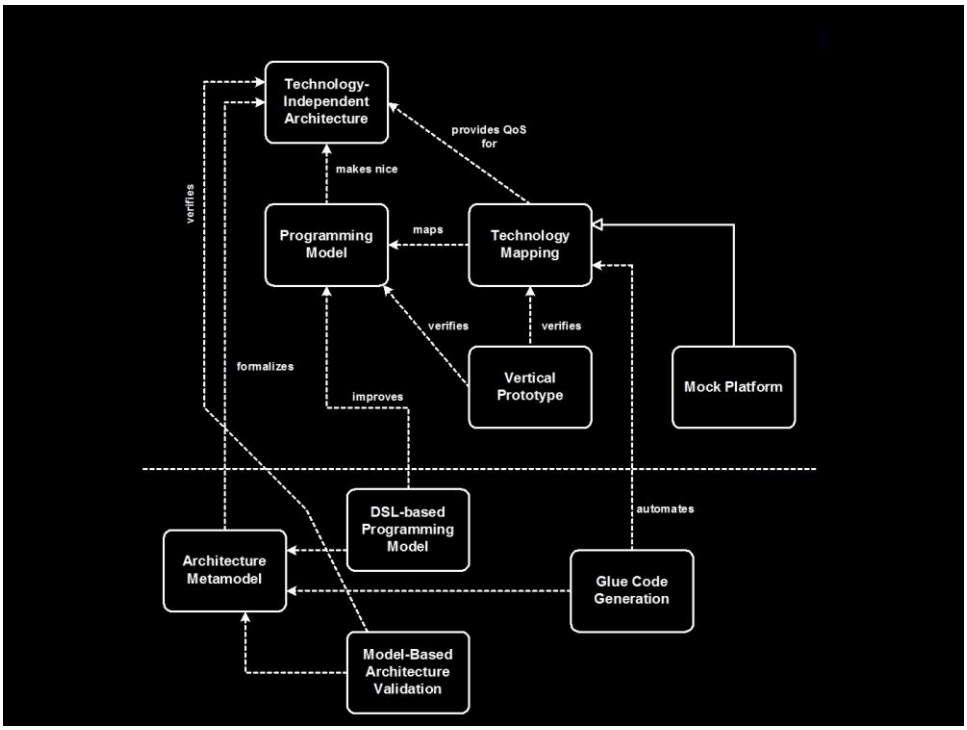

Make Explicit
Dependencies
SOA, Messaging
Monads
PLE Variabilities
Persistence: Loading Data

What is Architecting?

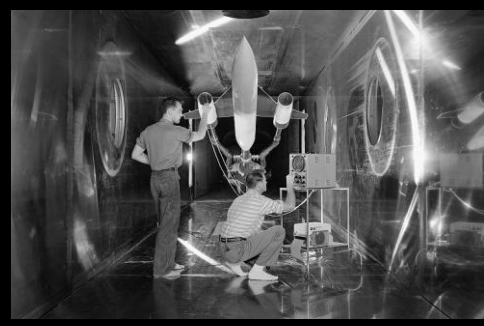
describe
verify
communicate
implement
check

describe
unambiguously
formally


Modeling


Technology-Independent Architecture
provides QoS for
makes nice
verifies
Programming Model
maps
Technology Mapping
verifies
verifies
formalizes
improves
Vertical Prototype
Mock Platform
DSL-based Programming Model
automates
Architecture Metamodel
Glue Code Generation
Model-Based Architecture Validation

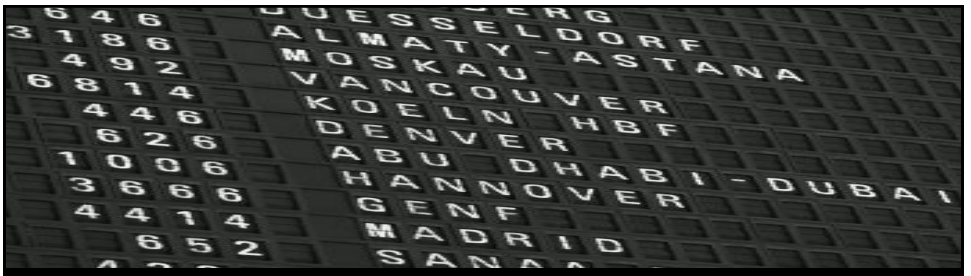
Domain Specific Languages



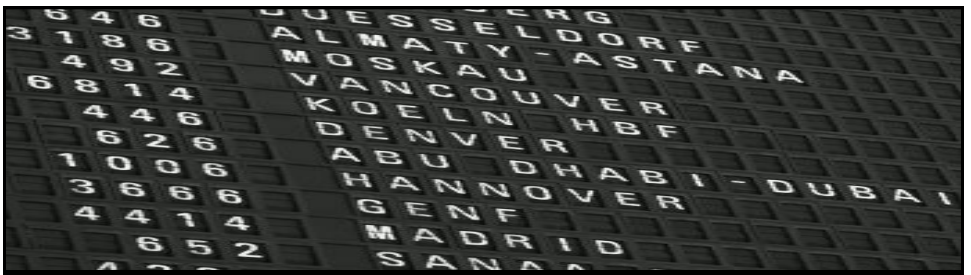
DSL

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.

Example 1:
Distributed System



Airport Management System



Monitors
Website
Aircraft-Module

Data Center

```
component DelayCalculator {}

component InfoScreen {}

component AircraftModule {}
```

```
component DelayCalculator {
  provides IDelayCalculator
  requires IInfoScreen
}
component InfoScreen {
  provides IInfoScreen
}
component AircraftModule {
  provides IAircraftModule
  requires IDelayCalculator
}

interface IDelayCalculator {}
interface IInfoScreen {}
interface IAircraftModule {}
```

```
component InfoScreen {
  provides IInfoScreen
}

instance screen1: InfoScreen
instance screen2: InfoScreen
…
```

```
component DelayCalculator {
  provides default: IDelayCalculator
  requires screens[0..n]: IInfoScreen
}

component InfoScreen {
  provides default: IInfoScreen
}

component AircraftModule {
  provides default: IAircraftModule
  requires calculator[1]: IDelayCalculator
}
```

```
component DelayCalculator {
  provides aircraft: IAircraftStatus
  provides managementConsole:
                IManagementConsole
  requires screens[0..n]: IInfoScreen
}

component Manager {
  requires backend[1]: IManagementConsole
}

component InfoScreen {
  provides default: IInfoScreen
}

component AircraftModule {
  requires calculator[1]: IAircraftStatus
}
```

```
component DelayCalculator {
  requires screens[0..n]: IInfoScreen
  …
}

component InfoScreen {
  provides default: IInfoScreen
}

instance dc: DelayCalculator
instance screen1: InfoScreen
instance screen2: InfoScreen

connect dc.screens
    to (screen1.default, screen2.default)
```

```
namespace com.mycompany.production {
  instance dc: DelayCalculator

  // InfoScreen instances are created and
  // started in other configurations
  dynamic connect dc.screens every 60 query {
    type = IInfoScreen
    status = active
  }
}
```

```
interface IAircraftStatus {

  oneway message reportPosition
                 (aircraft: ID, pos: Position )

  request-reply message reportProblem {
    request (aircraft: ID, problem: Problem,
             comment: String)
    reply (repairProcedure: ID)
  }

}
```

```
struct FlightInfo {
  from: Airport
  to: Airport
  scheduled: Time
  expected: Time
  …
}

replicated singleton flights {
  flights: FlightInfo[]
}

component DelayCalculator {
  publishes flights
}

component InfoScreen {
  consumes flights
}
```

```
interface IAircraftStatus {
  oneway message registerAircraft(aircraft: ID! )
  oneway message unregisterAircraft(aircraft: ID! )
  oneway message reportPosition(aircraft: ID!,
                 pos: Position! )
  request-reply message reportProblem {
    request (aircraft: ID!, problem: Problem!,
             comment: String!)
    reply (repairProcedure: !ID)
  }
  protocol initial = new {
    state new {
      registerAircraft => registered
    }
    state registered {
      unregisterAircraft => new
      reportPosition
      reportProblem
    }
  }
}
```

Example 2:
OSGi-based System

```
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]
        }

}
```

```
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

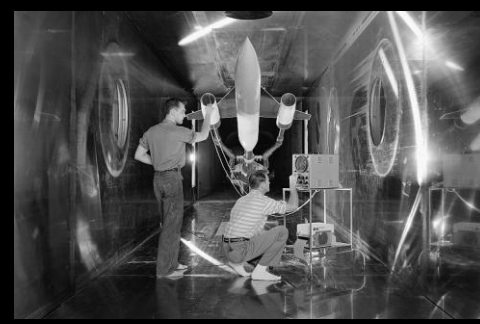

```
subsystem the.world.scenarios {

    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
    }
}
```

Example 3:
Embedded Protocol Handler



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

    datatypes {
        SinglePointIndicationWithoutTime;
        SinglePointIndicationWithTime;
        DoublePointIndicationWithoutTime;
        DoublePointIndicationWithTime;
        BitStringTypeI8BitWithoutTime;
        BitStringTypeI8BitWithTime;
    }

    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 ;
}
```

```
hal DigitalInHAL memory DID {
    command HighSpeedCounterInit;
    command HardwareInit;

    struct DID {
        int16 ch1_16;
        int32 hs_counter;
    }
}

halImpl DigitalInHALImpl for DigitalInHAL;
halImpl WinBiHALImpl for DigitalInHAL;
```

```
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];
    } ;
}
```

```
system DemoInput target atmel extendedModuleType 0x30 rack 1 {

    dip: DigitalIn slot 1 with DigitalInHALImpl channels 16
        using datatypes {
            SinglePointIndicationWithoutTime;
            SinglePointIndicationWithTime;
            BitStringTypeI8BitWithoutTime;
            BitStringTypeI8BitWithTime;
        };

    dispatcher {
        processingcomponent DigitalIn {
            on normalizeModule;
            on readStandardStatus;
            on setModuleInService;
            on readDataOfInputChannelX;
            on repeatLastReply;
            on resetModule;
            on writeRegisterNumberZ;
        }
    }
}
```

```
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];
    } ;
}
```

tests

```
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;
    };
};
```

Example 4:
Embedded Systems & Robots

```
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
  statemachine linefollower {
    event initialized;
    initial state initializing {
      initialized [true] -> runn
    }
    state running {

    }
  }

  initialize {
    ecrobot_set_light_sensor_act
    event linefollower:initializ
  }
```

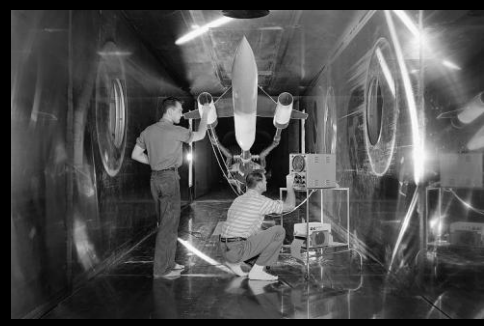
```
doc This is the cyclic task that is called every 1ms to do the actual control of the
task run cyclic prio = 2 every = 2 {
  stateswitch linefollower
    state running
      int32 light = 0;
      light = ecrobot_get_light_sensor(SENSOR_PORT_T::NXT_PORT_S1);
      if ( light < ( WHITE + BLACK ) /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

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

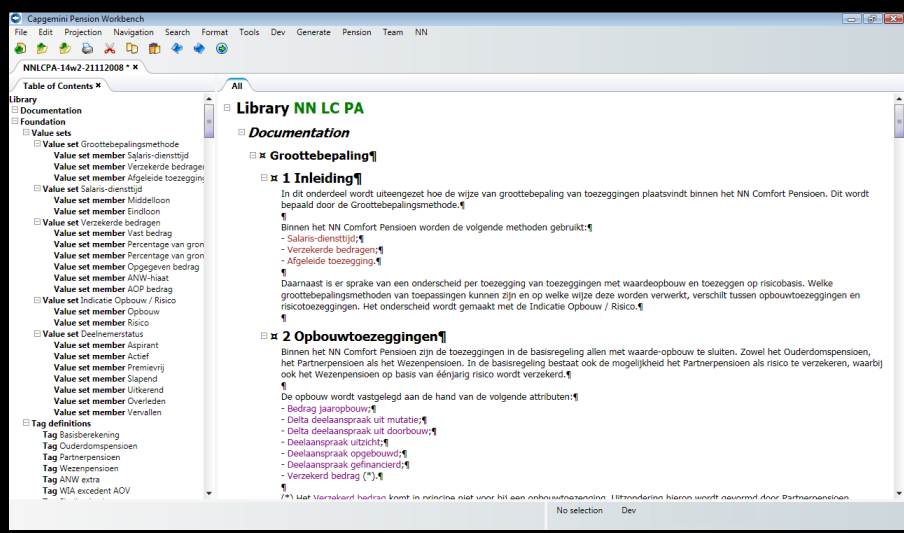
```
exported interface MotorControl {
  void stop(   );
  void setLeftSpeed( int8 speed );
  void setRightSpeed( int8 speed );
}

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

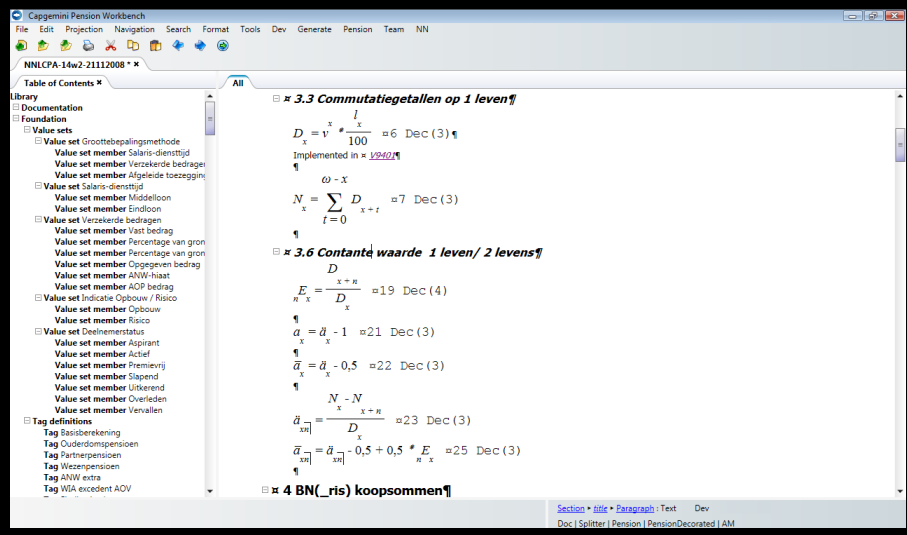
```
exported component implementation MotorsNXT : Motors {

  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);
  }
}
```

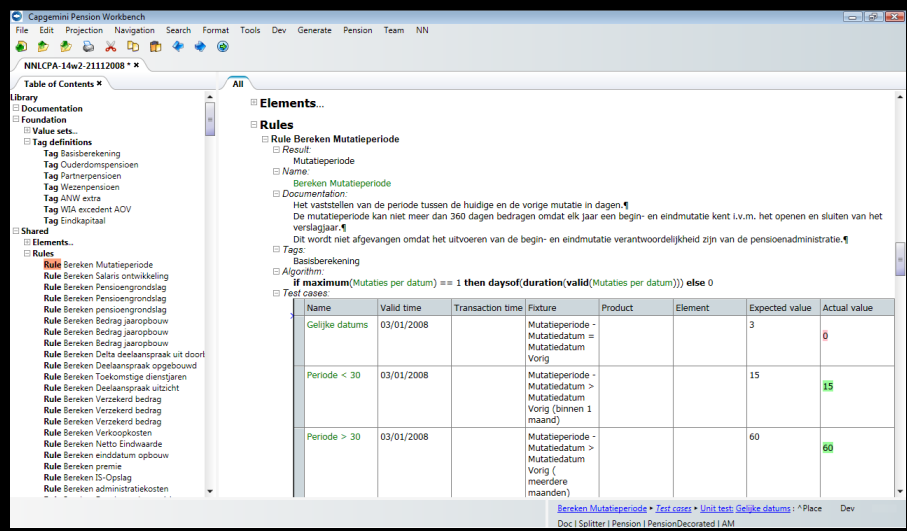
```
module impl imports <<imports>> {

  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

}
```

Example 5:
„Business“ Architecture






Capgemini Pension Workbench
File Edit Projection Navigation Search Format Tools Dev Generate Pension Team NN
NNLCPA-14w2-21112008 * ×
Table of Contents ×
All
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
¤ 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¶
$E_{n\ x} = \frac{D_{x+n}}{D_x}$ ¤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)
$\bar{a}_{x\overline{n}|} = \ddot{a}_{x\overline{n}|} - 0{,}5 + 0{,}5 * E_{n\ x}$ ¤25 Dec(3)
¤ 4 BN(_ris) koopsommen¶
Section ▸ Title ▸ Paragraph : Text Dev
Doc | Splitter | Pension | PensionDecorated | AM


Capgemini Pension Workbench
File Edit Projection Navigation Search Format Tools Dev Generate Pension Team NN
NNLCPA-14w2-21112008 * ×
Table of Contents ×
All
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 pensioengrondslag
Rule Bereken Bedrag jaaropbouw
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 eindtdatum 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

What did we do?

As you understand and develop your Architecture…

Develop a language to express it!

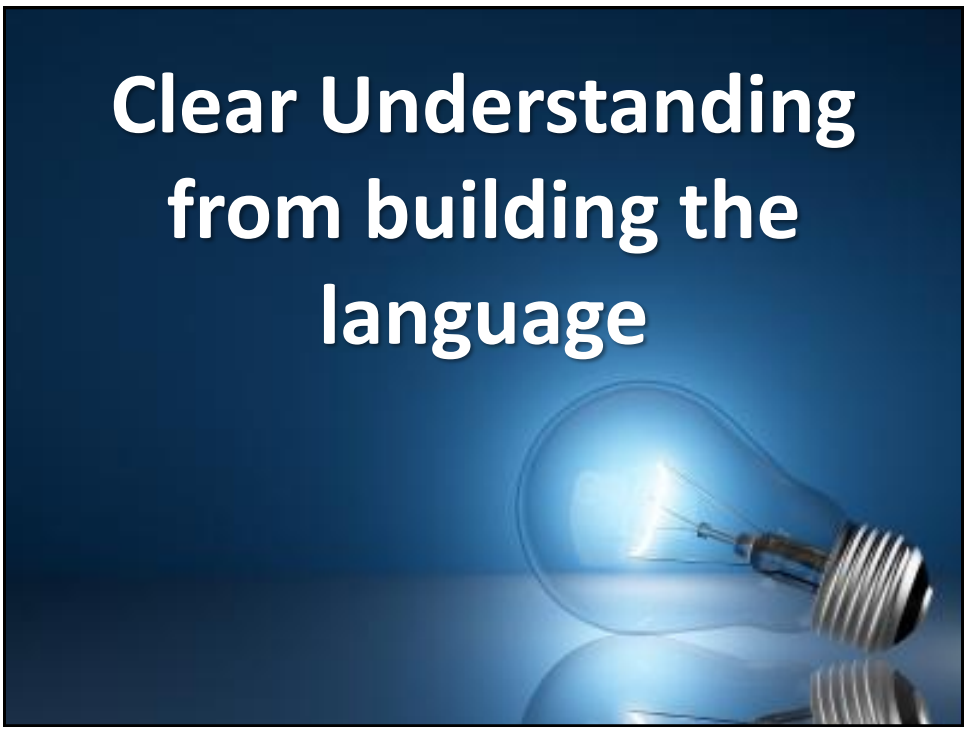
Language resembles architectural concepts

We express the application(s) with the language.

Architecture

DSL

Clear Understanding from building the language



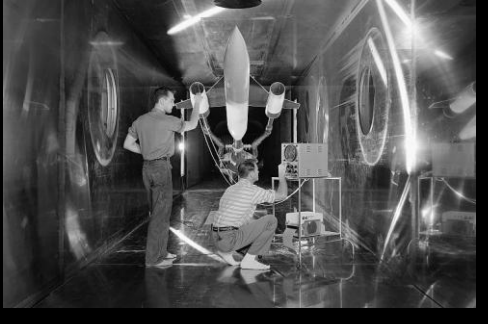
Unambigious Vocabulary

Concepts independent from Technology

Programming Model can be defined based on Conceptual Arcitecture

Architecture „executable“
(i.e. more than rules and docs)

Why textual?



Languages and Editors are easier to build

Languages and Editors are easier to build

Evolve Language and simple editor as you understand and discuss the architecture, in real time!

Integrates easily with current infrastructure: CVS/SVN diff/merge

adapting existing models as the DSL evolves

Model evolution is trivial, you can always use *grep*.

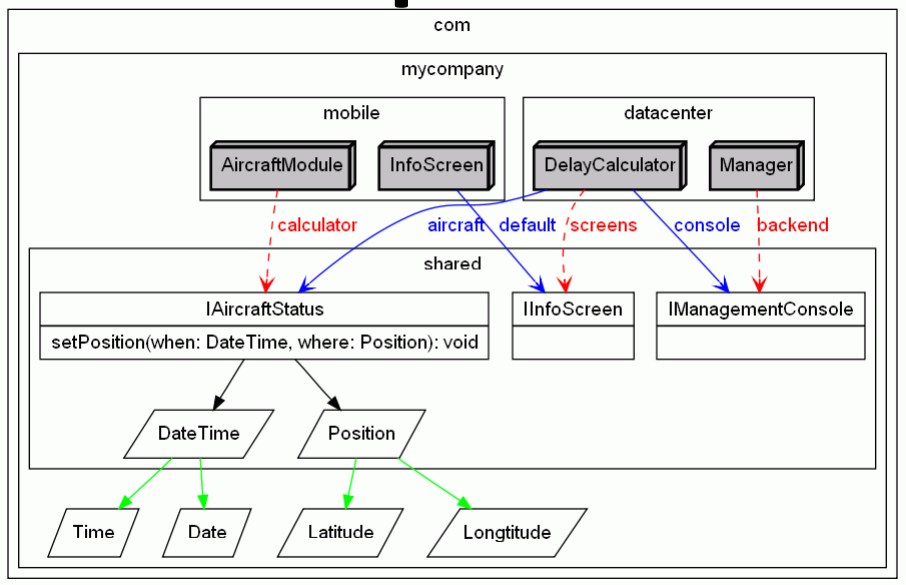
Many Developers prefer textual notations

When a graphical notation is better, you can visualize.

Via M2M
Read-Only
Auto-Layout
Drill-Down

Textual DSLs
vs.
Graphical
vs.
Visualization

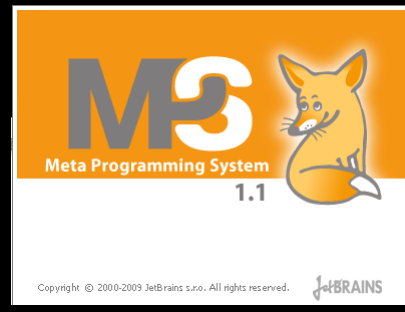
Graphviz


com
mycompany
mobile
datacenter
AircraftModule
InfoScreen
DelayCalculator
Manager
calculator
aircraft
default
screens
console
backend
shared
IAircraftStatus
setPosition(when: DateTime, where: Position): void
IInfoScreen
IManagementConsole
DateTime
Position
Time
Date
Latitude
Longtitude

Tools

eclipse modeling PROJECT

xtext


MPS
Meta Programming System
1.1
Copyright © 2000-2009 JetBrains s.r.o. All rights reserved.
JetBRAINS


INTENTIONAL®
SOFTWARE

Validation

Grammar is not expressive enough

More Validation Rules Required: Constraints

Simple Examples:

Name-Uniqueness
Type Checks
Non-Nullness

More Complex Examples:

Version Compatibility
Overall completeness
Quality of Service

```
context Interface ERROR "interface names must start "+
                        "with a capital I":
  name.startsWith("I");


context Component ERROR "Qualified Name "+
                        qualifiedName()+" must be unique" :
   allComponents().
     select( c | c.qualifiedName() == qualifiedName() )
     .size == 1;


context Attribute ERROR "no type defined: "+
                        type.name:
  visibleInstancesOfType(this, DataType)
    .contains(type);


context Connector ERROR "target must be provided port":
  ProvidedPort.isInstance( u);
```

Precondition I:
Algorithm for Constraint

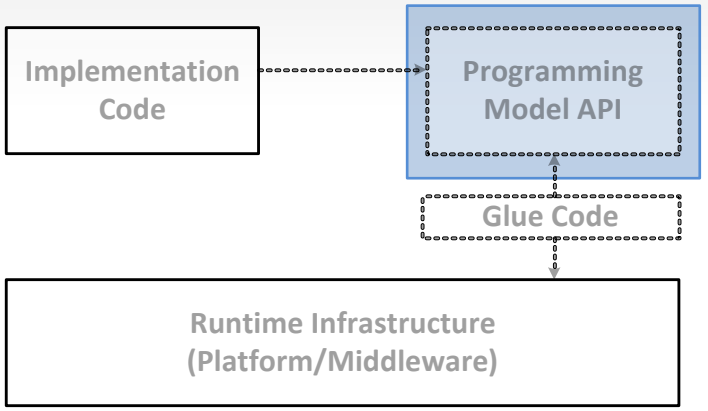

Precondition II:
All Data Available

Generating Code

Since we already have a formal model….

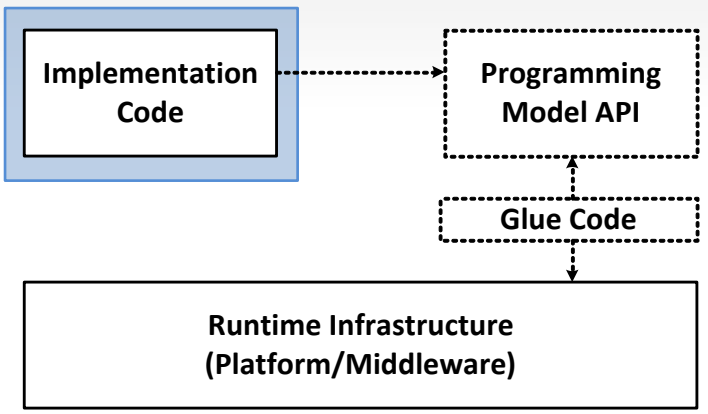
Generate API

Maps Architectural Concepts to Implementation language (non-trivial!)



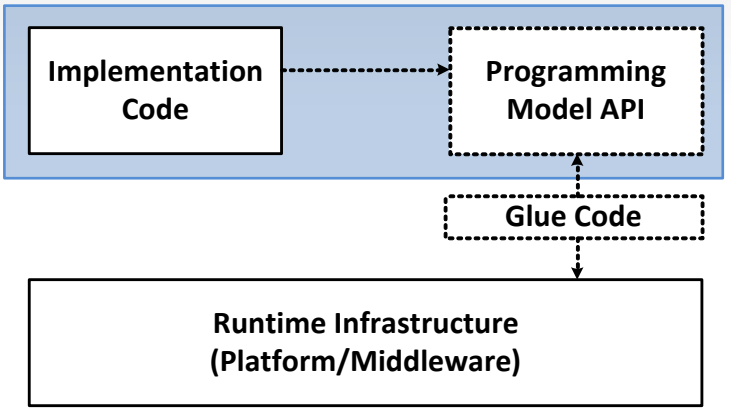
Implementation

Implementation only depends on the generated programming model API



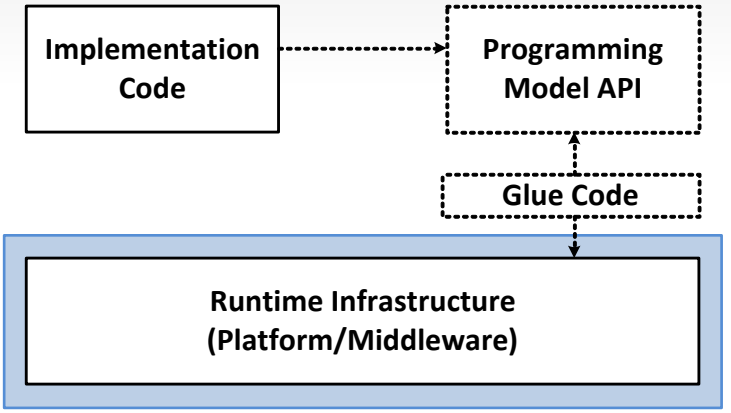
Programming Model

Generated API + Usage Idioms
Completely Technology-Independent



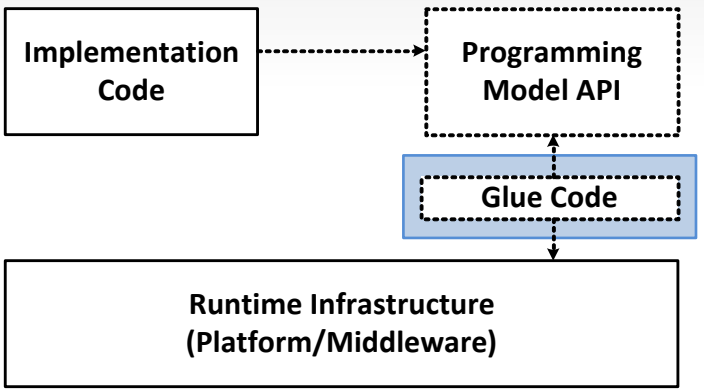
Runtime Infrastructure

Select based on fit wrt. to architectural concepts and non-functional requirements



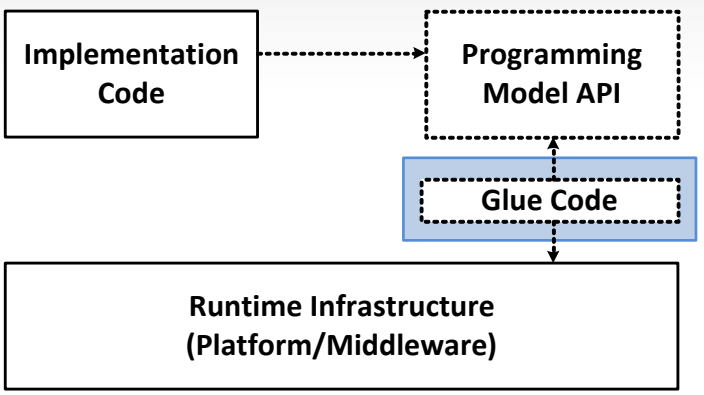
Glue Code

Aka Technology Mapping Code
Maps API to selected platform



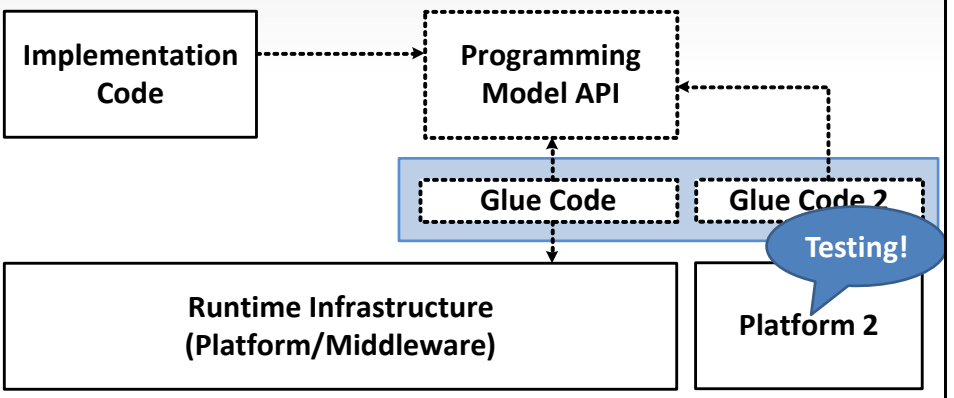
Glue Code

Contains Configuration Files for Platform
Might require „mix in models“

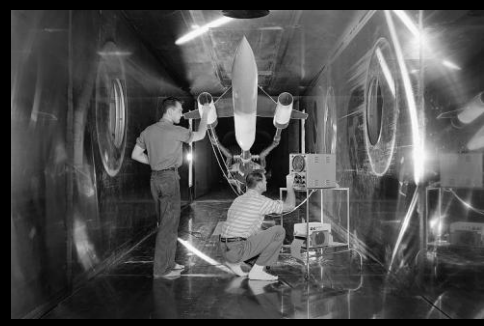


Several Platforms

Different Platforms, not Languages
Support for Scaling (non-functional req)



Architecture Analysis Tools



Software Tomography
SonarJ
Structure 101

… and the like

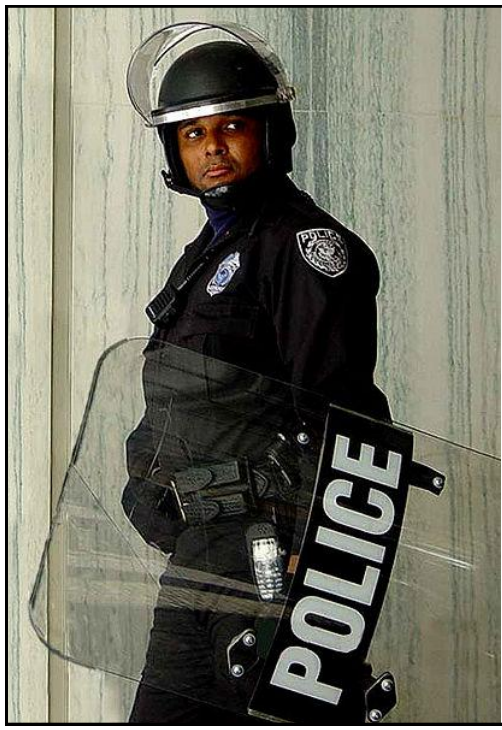
How do you describe Architecture?

Constraint-Driven:

Assemble Packages into Components
Define layers/columns
Define and Check valid Dependencies

You do not get a way to define and express architectural abstractions beyond those!

Dependency Analysis
Anti Pattern Detection
Metrics
[Trend Analysis]
[Visualization]



Police Approach:

Detect Problems After the fact!

Anti Pattern Detection Metrics

Useful for manually written parts of the system

Dependency Analysis

Useful for manually written parts of the system

Constraints generated from the model.

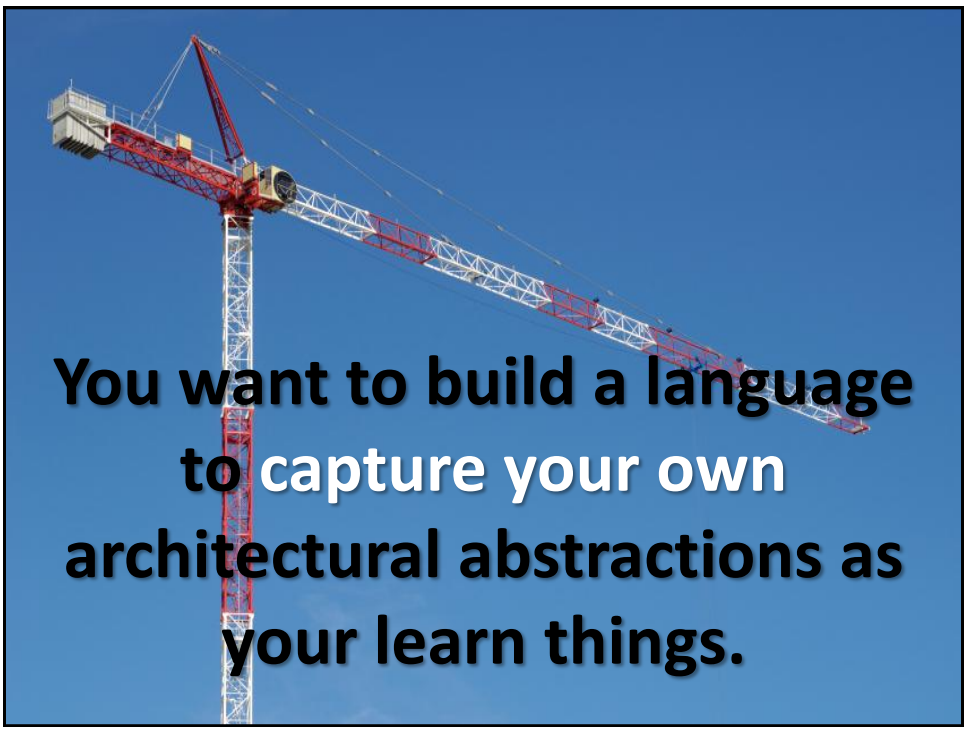
Standards & UML

Formal Architecture Description **is not new:** ADLs, UML

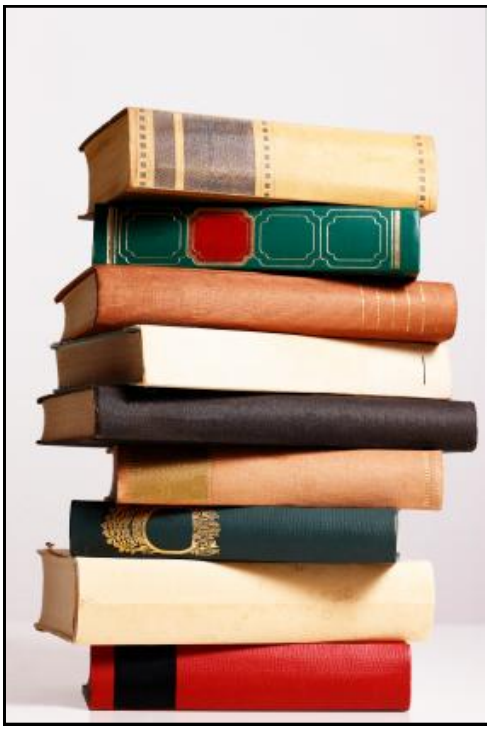
But all of those use
existing, generic
languages!

This misses the point!

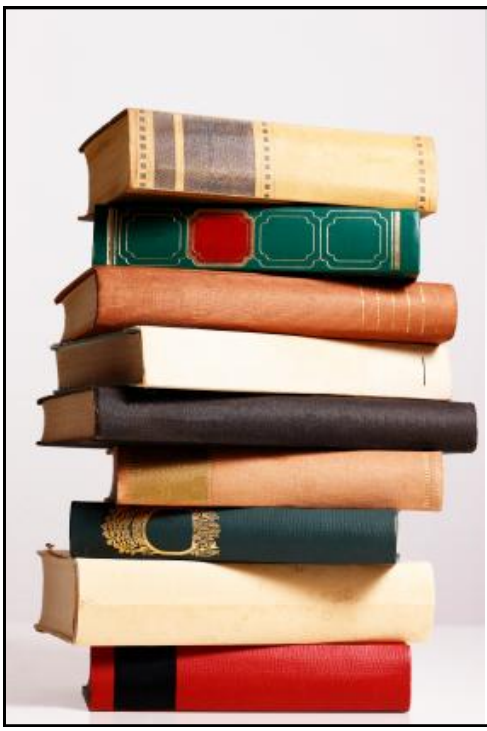
Trying to express **your specific** architecture with **predefined** abstractions is not useful!



You want to build a language to capture your own architectural abstractions as your learn things.



Where are standards useful?



People have to learn architectural concepts anyway.

Is UML with a profile still a **standard language**?

On which **meta level** do I want to standardize?

M2 (UML), M3 (MOF)?

Isn‘t a DSL based on MOF as „standard“ as a profile based on UML?

UML Profiles instead?

You‘ll think more about UML-ities than your own concepts

UML Profiles instead?

Tool integration issues (repository, diff/merge, versioning)

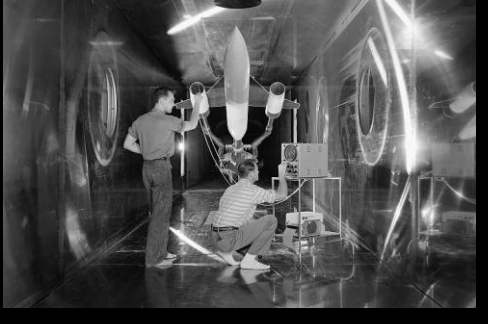
UML Profiles instead?

Tools are often complex, heavyweight, bloated. Acceptance limited.

UML Generally Useless?

No. UML can be used for docu-mentation (sequence diagrams, eg)

Documentation



The DSL and the „programs“ are documentation.

So I don‘t have to write…
… more docs?

Not quite.

1

Grammar is a good formal definition, but not a teaching tool.

Grammar is a good formal definition, but not a teaching tool.

→ Tutorials

Tutorials:

Architectural Concepts (Meta Model)
How to use the language
How to use the programming model
(How to generate code)
(How to add manual code)

Tutorials:

Architectural Concepts (Meta Model)
How to use the language
How to use the programming model
(How to generate code)
(How to add manual code)

Example Driven!

2

Grammar describes what the architecture looks like.

Grammar describes what the architecture looks like.

But: Why is it that way?

Rationales

Rationales

Conceptual Architecture

Relate to Grammar

Rationales

Conceptual Architecture

Relate to Grammar

Technological Decisions

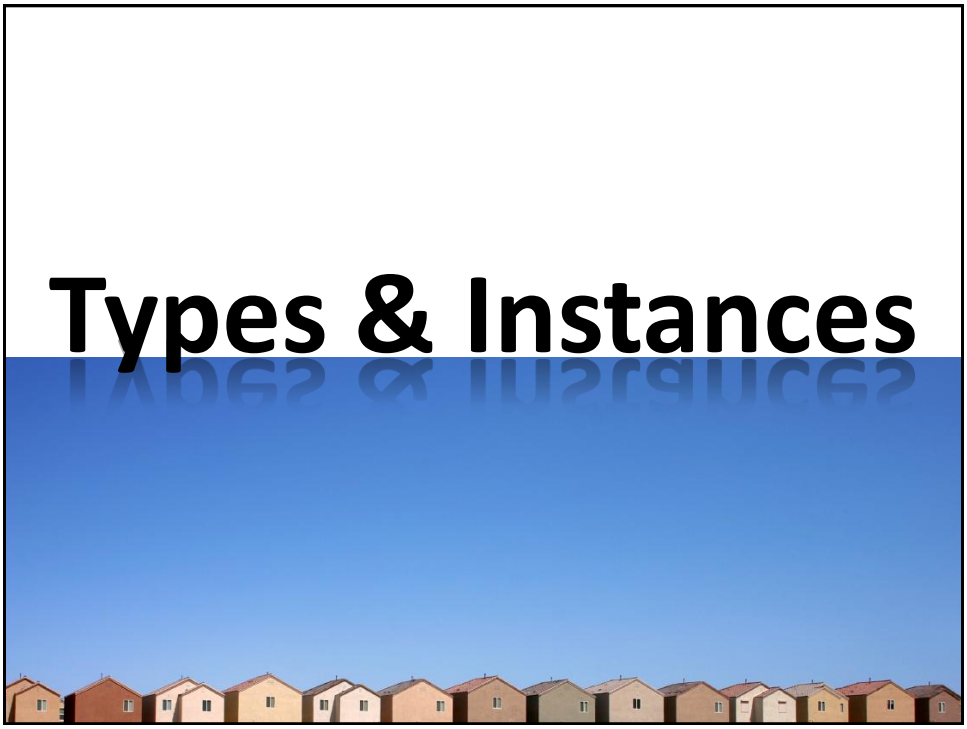
Non-Func. Req.

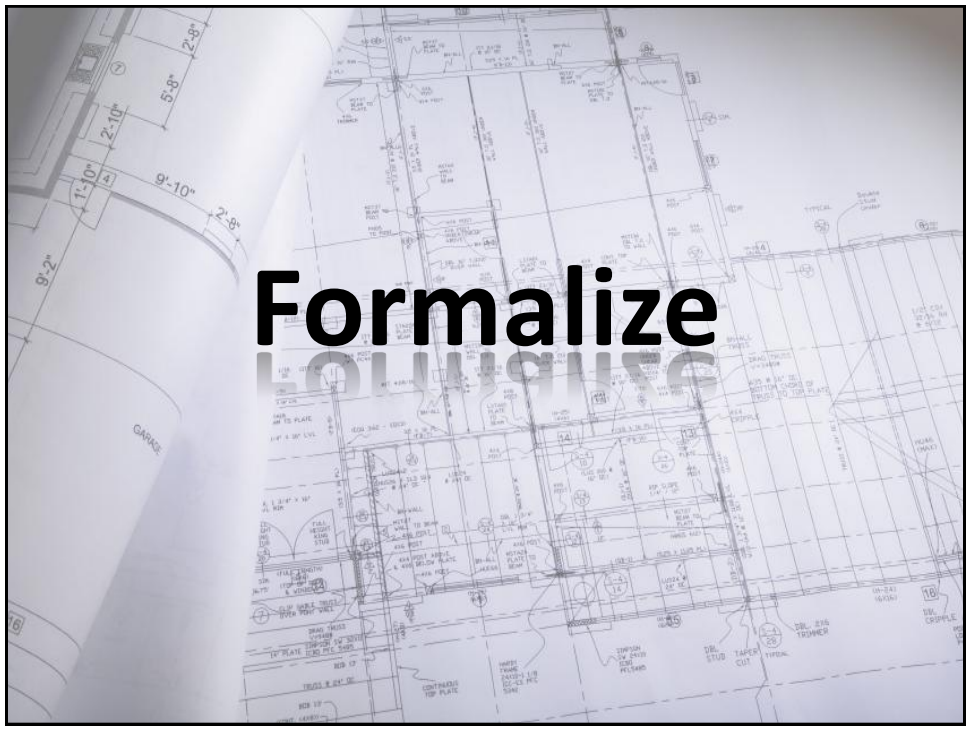
Summary

Make Explicit

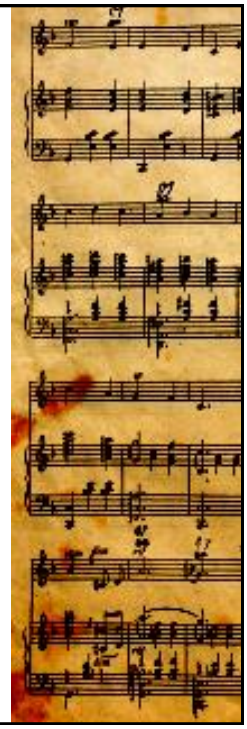
Abstraction


Viewpoints


Types & Instances



Notation

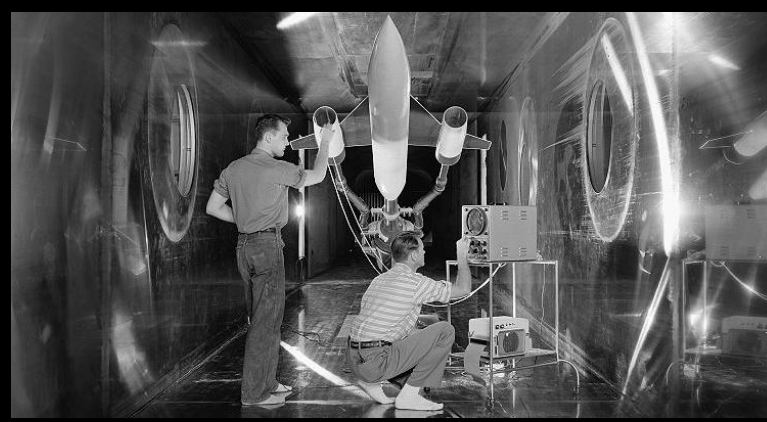


Isolate Technology

Translate

Automate

THE END.



.coordinates

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

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

itemis