

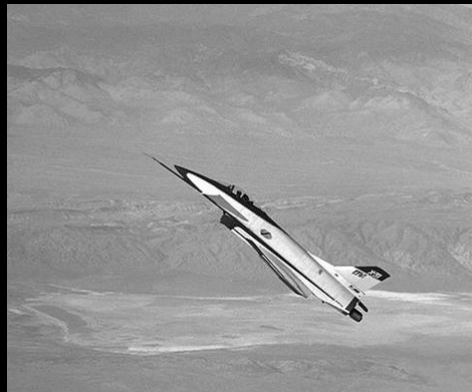
# Domain Specific Languages and Requirements (Engineering)

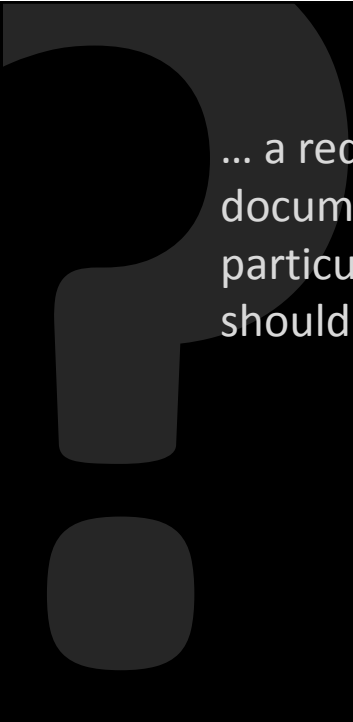


itemis

Markus Voelter  
[www.voelter.de](http://www.voelter.de)  
[voelter@acm.org](mailto:voelter@acm.org)

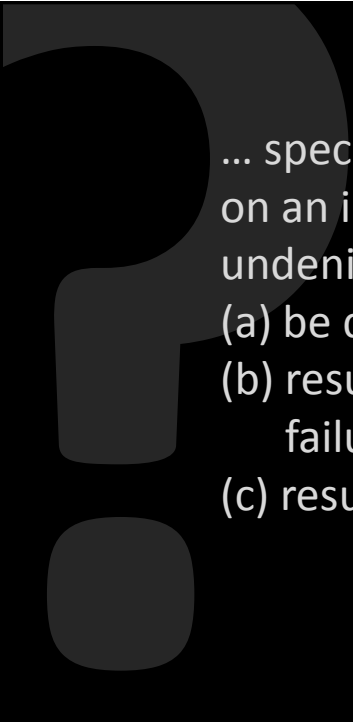
## What are Requirements?





... a requirement is a singular documented need of what a particular product or service should be or perform.

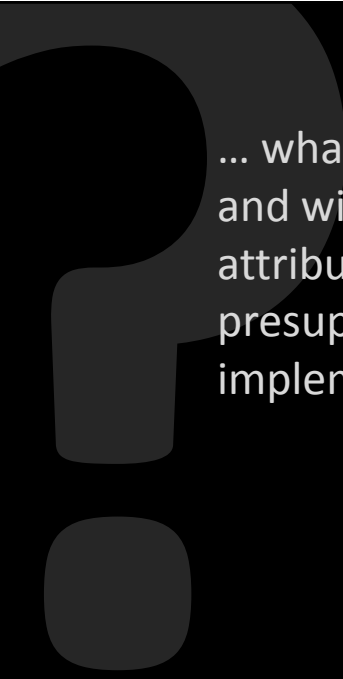
Wikipedia



... specifies a verifiable constraint on an implementation that it shall undeniably meet or


- (a) be deemed unacceptable, or
- (b) result in implementation failure, or
- (c) result in system failure.

Wiktionary



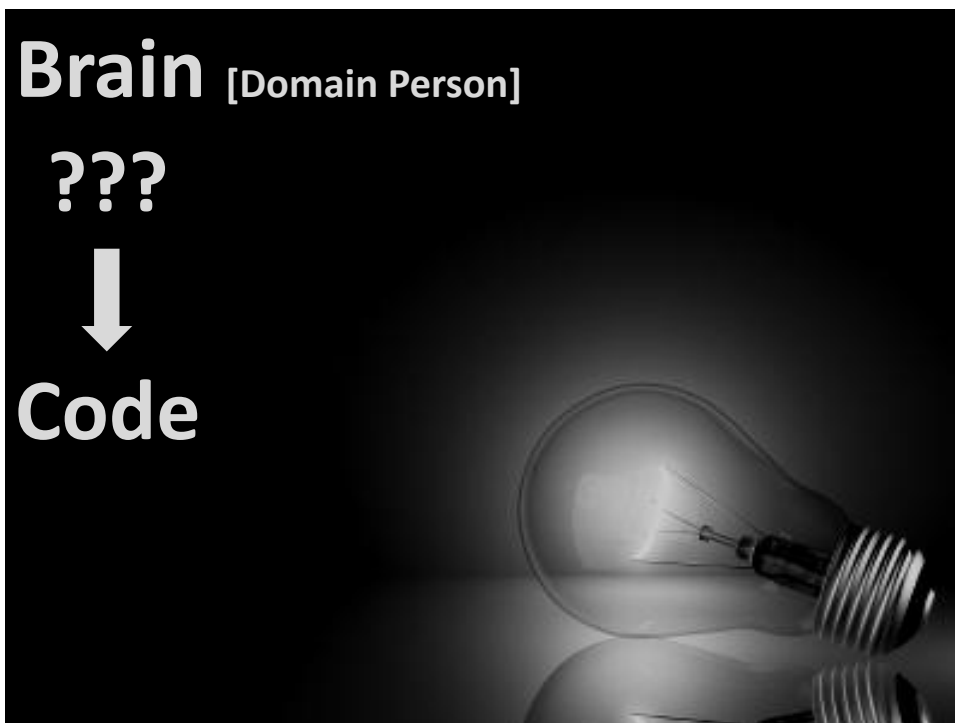
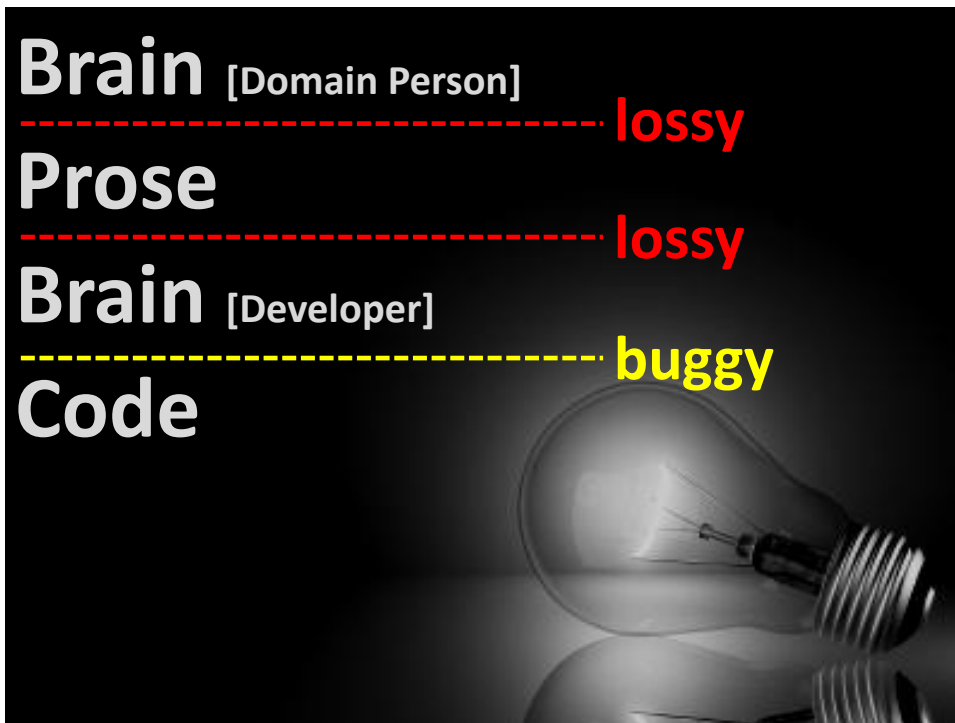
... what a system should do,  
and with which quality  
attributes, without  
presupposing a specific  
implementation.

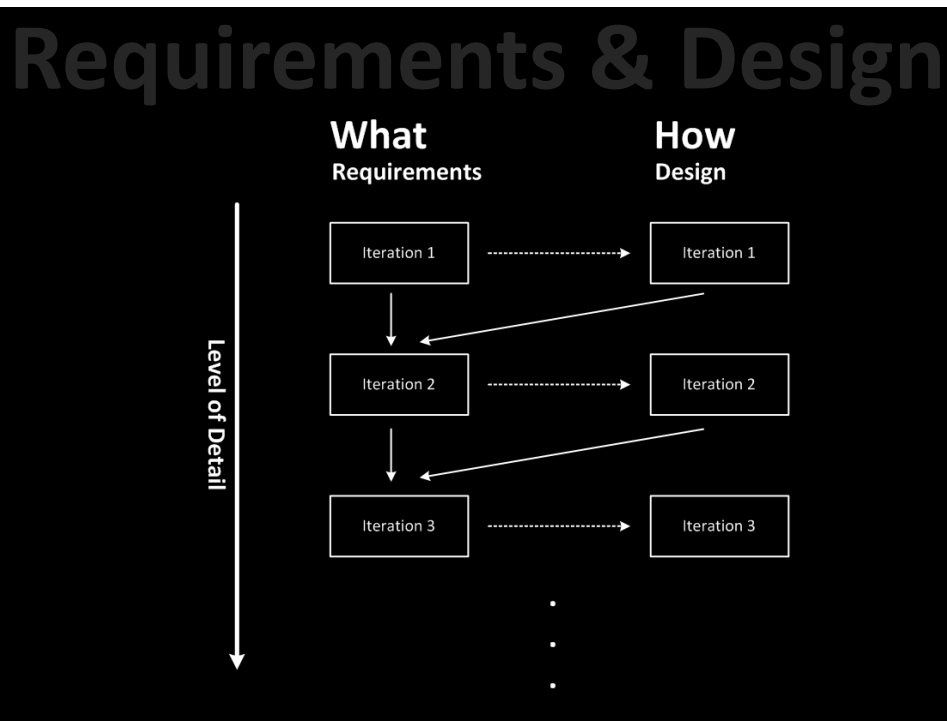
Mine.



Cohesive  
Complete  
Consistent  
Atomic  
Traceable  
Current  
Feasible  
Unambiguous  
Mandatory  
Verifiable

Wikipedia





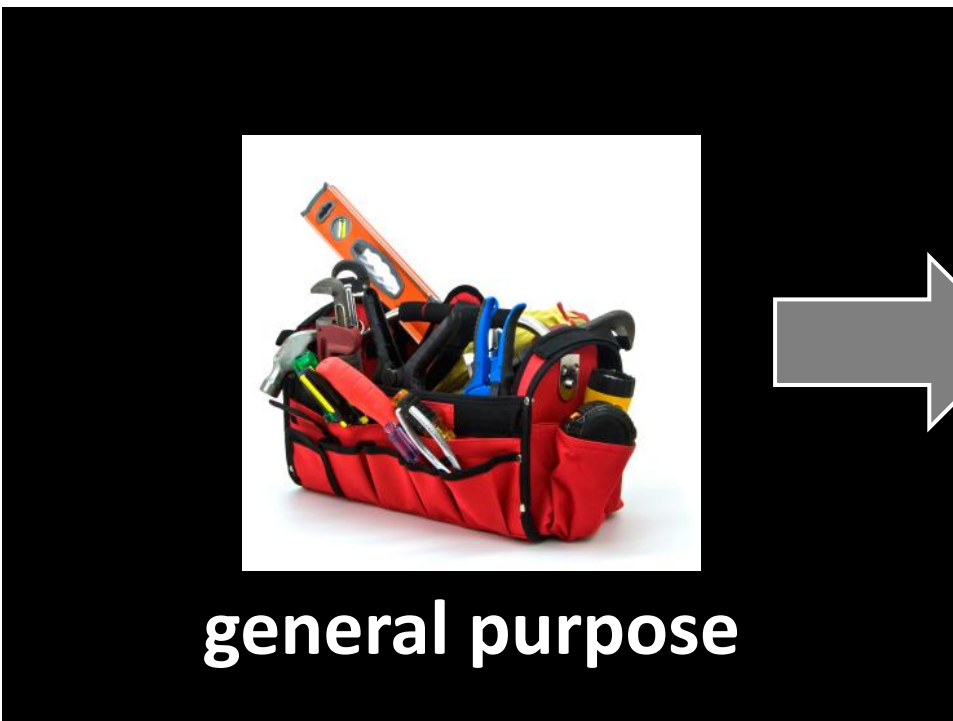
**Formal** requirements specify **what** a system should do from a **domain** perspective, and with which quality attributes, without presupposing a specific software implementation, but **processable by tools**.

# Domain Specific Languages



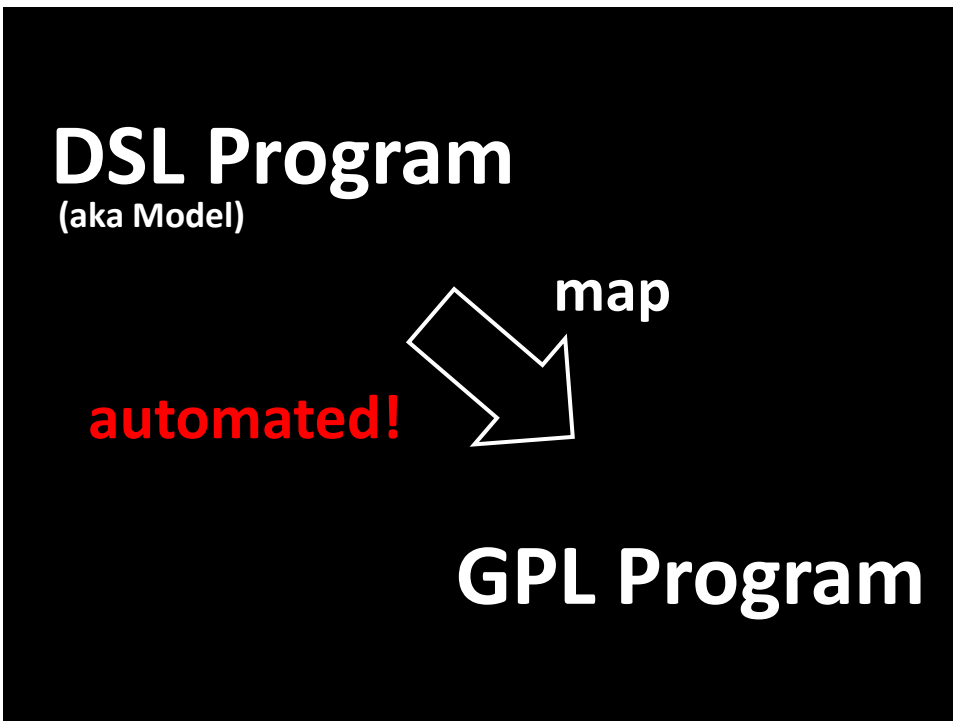
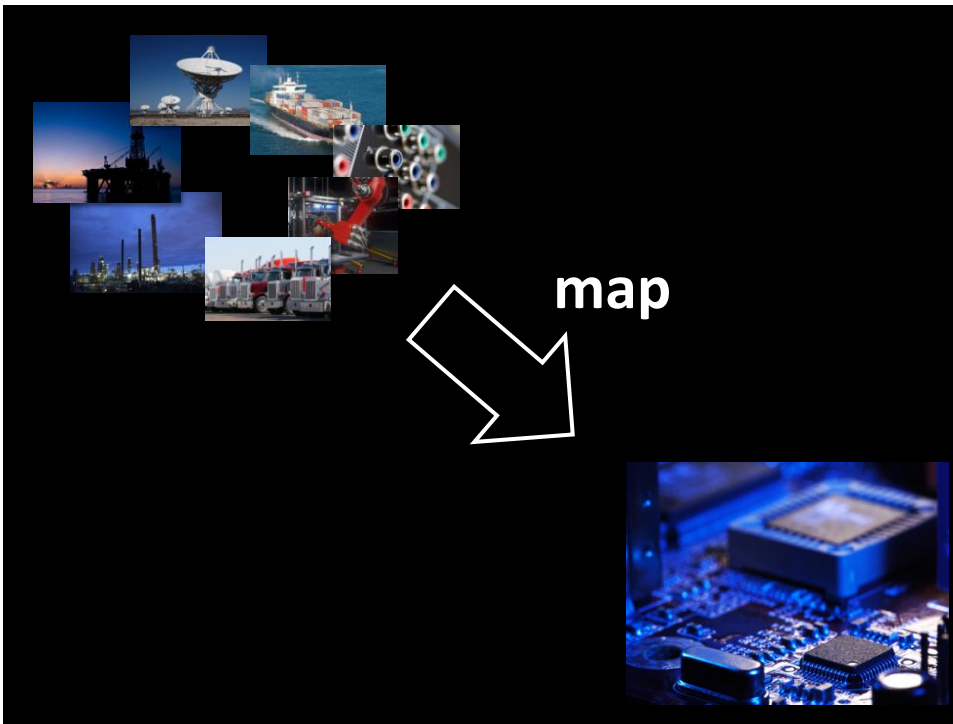
# DSL

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.











# Generation

Transformation  
Compilation

# Interpretation

Example 1:

## Pension Fund Specification



# Textual Documentation

SOLUTION

Capgemini Pension Workbench

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

NLCPA-14w2-21112008 \* x

Table of Contents \* All

Library

- Documentation
  - Foundation
    - Value sets
      - Value set Groottebepalingmethode
        - Value set member Salaris-diensttijd
        - Value set member Verzekerde bedragen
      - Value set Salaris-diensttijd
        - Value set member Middeloon
        - Value set member Eindloon
      - Value set Verzekerde bedragen
        - Value set member Vast bedrag
        - Value set member Percentage van gron
        - Value set member Opgegeven bedrag
      - Value set ANW-hiaat
        - Value set member AOP bedrag
      - Value set Indicatie Opbouw / Risico
        - Value set member Opbouw
        - Value set member Risico
      - Value set Doelmeritatus
        - 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 Ouderdompensioen
      - Tag Partnerpensioen
      - Tag Wezempensioen
      - Tag ANW extra
      - Tag WA excident AOV

Library NN LC PA

Documentation

- Groottebepaling
  - Inleiding
    - In dit onderdeel wordt uiteengezet hoe de wijze van groottebepaling van toezeggingen plaatsvindt binnen het NN Comfort Pensioen. Dit wordt bepaald door de Groottebepalingmethode.
    - Binnen het NN Comfort Pensioen worden de volgende methoden gebruikt:
      - Salaris-diensttijd
      - Verzekerde bedragen
      - Afgeleide toezegging
    - Daarnaast is er sprake van een onderscheid per toezegging van toezeggingen met waardeopbouw en toezeggen op risicobasis. Welke groottebepalingmethoden van toepassingen kunnen zijn en op welke wijze deze worden verwerkt, verschilt tussen opbouwtoezeggingen en risicotoezeggingen. Het onderscheid wordt gemaakt met de Indicatie Opbouw / Risico.
  - Opbouwtoezeggingen
    - Binnen het NN Comfort Pensioen zijn de toezeggingen in de basisregeling allen met waarde-opbouw te sluiten. Zowel het Ouderdompensioen, het Partnerpensioen als het Wezempensioen. In de basisregeling bestaat ook de mogelijkheid het Partnerpensioen op basis van éénjarig risico wordt verzekerd.
    - De opbouw wordt vastgelegd aan de hand van de volgende attributen:
      - Bedrag jaaropbouw
      - Delta deelaanspraak uit mutatie
      - Delta deelaanspraak uit doorbouw
      - Deelaanspraak ultsicht
      - Deelaanspraak opgebouwd
      - Deelaanspraak gefinancierd
      - Verzekerd bedrag

No selection Dev

# Insurance Mathematics

SOLUTION

Capgemini Pension Workbench

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

NLCPA-14w2-21112008 \* x

Table of Contents \* All

Library

- Documentation
  - Foundation
    - Value sets
      - Value set Groottebepalingmethode
        - Value set member Salaris-diensttijd
        - Value set member Verzekerde bedragen
      - Value set Salaris-diensttijd
        - Value set member Middeloon
        - Value set member Eindloon
      - Value set Verzekerde bedragen
        - Value set member Vast bedrag
        - Value set member Percentage van gron
        - Value set member Opgegeven bedrag
      - Value set ANW-hiaat
        - Value set member AOP bedrag
      - Value set Indicatie Opbouw / Risico
        - Value set member Opbouw
        - Value set member Risico
      - Value set Doelmeritatus
        - 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 Ouderdompensioen
      - Tag Partnerpensioen
      - Tag Wezempensioen
      - Tag ANW extra
      - Tag WA excident AOV

Library NN LC PA

Documentation

- 3.3 Commutatieteggetallen op 1 leven
  - $$D_x = v \cdot \frac{1}{i} = 6 \text{ Dec (3)}$$
    - Implemented in `1962`
  - $$N_x = \sum_{t=0}^{\omega-x} D_{x+t} = 7 \text{ Dec (3)}$$
- 3.6 Contante waarde 1 leven/ 2 levens
  - $$E_{\overline{n}|x} = \frac{1-v^n}{i} = 19 \text{ Dec (4)}$$
  - $$\ddot{a}_x = \frac{1-v^n}{d} = 21 \text{ Dec (3)}$$
  - $$\ddot{a}_x = \ddot{a}_x + 0,5 = 22 \text{ Dec (3)}$$
  - $$\ddot{a}_{\overline{n}|x} = \frac{1-v^n}{d} = 23 \text{ Dec (3)}$$
  - $$\ddot{a}_{\overline{n}|x} = \ddot{a}_{\overline{n}|x} - 0,5 + 0,5 \cdot v^n = 25 \text{ Dec (3)}$$
- 4 BN(ris) koopsommen

Section • Title • Paragraph | Text Dev  
Doc | Splitter | Pension | PensionDecorated | AM

# Calculation Rules and Tests

SOLUTION

The screenshot displays the Capgemini Pension Workbench interface. The main window shows a list of rules under the 'Rules' section, including 'Rule Berekken Mutatieperiode'. Below the rules, a table displays test results for various scenarios.

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

 INTENTIONAL<sup>®</sup>  
SOFTWARE

## Intentional Software's Domain Workbench

TOOLS

## Example 3: Radar Systems Engineering



## Component Definition

```

import "classpath:/test.mm"

quantity voltage is double
quantity temperature is double

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

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

processing component Processor {
  consumes input: { measurement: voltage }
  produces earthtemp: { temperature }
  behavior {
    earthtemp.temperature = input.measurement
  }
}

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

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

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

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

```

**SOLUTION**

## Component Behavior Specification

SOLUTION

```

ComponentDSL.txt  test.in  C
BeginPackage["MappingSatellite"]
Begin["Private"]

sensorM[t_Int] := 23*t
sensorT[t_Int] := 300
calibrate[ m_Double, temp_Double ] := m - temp/10
process[ m_Double ] := m*3

End[ ]
EndPackage[ ]

(* Mathematica Raw Program *)

```

## Resulting System Behaviour

SOLUTION

```

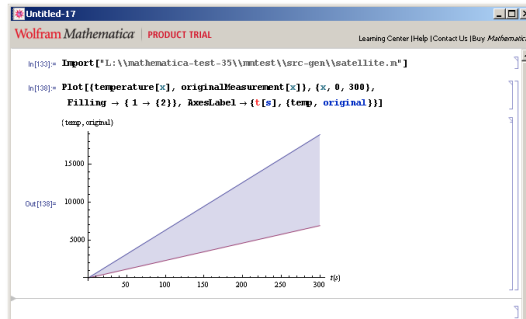
satellite.m
BeginPackage["satellite"]
Begin["Private"]

temperature[t_Int] := process[ calibrate[ sensorM[ t ], sensorT[ t ] ] ]
originalMeasurement[t_Int] := sensorM[ t ]

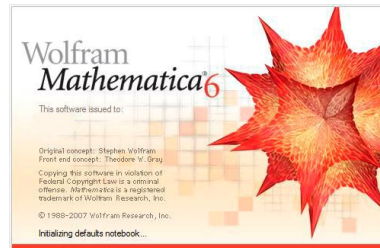
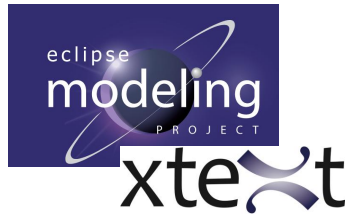
End[ ]
EndPackage[ ]

```

## Analysis



SOLUTION



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

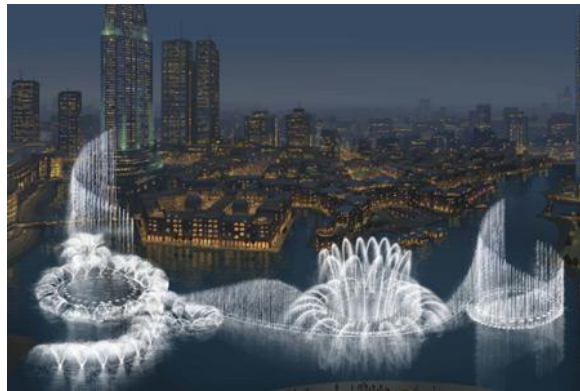
TOOLS

## Example 2: Fountains

---



CONTEXT





## Hardware Structure

SOLUTION

```

feature BasicOnePump
  pump compartment cc1
  static compressor c1

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

feature[f] SuperPowerCompartment
  water compartment adds to f
  superPowerMode

feature WithAlarm
  level alarm al

fountain StdFountain extends AtLeastOneZone

```

## Behaviour

SOLUTION

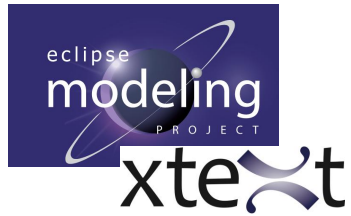
```

pumping program P1 for AtLeastOneZone + WithAlarm +
  SuperPowerCompartment[f=comb1] {
  parameter defaultWaterLevel : int
  parameter superWaterLevel: int
  event superPowerTimeout

  init {
    set comp1->targetHeight = defaultWaterLevel
  }

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

```

**SOLUTION****Plus:****In-IDE Simulator  
Unit Test Support****TOOLS****Eclipse Modeling  
Eclipse Xtext**



What if I don't yet  
have a language?



Actually, this is the  
normal case!  
Domain Specific Language

# Building Languages

---



**As you  
understand  
the  
domain...**

**...develop a language to  
express it!**



**Language resembles  
domain concepts**



**Then express the  
design with  
the language.**



**Clear understanding of  
the domain from building  
the language**



# Iterate!



## Understand Domain

Domain Expert  
Language Engineer

## Define Language

Language Engineer

## Use Language

Domain Expert  
Domain User

# Iterate!



## Understand Domain

Domain Expert  
Language Engineer

## Define Language

**FORMAL!**

## Use Language

**FORMAL!**



## DSL Engineering Tools







The top slide features two logos. On the left is the Eclipse Modeling Project logo, which includes the text "eclipse modeling PROJECT" and a stylized planet with rings. To its right is the xtext logo, consisting of the word "xtext" in a bold, lowercase font with a stylized 'x'.

**Open Source  
Eclipse Public License**

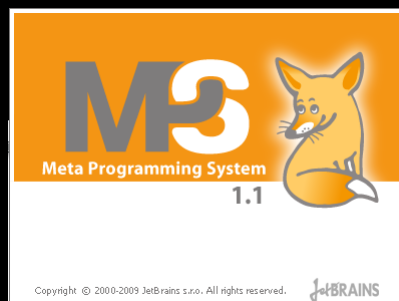


The bottom slide features the same two logos as the top slide: the Eclipse Modeling Project logo and the xtext logo.

**Large world wide  
community**



**graphical, textual and  
form-based DSLs**



**Developed by JetBrains  
Open Source  
Apache 2.0**



**Projectional Editor**  
**all kinds of notations,**  
**mainly textual**



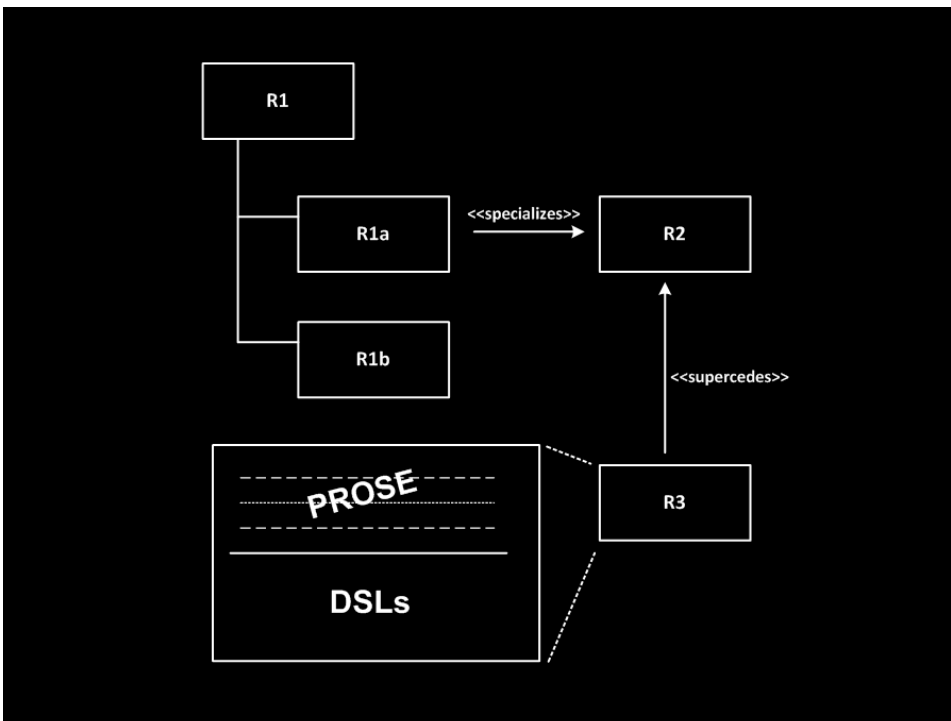
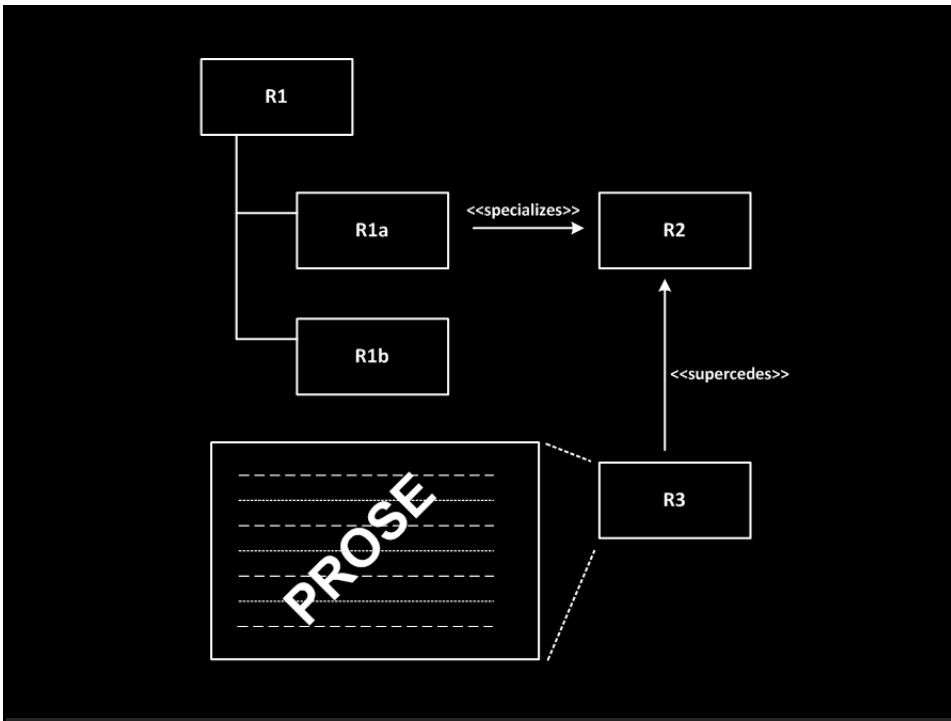
**Commercial Product**  
**Projectional Editor**  
**Most flexible notations**

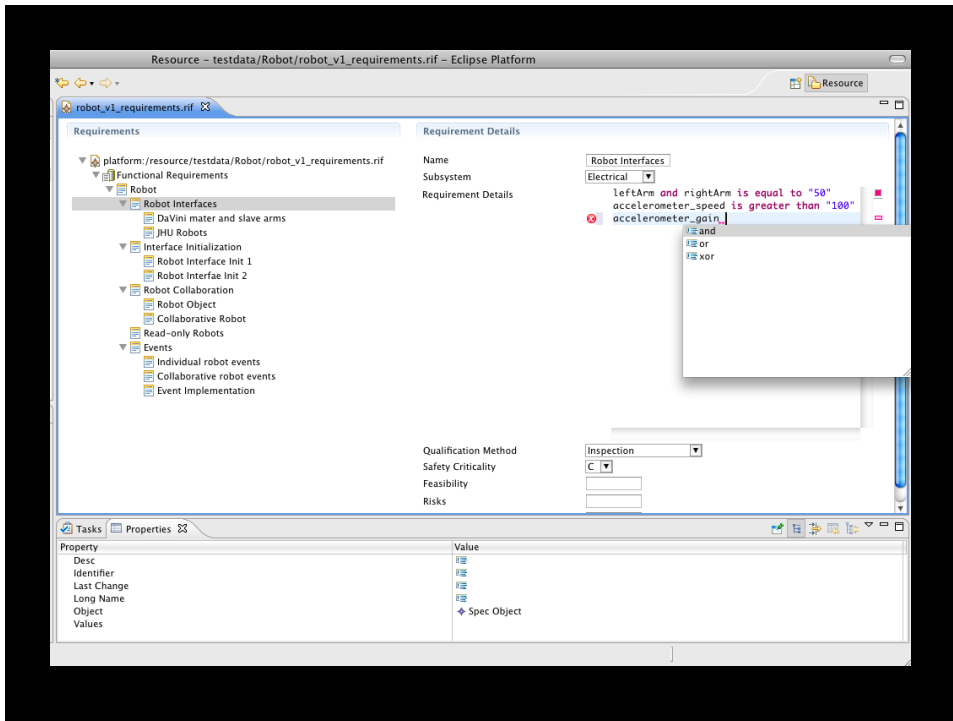
## Textual Requirements?



## Textual Requirements

still necessary.






# Tracing

```

DummyRequirementsCollection x LineFollower x
trace Cyclic
do 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 = ecreobot_get_touch_sensor(SENSOR_PORT_T::NXT_PORT_S3);
    if ( bump == 1 ) {
      event linefollower:bumped
      terminate;
    }
    trace Init
    int32 light = 0;
    light = ecreobot_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>;
}

```






Cohesive  
 Complete  
**Consistent**  
 Atomic  
 Traceable  
 Current  
 Feasible  
 Unambiguous  
 Mandatory  
 Verifiable

Validation  
 Checking  
 Simulation

Wikipedia




Cohesive  
 Complete  
 Consistent  
 Atomic  
**Traceable**  
 Current  
 Feasible  
 Unambiguous  
 Mandatory  
 Verifiable

Everything is a model  
 tracing links simple


Wikipedia






Cohesive  
Complete  
Consistent  
Atomic  
Traceable  
Current  
Feasible  
**Unambiguous** Described Formally  
Mandatory  
Verifiable

Wikipedia



Cohesive  
Complete  
Consistent  
Atomic  
Traceable  
Current  
Feasible  
Unambiguous  
Mandatory  
**Verifiable** Domain Expert involved  
in Definition and Review

Wikipedia



Cohesive  
 Complete  
 Consistent  
 Atomic  
 Traceable  
 Current  
 Feasible  
 Unambiguous  
 Mandatory  
 Verifiable  
**Executable** Automatic Refinement  
 downstream, Code Gen.



Cohesive  
 Complete  
**Consistent**  
 Atomic  
**Traceable**  
 Current  
 Feasible  
**Unambiguous**  
 Mandatory  
**Verifiable**  
**Executable**

**Reward**  
 for the  
 additional effort  
 of formalization!

## And Developers???

- ... Languages
- ... Technology Evaluation
- ... Generators
- ... Testing
- ... Operations
- ... what they want to do anyway!



**Brain** [Domain Person]

**DSL**



**Brain** [Developer]

**Code**



# Getting Started

---



# Getting Started

---

- 1** Familiarize yourself with some of the tools to better understand their capabilities

## Getting Started

---

- 2 Identify a limited, but meaningful aspect of your system and build a prototype**

## Getting Started

---

- 3 Evaluate lessons learned (technical)**

## Getting Started

---

- 4** Think about „more strategic“ places where DSLs can be used, and consider process consequences

## Getting Started

---

- 5** Repeat forever :-)

# THE END.



## .coordinates

web [www.voelter.de](http://www.voelter.de)  
email [voelter@acm.org](mailto:voelter@acm.org)  
skype [schogglad](#)

xing [http://www.xing.com/profile/Markus\\_Voelter](http://www.xing.com/profile/Markus_Voelter)  
linkedin <http://www.linkedin.com/pub/0/377/a31>

itemis