Domain Specific Languages
and
Requirements (Engineering)

itemis

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

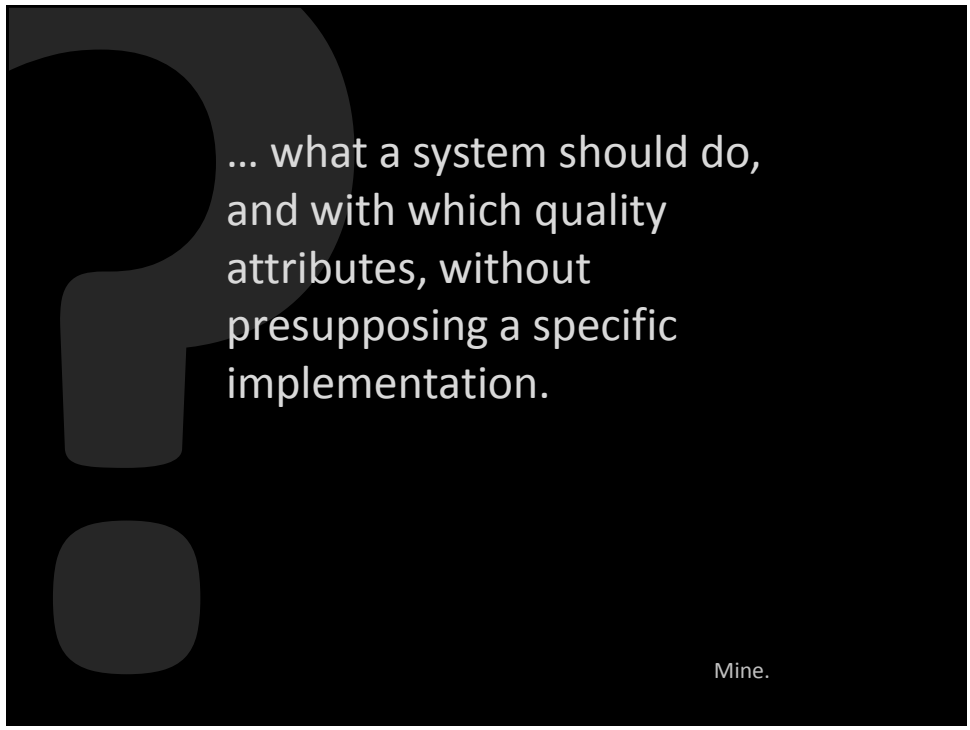
What are Requirements?

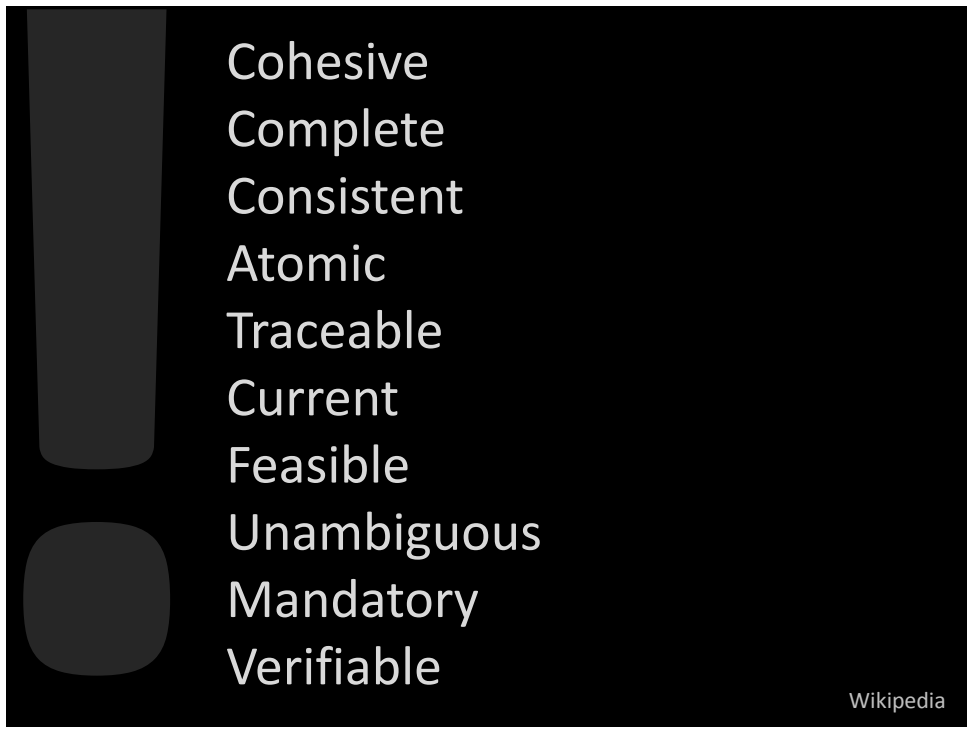
… a requirement is a singular documented need of what a particular product or service should be or perform.

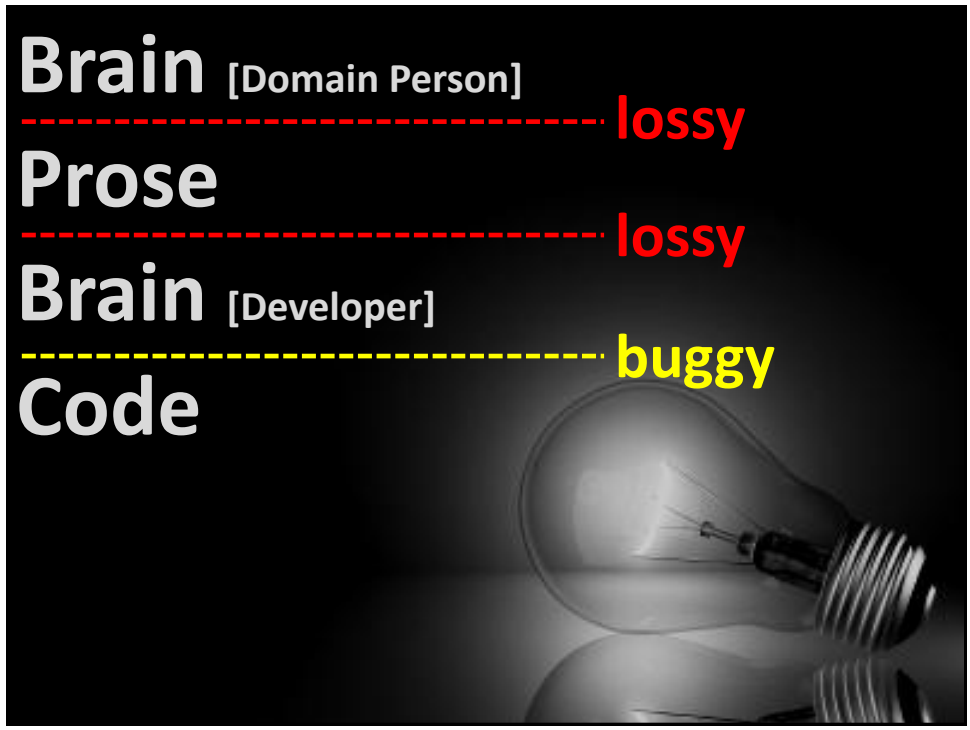
Wikipedia

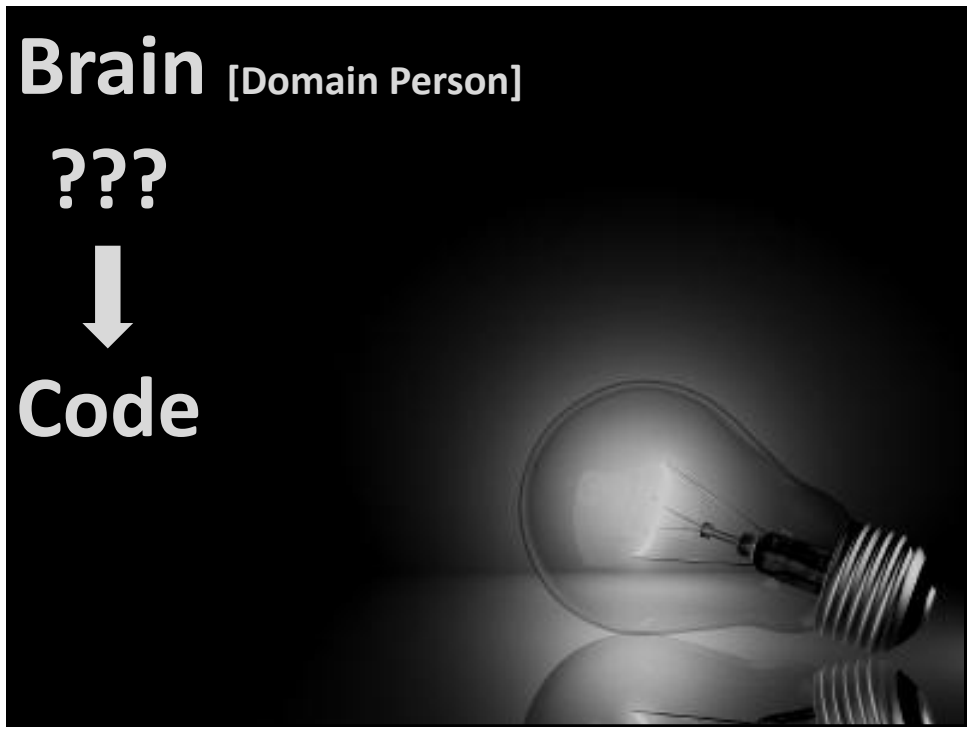
… specifies a verifiable constraint on an implementation that it shall undeniably meet or
(a) be deemed unacceptable, or
(b) result in implementation failure, or
(c) result in system failure.

Wiktionary

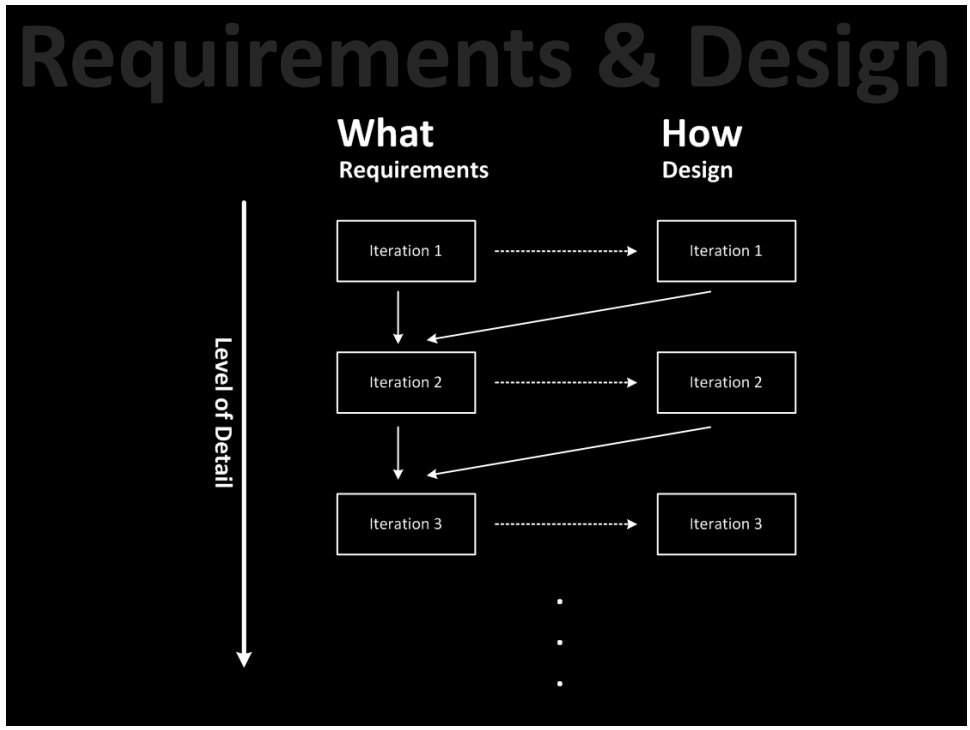

… what a system should do, and with which quality attributes, without presupposing a specific implementation.
Mine.


Cohesive
Complete
Consistent
Atomic
Traceable
Current
Feasible
Unambiguous
Mandatory
Verifiable
Wikipedia


Brain [Domain Person]
lossy
Prose
lossy
Brain [Developer]
buggy
Code


Brain [Domain Person]
???
Code

Requirements & Design



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**.

Mine++

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.

general purpose

domain specific

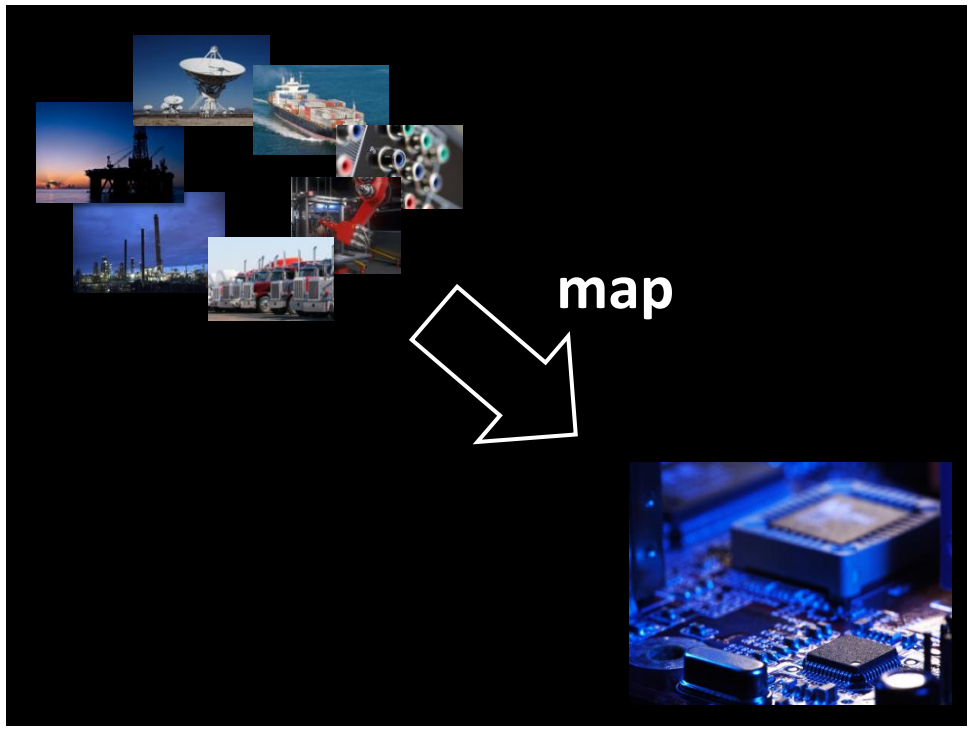
tailor made

effective++

specialized, limited

used by experts

together with other specialized tools



DSL Program
(aka Model)

map

automated!

GPL Program

map

Generation

Transformation
Compilation

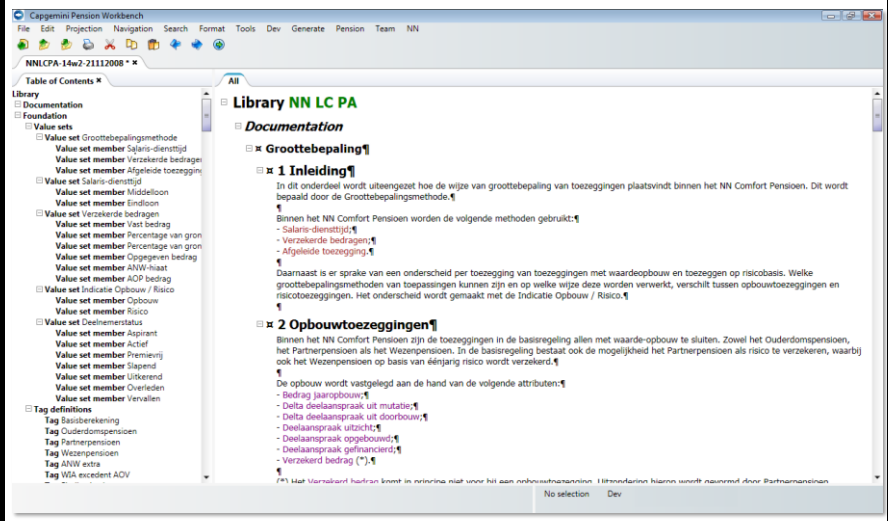
Interpretation

Example 1:

Pension Fund Specification

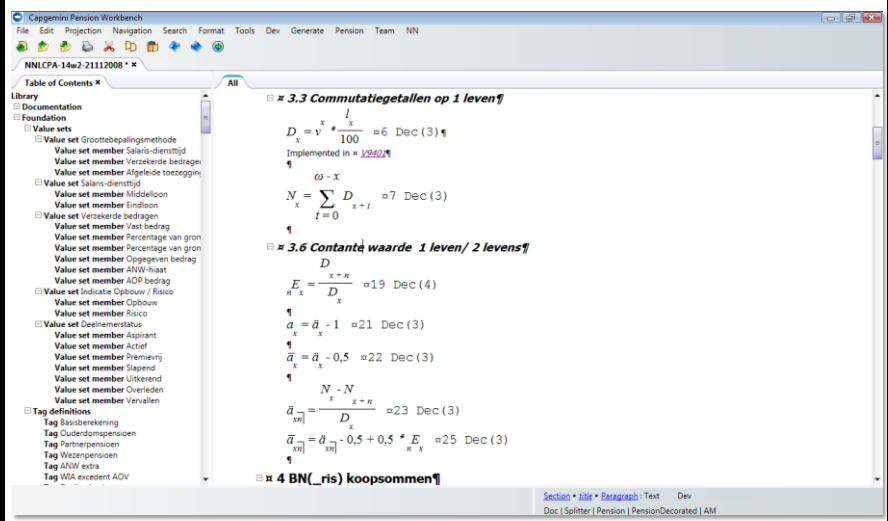
SOLUTION

Textual Documentation



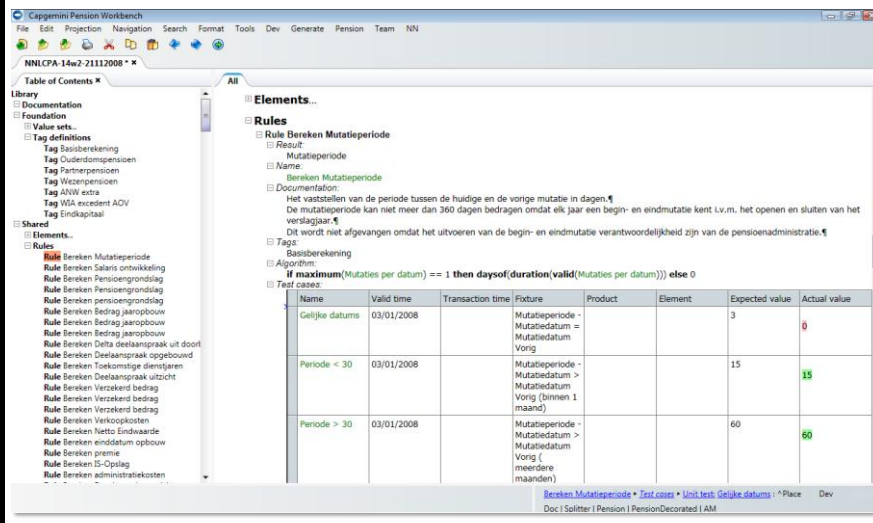
SOLUTION

Insurance Mathematics



SOLUTION

Calculation Rules and Tests



TOOLS

INTENTIONAL SOFTWARE

Intentional Software's Domain Workbench

Example 3:

Radar Systems Engineering

SOLUTION

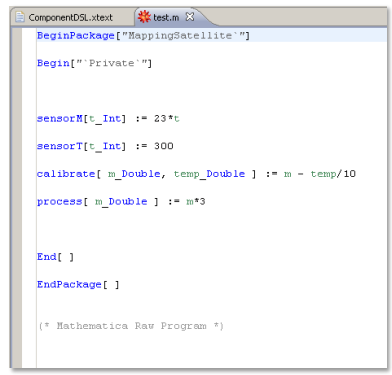
Component Definition

```
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.measureme
    }
}

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

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

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

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

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

}
```

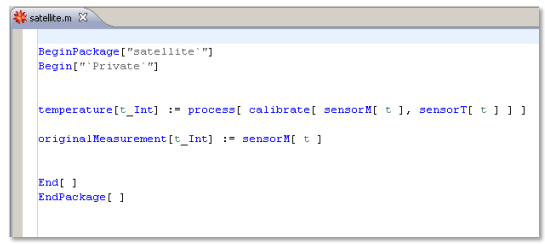
SOLUTION

Component Behavior Specification



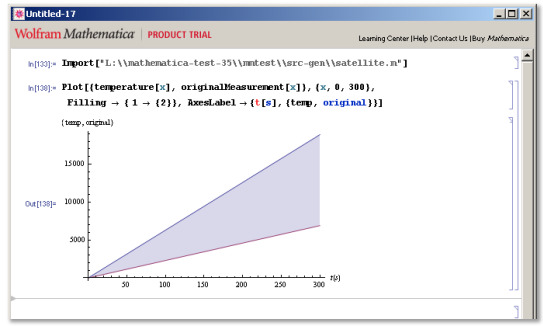
SOLUTION

Resulting System Behaviour

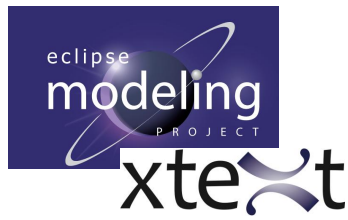


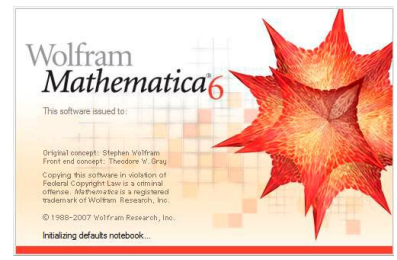
SOLUTION

Analysis



TOOLS





Eclipse Modeling
Eclipse Xtext
Wolfram Mathematica
Mathematica Workbench

Example 2:
Fountains

CONTEXT

SOLUTION

Hardware Structure

```
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 a1

fountain StdFountain extends AtLeastOneZone
```

SOLUTION

Behaviour

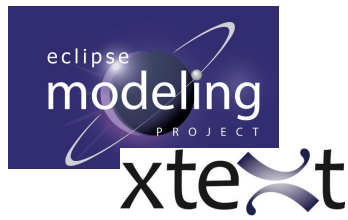
```
pumping program P1 for AtLeastOneZone + WithAlarm +
                       SuperPowerCompartment[f=comp1]  {
    parameter defaultWaterLevel : int
    parameter superWaterLevel: int
    event superPowerTimeout

    init {
        set comp1->targetHeight = defaultWaterLevel
    }

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

SOLUTION

Plus:

In-IDE Simulator
Unit Test Support

TOOLS



Eclipse Modeling
Eclipse Xtext

What if I don‘t yet have a language?

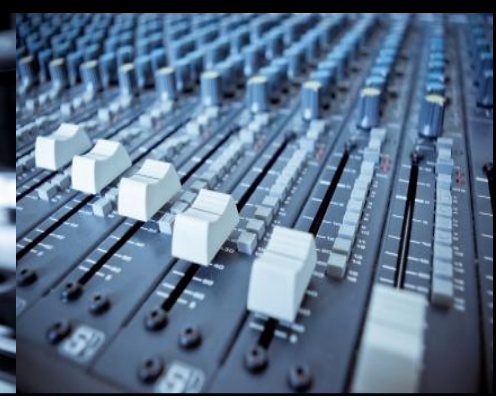
Actually, this is the normal case!
Domain Specific Language

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

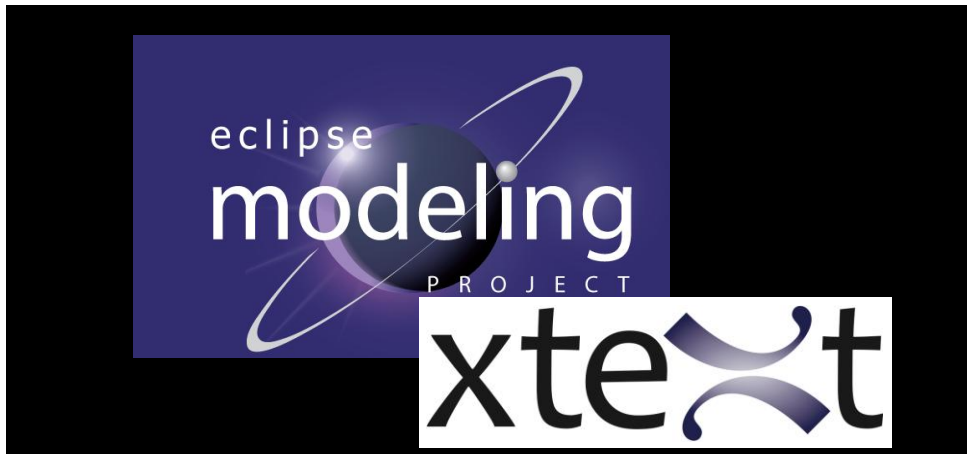
Iterate!

Understand Domain
- Domain Expert
- Language Engineer

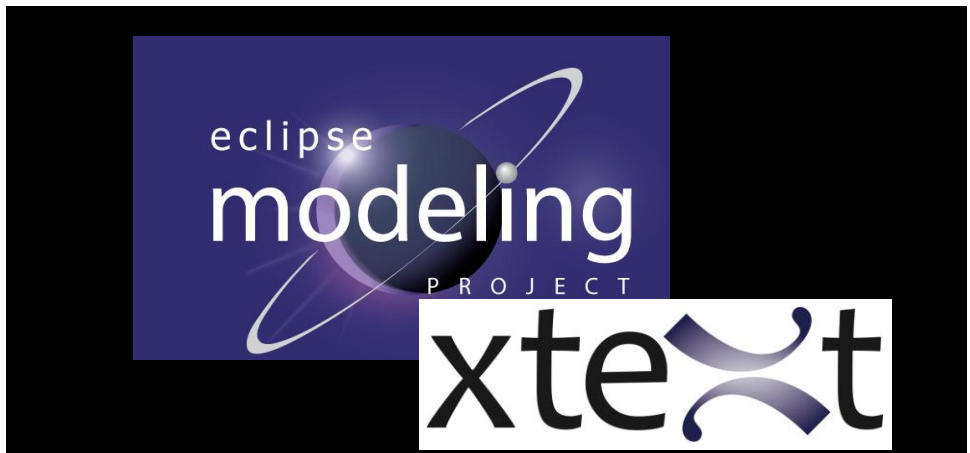
Define Language
- FORMAL!

Use Language
- FORMAL!

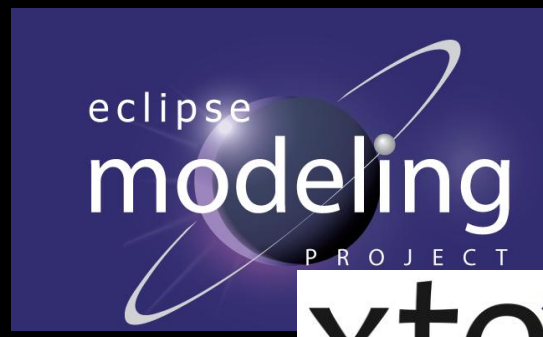
DSL Engineering Tools



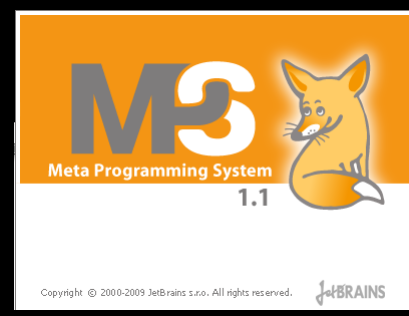
Open Source
Eclipse Public License



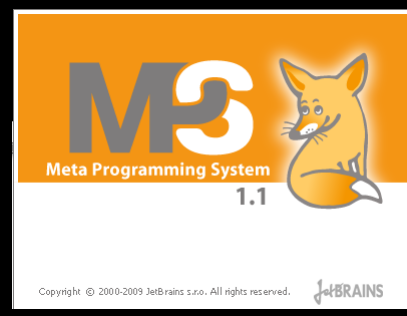
Large wold wide community



graphical, textual and form-based DSLs



Developed by JetBrains
Open Source
Apache 2.0



Projectional Editor
all kinds of notations, mainly textual

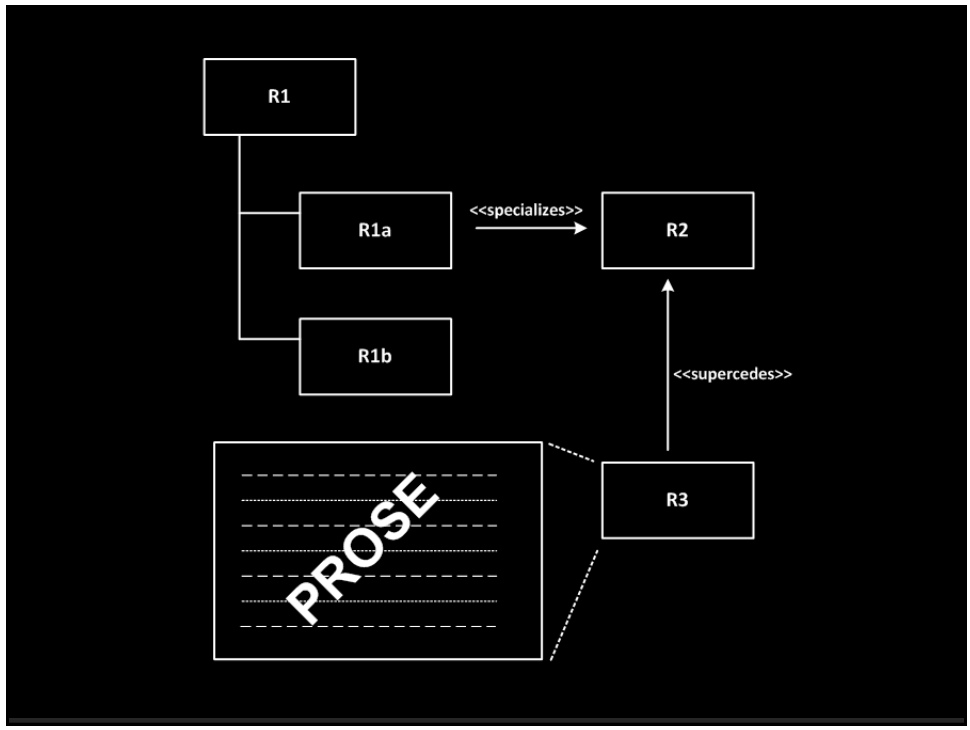
INTENTIONAL® SOFTWARE

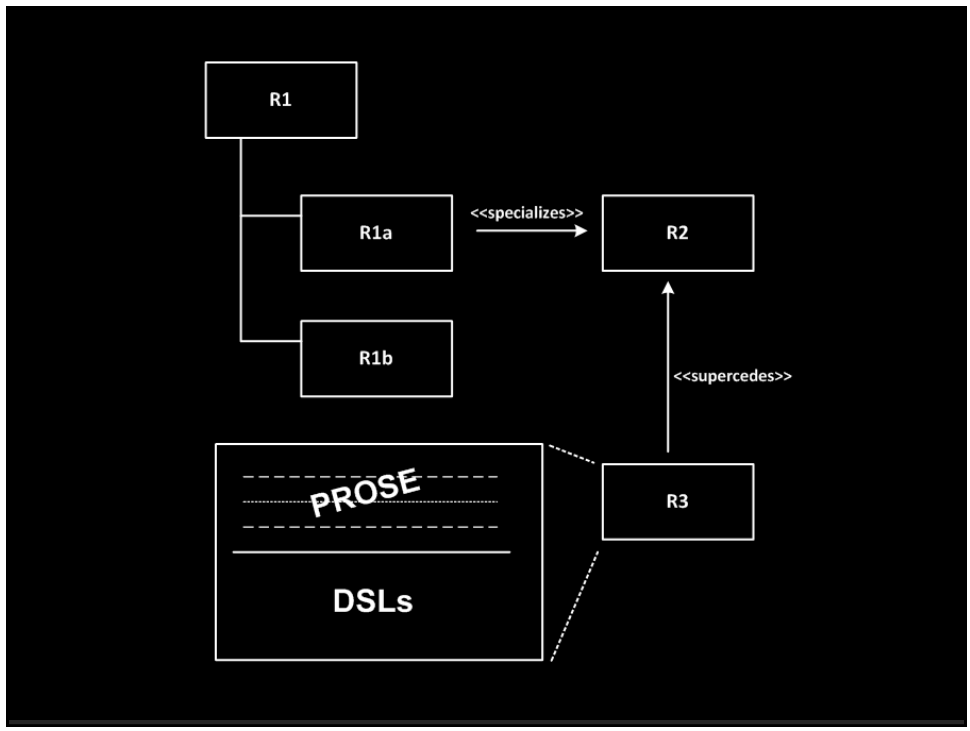
Commercial Product
Projectional Editor
Most flexible notations

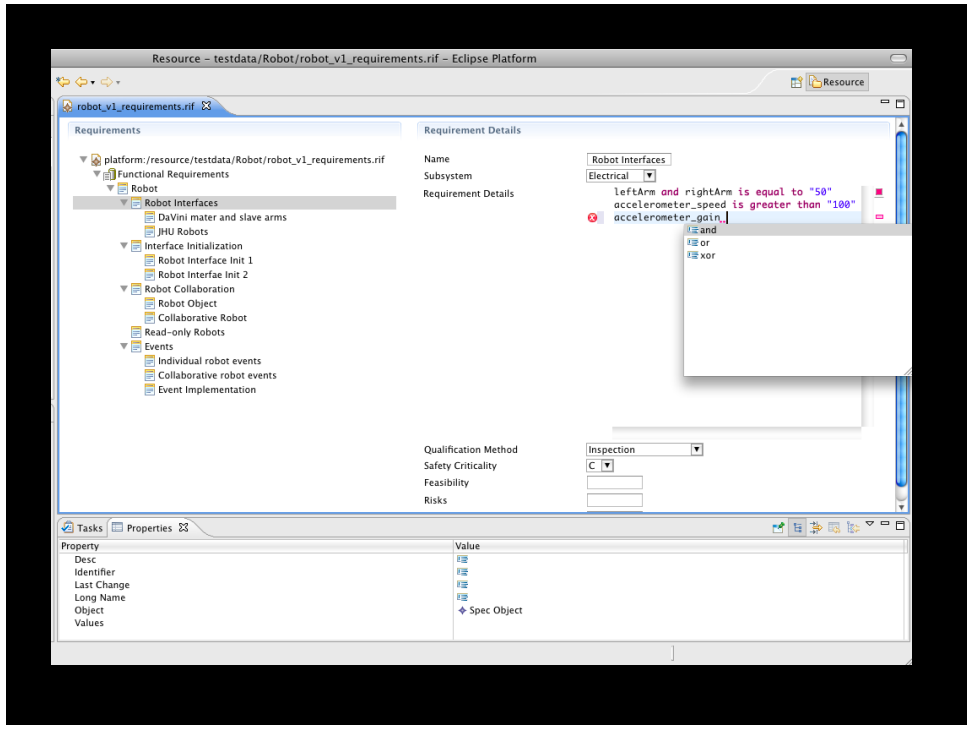
Textual Requirements?

Textual Requirements

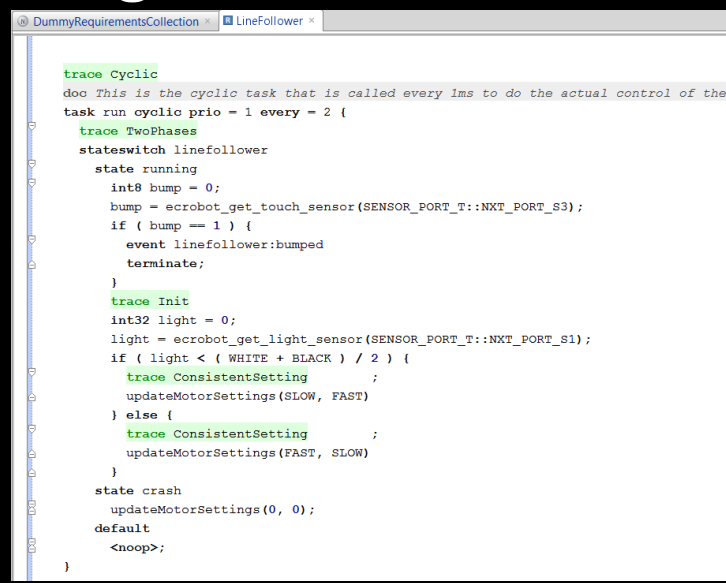
still necessary.


R1
R1a
<<specializes>>
R2
R1b
<<supercedes>>
PROSE
R3

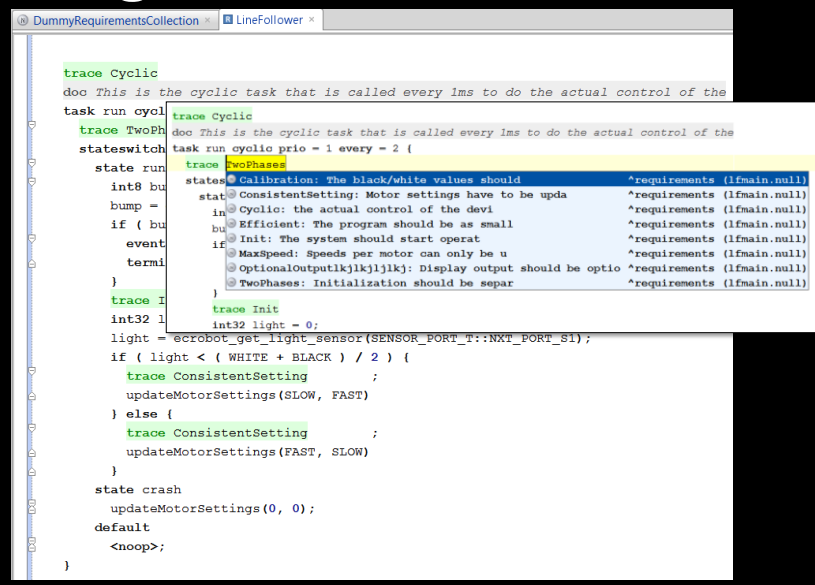

R1
R1a
<<specializes>>
R2
R1b
<<supercedes>>
PROSE
R3
DSLs


Resource – testdata/Robot/robot_v1_requirements.rif – Eclipse Platform
Resource
robot_v1_requirements.rif
Requirements
platform:/resource/testdata/Robot/robot_v1_requirements.rif
Functional Requirements
Robot
Robot Interfaces
DaVinci mater and slave arms
JHU Robots
Interface Initialization
Robot Interface Init 1
Robot Interfae Init 2
Robot Collaboration
Robot Object
Collaborative Robot
Read-only Robots
Events
Individual robot events
Collaborative robot events
Event Implementation
Requirement Details
Name
Robot Interfaces
Subsystem
Electrical
Requirement Details
leftArm and rightArm is equal to "50"
accelerometer_speed is greater than "100"
accelerometer_goin
and
or
xor
Qualification Method
Inspection
Safety Criticality
C
Feasibility
Risks
Tasks
Properties
Property
Value
Desc
Identifier
Last Change
Long Name
Object
Values
Spec Object

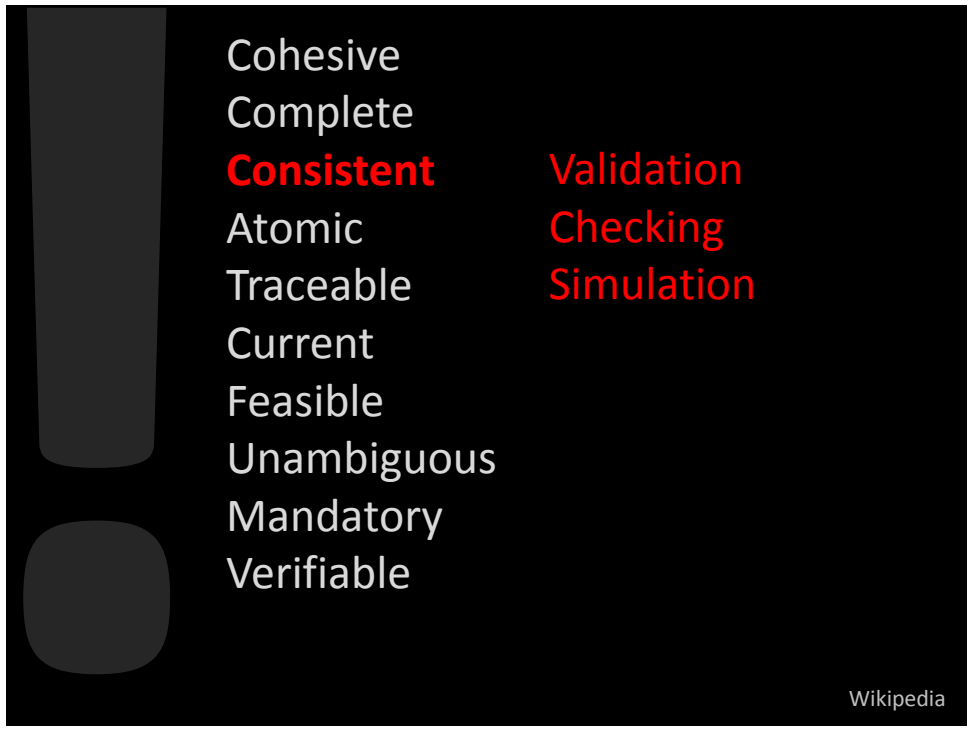
Tracing

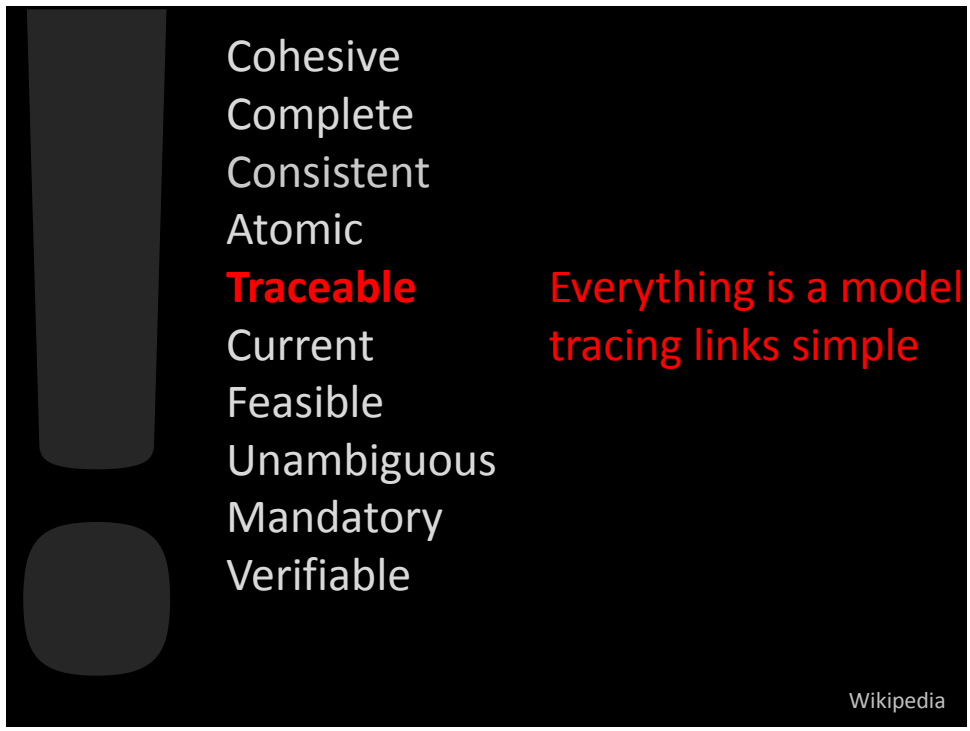

DummyRequirementsCollection
LineFollower
trace Cyclic
doc This is the cyclic task that is called every 1ms to do the actual control of the
task run cyclic prio = 1 every = 2 {
trace TwoPhases
stateswitch linefollower
state running
int8 bump = 0;
bump = ecrobot_get_touch_sensor(SENSOR_PORT_T::NXT_PORT_S3);
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>;
}

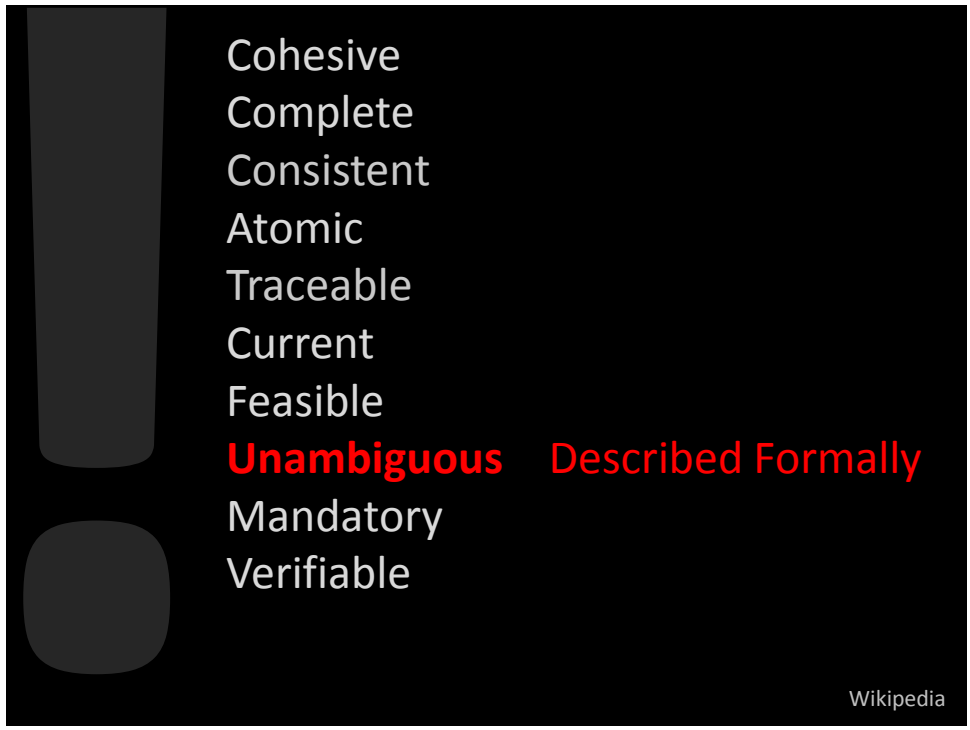
Tracing

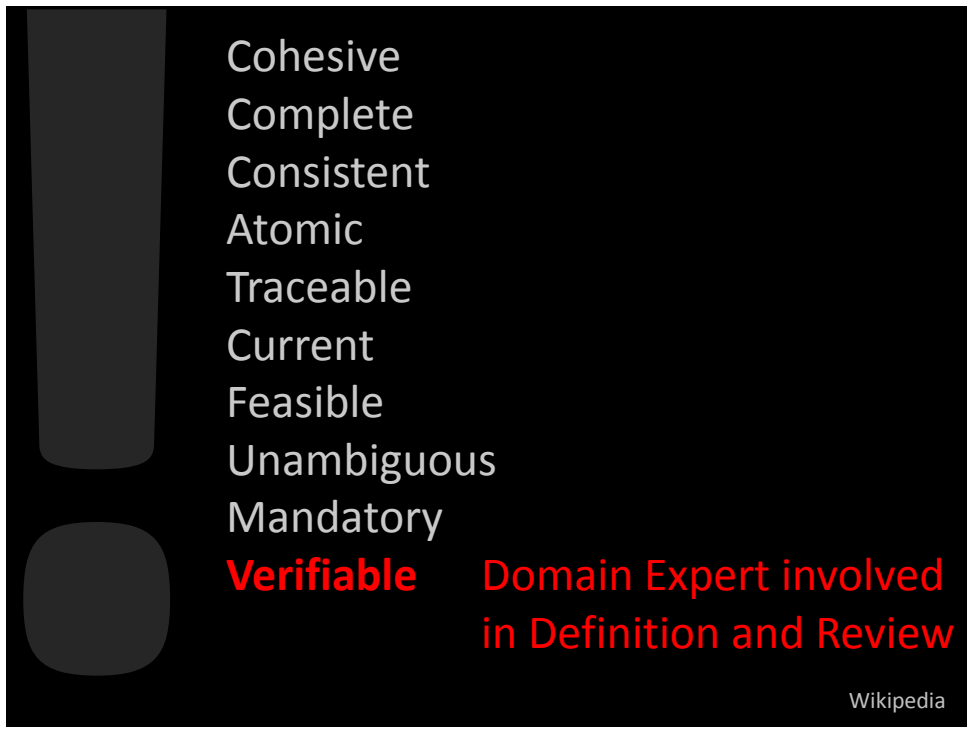


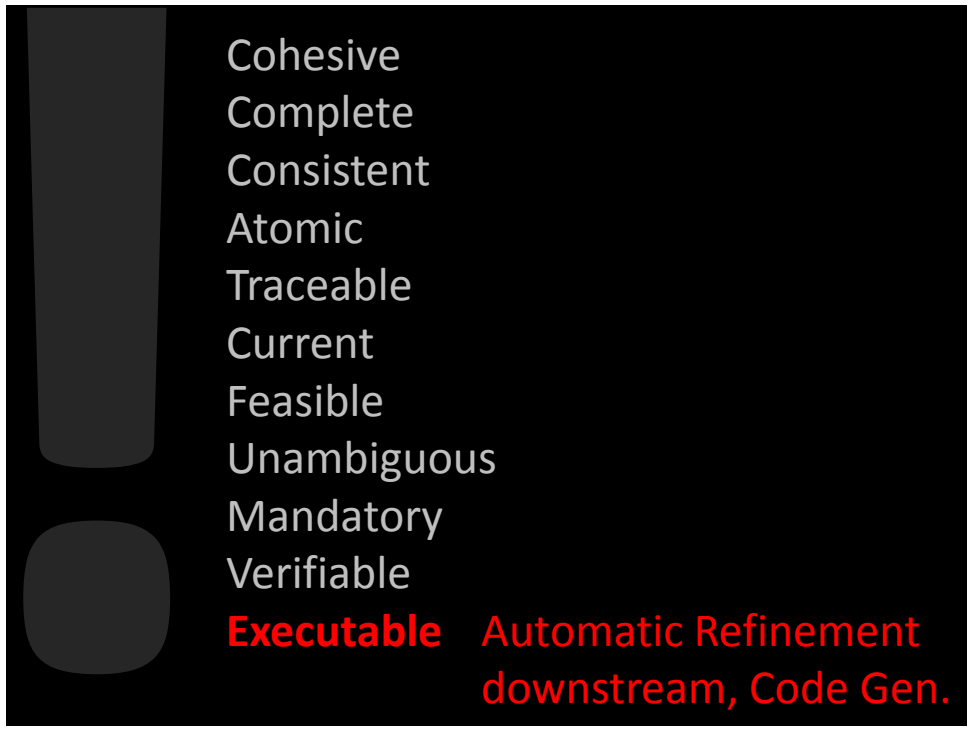
Summary


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 Evaulation
… 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
email	voelter@acm.org
skype	schogglad
xing	http://www.xing.com/profile/Markus_Voelter
linkedin	http://www.linkedin.com/pub/0/377/a31

itemis