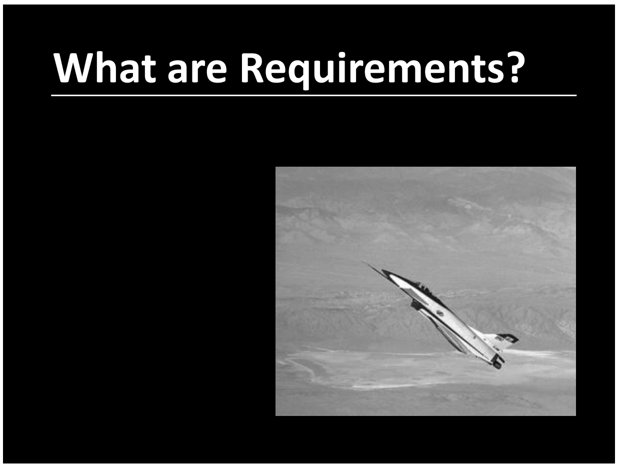
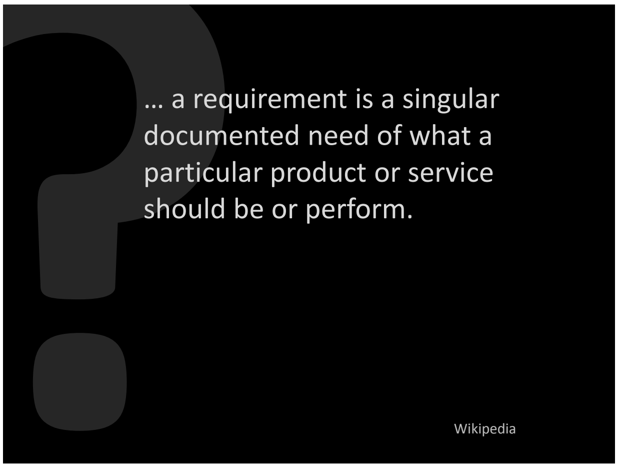
Domain Specific Languages
and
Requirements (Engineering)

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

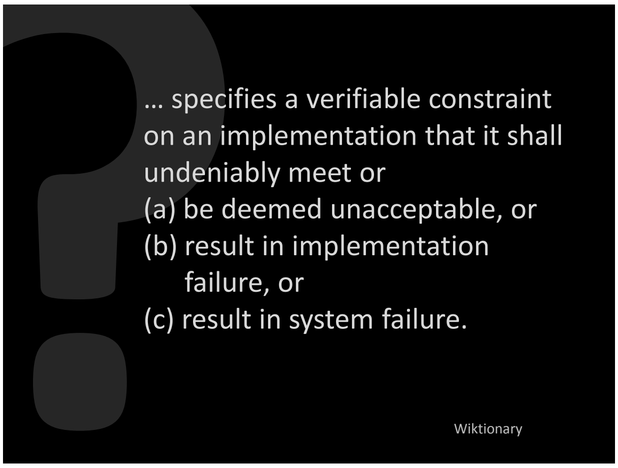


What are Requirements?



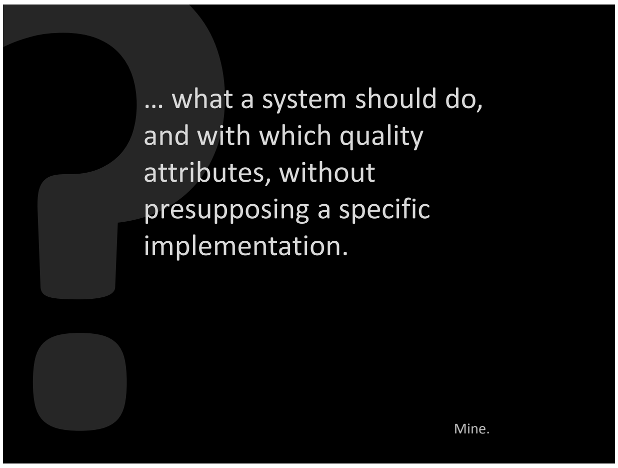
… a requirement is a singular documented need of what a particular product or service should be or perform.

Wikipedia



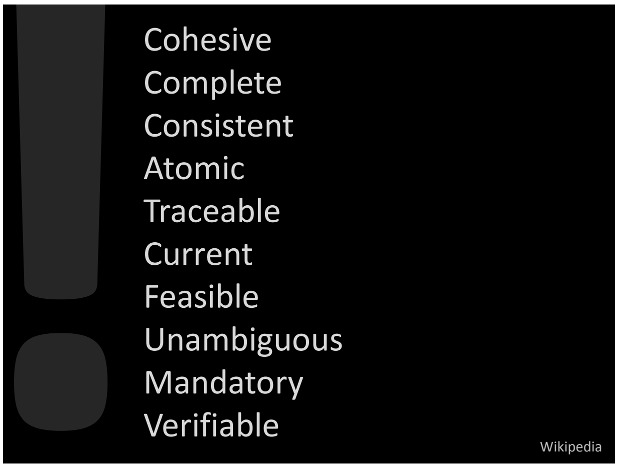
… specifies a verifiable constraint on an implementation that it shall undeniably meet or
(a) be deemed unacceptable, or
(b) result in implementation failure, or
(c) result in system failure.

Wiktionary



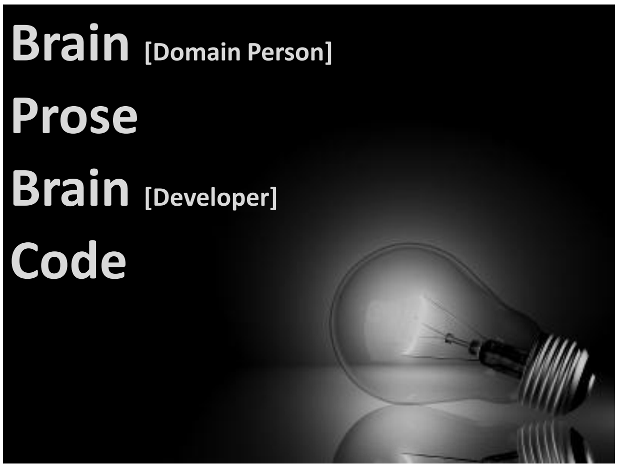
… what a system should do, and with which quality attributes, without presupposing a specific implementation.

Mine.



Cohesive
Complete
Consistent
Atomic
Traceable
Current
Feasible
Unambiguous
Mandatory
Verifiable

Wikipedia

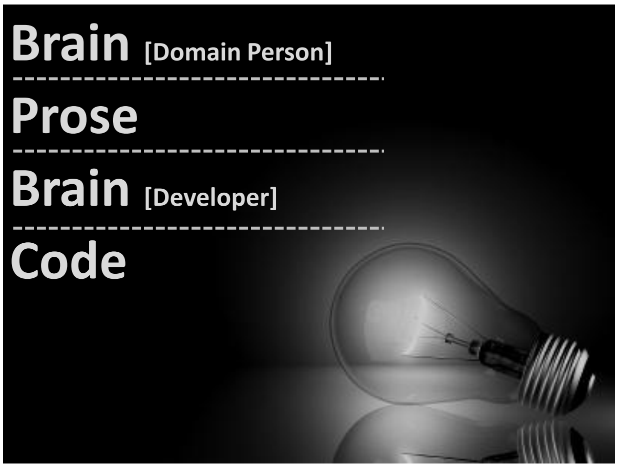


Brain [Domain Person]

Prose

Brain [Developer]

Code

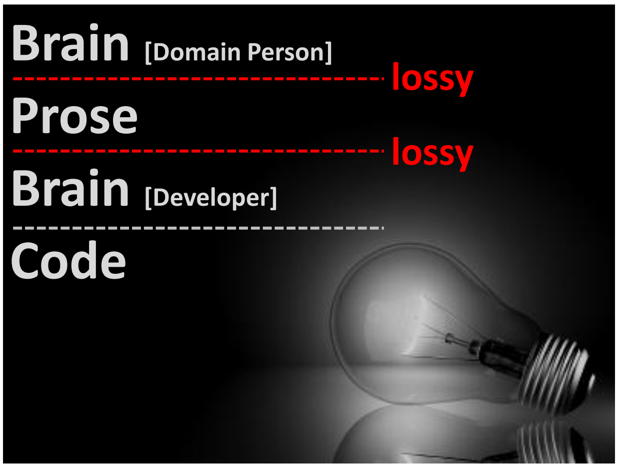


Brain [Domain Person]

Prose

Brain [Developer]

Code



Brain [Domain Person]

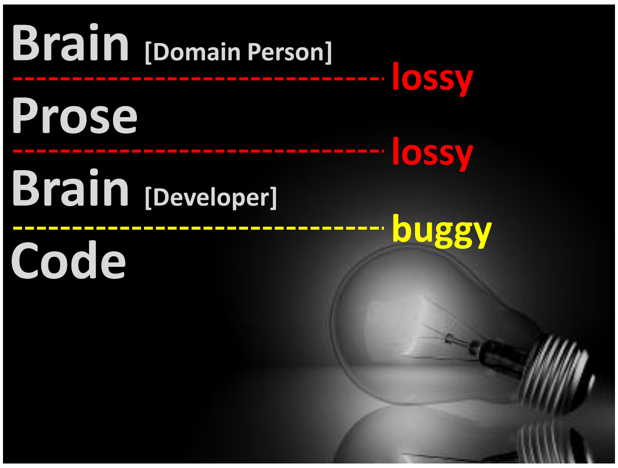
------------------------------- lossy

Prose

------------------------------- lossy

Brain [Developer]

Code



Brain [Domain Person]

lossy

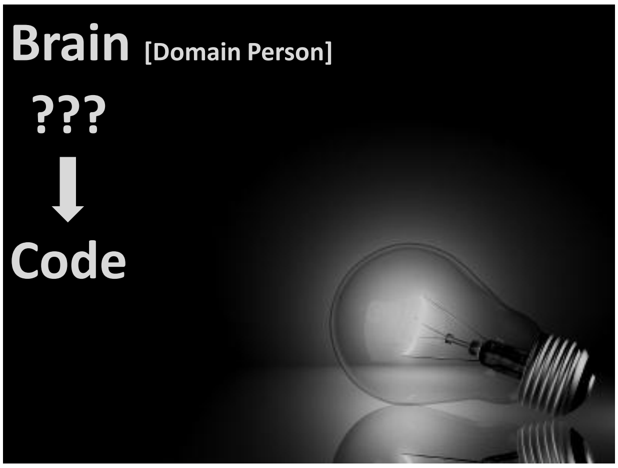
Prose

lossy

Brain [Developer]

buggy

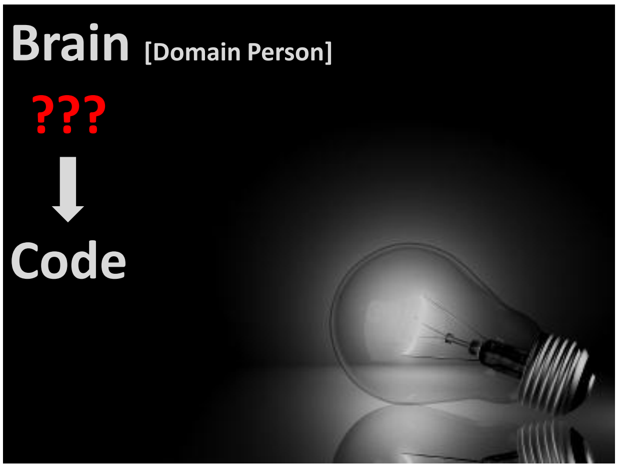
Code



Brain [Domain Person]

???

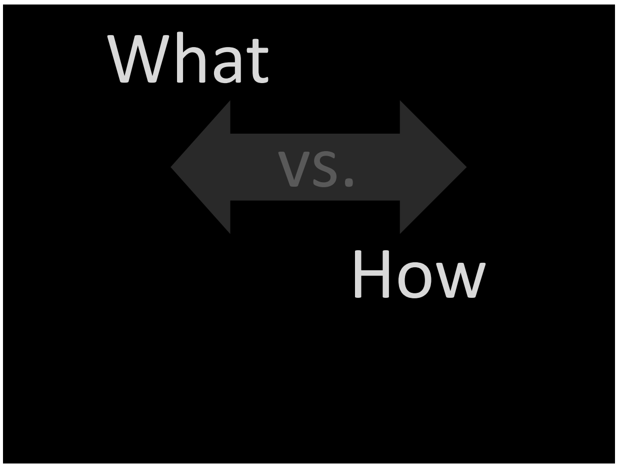
Code



Brain [Domain Person]

???

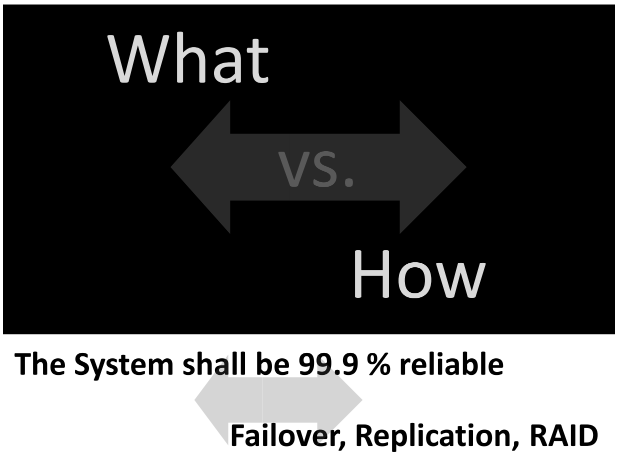
Code



What

vs.

How



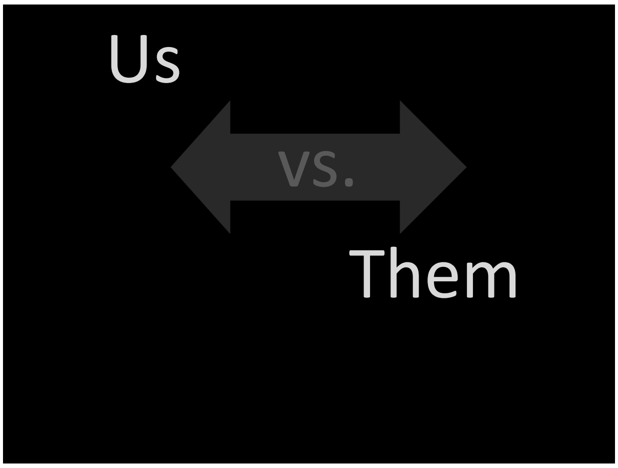
What

vs.

How

The System shall be 99.9 % reliable

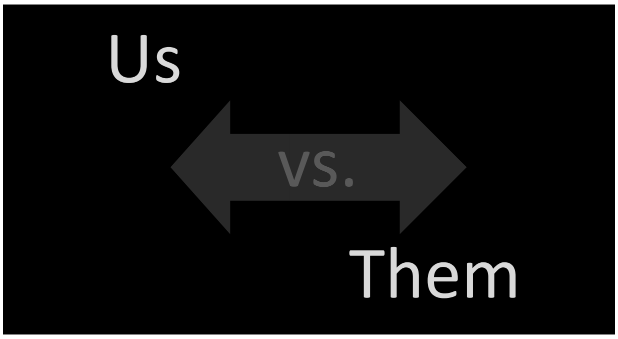
Failover, Replication, RAID



Us

vs.

Them



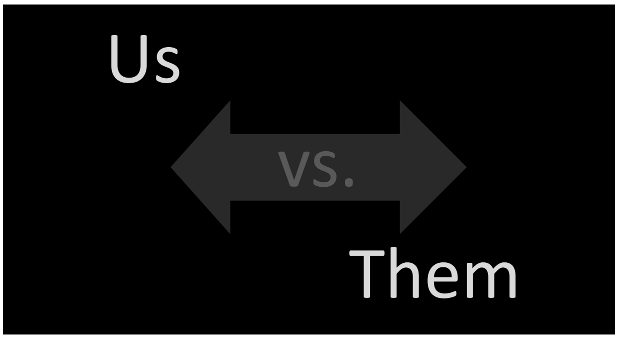
Us

vs.

Them

We specify the system…

…the offshore folks implement it



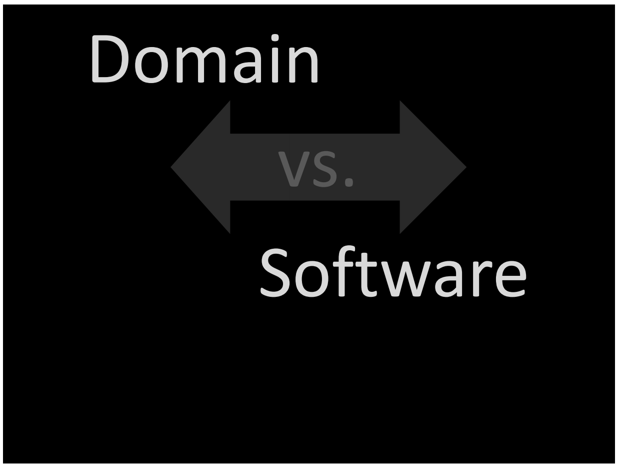
Us

vs.

Them

OEM specifies functions / interfaces …

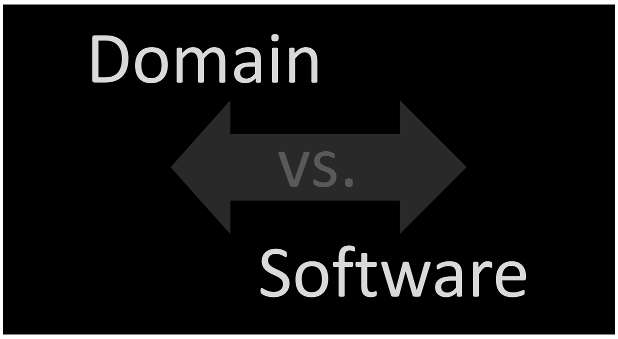
… the E/E vendor develops system



Domain

vs.

Software



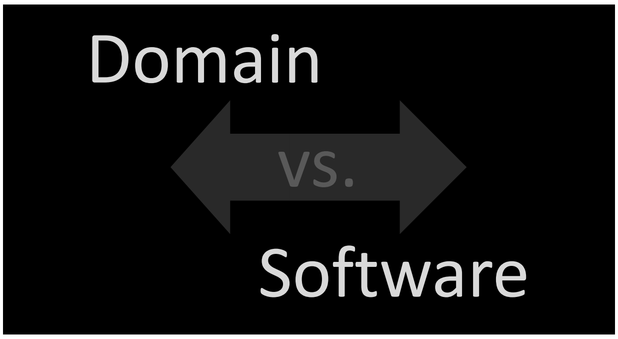
Domain

vs.

Software

Insurance contract rules…

… the actual realization as JEE app



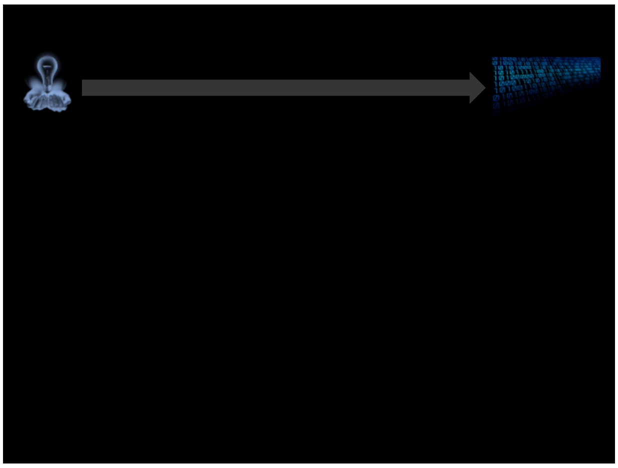
Domain

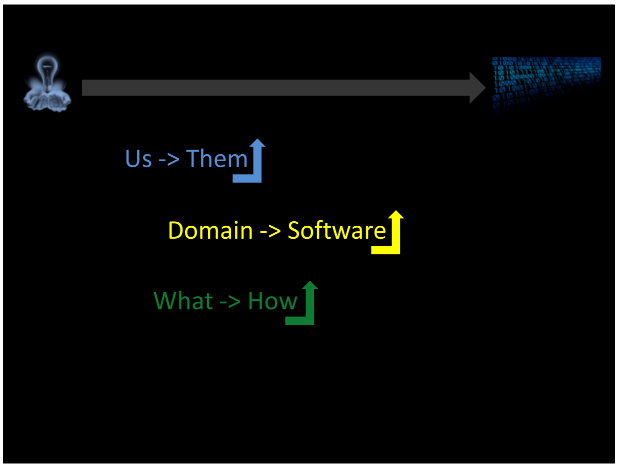
vs.

Software

Communication protocol spec

The protocol handler state machine

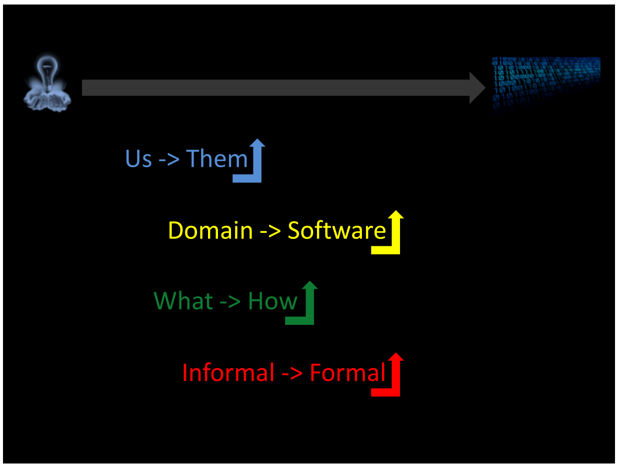




Us -> Them

Domain -> Software

What -> How

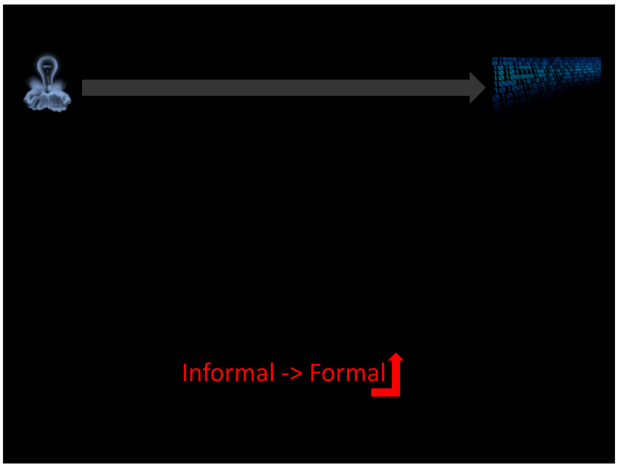


Us -> Them

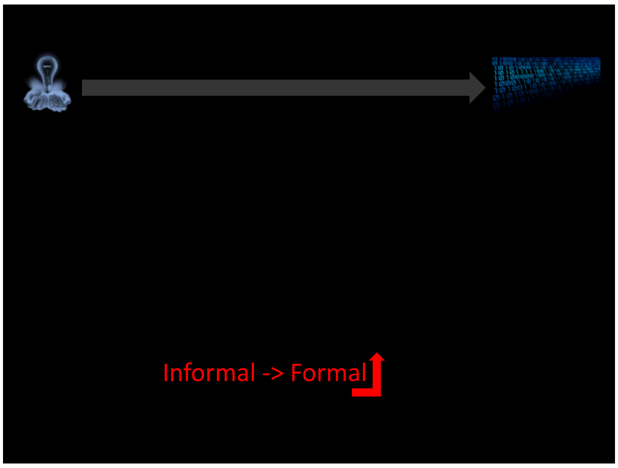
Domain -> Software

What -> How

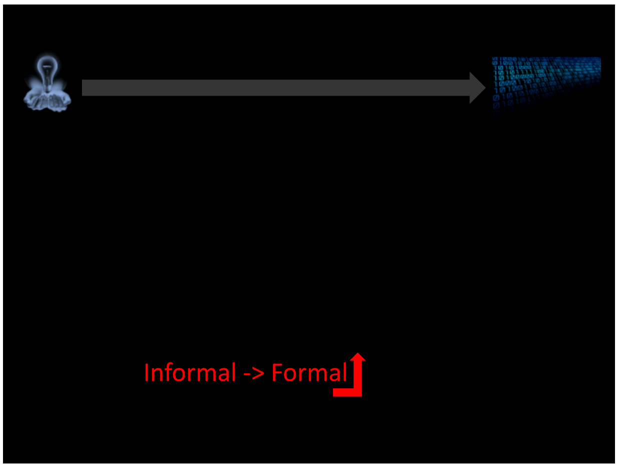
Informal -> Formal



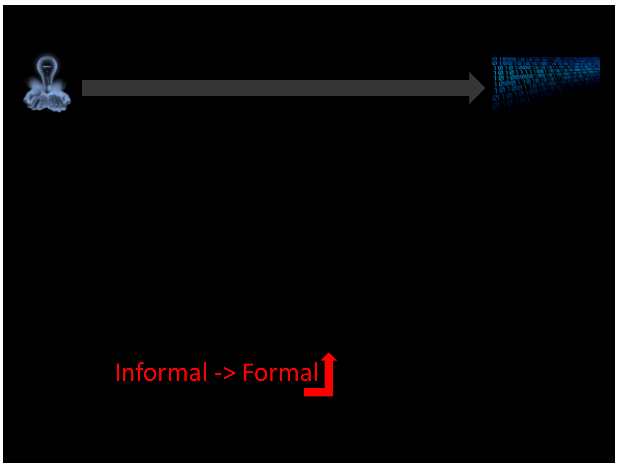
Informal -> Formal



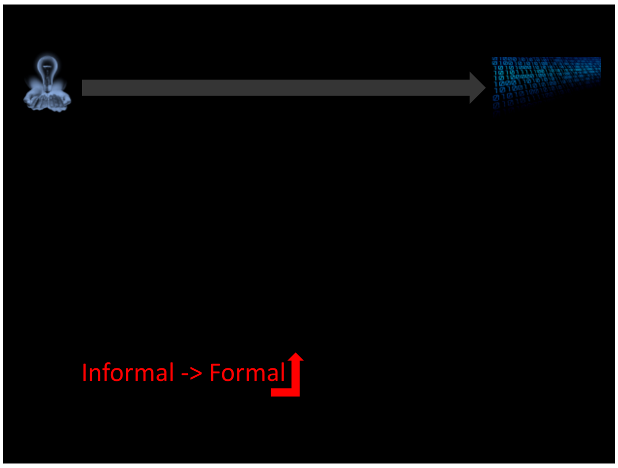
Informal -> Formal



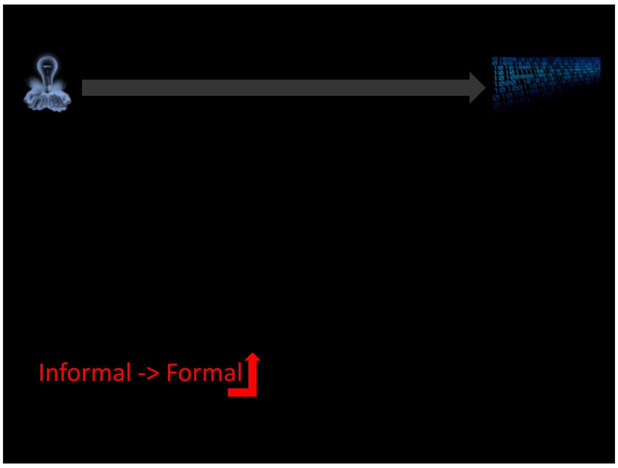
Informal -> Formal



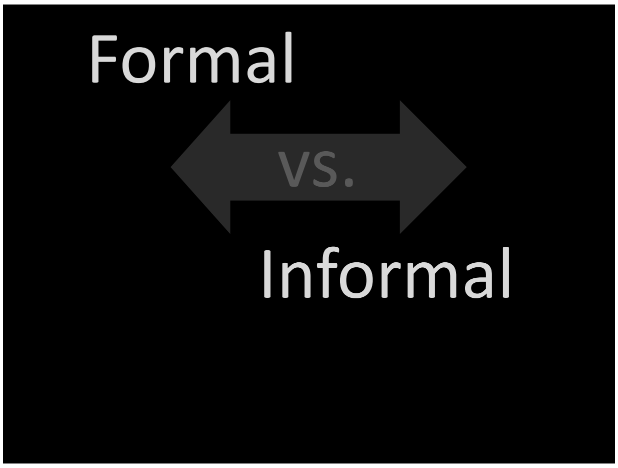
Informal -> Formal



Informal -> Formal



Informal -> Formal



Formal

vs.

Informal

Formal Requirements & Design

FORMAL

processable
by
tools

FORMAL

Cohesive
Complete
Consistent
Atomic
Traceable
Current
Feasible
Unambiguous
Mandatory
Verifiable

processable
by
tools

FORMAL

Cohesive
Complete
Consistent
Atomic
Traceable
Current
Feasible
Unambiguous
Mandatory
Verifiable

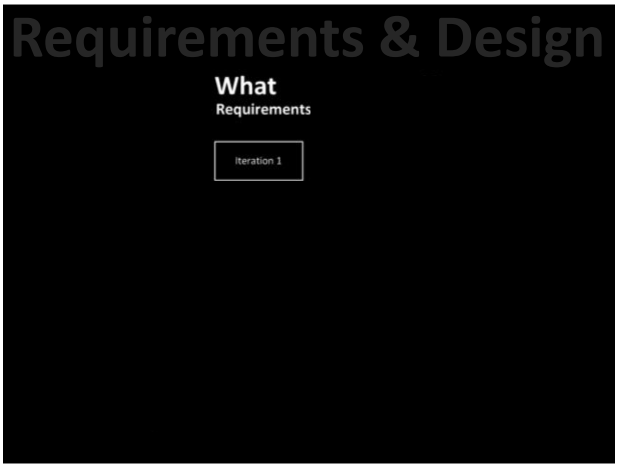
processable
by
tools

➡ **implementation**

Domain -> Software

What -> How

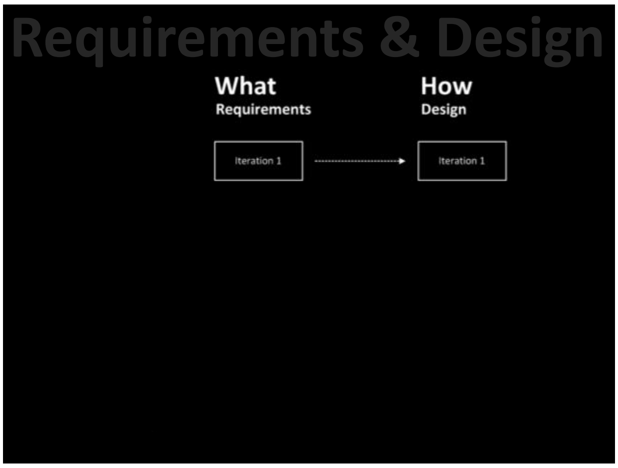
Informal -> Formal



Requirements & Design

What
Requirements

Iteration 1

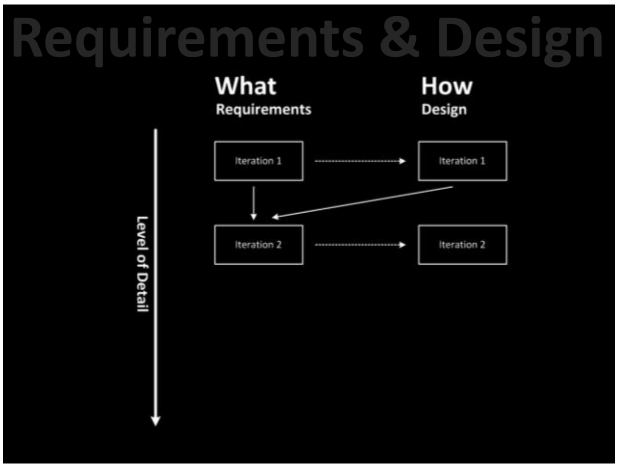


Requirements & Design

What
Requirements

How
Design

Iteration 1
Iteration 1



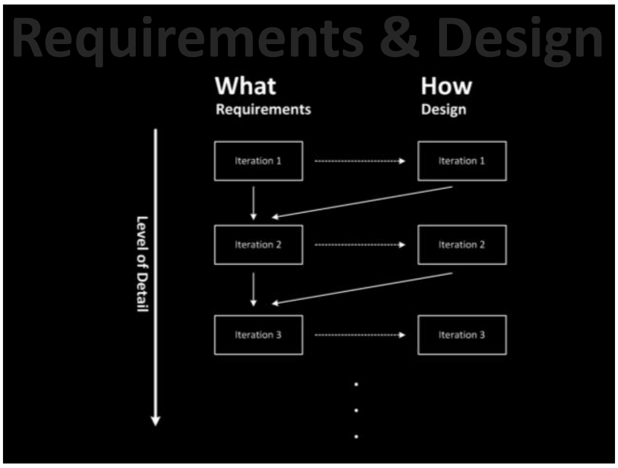
Requirements & Design

What
Requirements

How
Design

Level of Detail

Iteration 1
Iteration 1
Iteration 2
Iteration 2



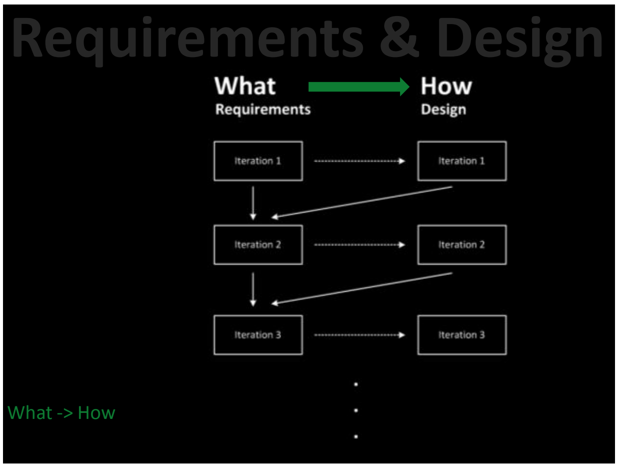
Requirements & Design

What
Requirements

How
Design

Level of Detail

Iteration 1
Iteration 1
Iteration 2
Iteration 2
Iteration 3
Iteration 3



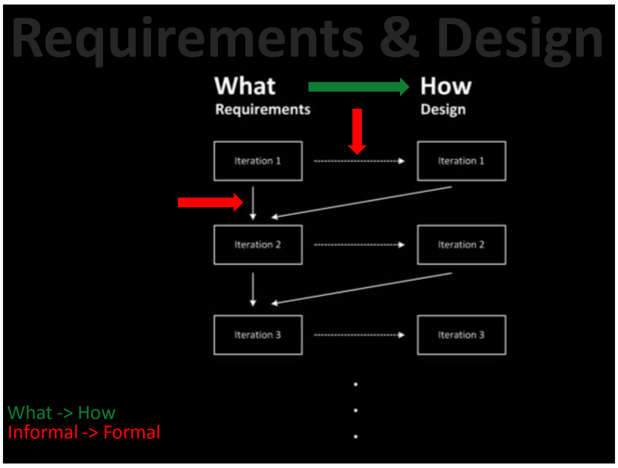
Requirements & Design

What
Requirements

How
Design

Iteration 1
Iteration 1
Iteration 2
Iteration 2
Iteration 3
Iteration 3

What -> How



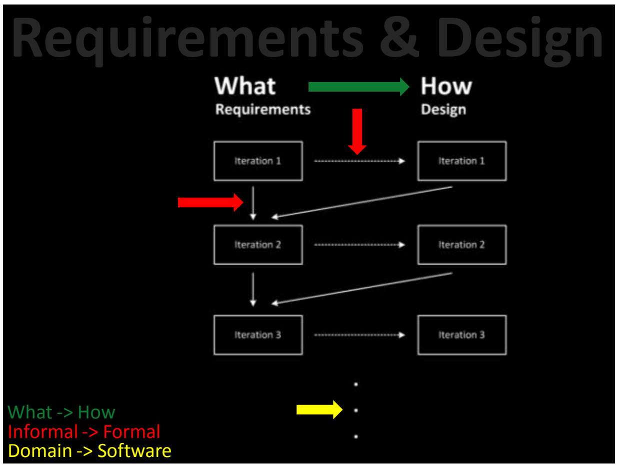
Requirements & Design

What
Requirements

How
Design

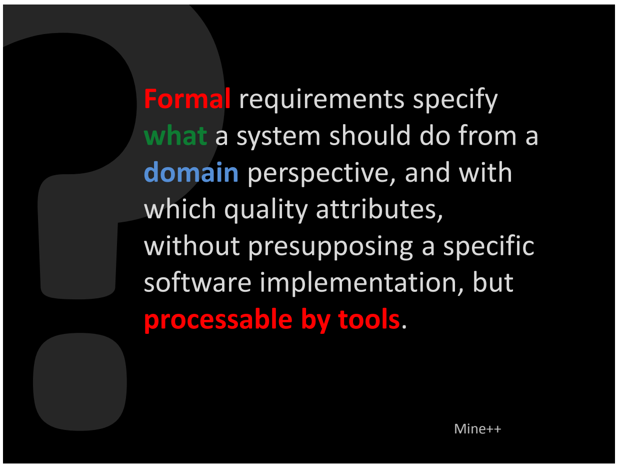
Iteration 1
Iteration 1
Iteration 2
Iteration 2
Iteration 3
Iteration 3

What -> How
Informal -> Formal



Requirements & Design

What -> How
Informal -> Formal
Domain -> Software



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++

Requirement/Informal:

It shall not be possible to get radiated when operating a microwave.

Requirement/Informal:

It shall not be possible to get radiated when operating a microwave.

Design/Informal:

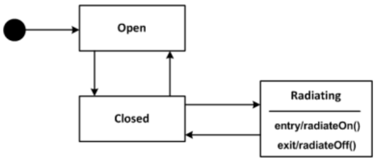
The radiator may only work iff the door of the microwave is closed.

+ door isolation
+ some quality requirements

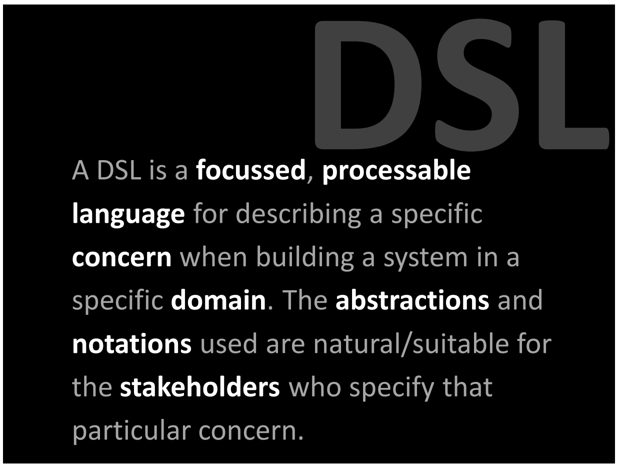
Requirement/Informal:

It shall not be possible to get radiated when operating a microwave.

Design/Formal:



Domain Specific Languages



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

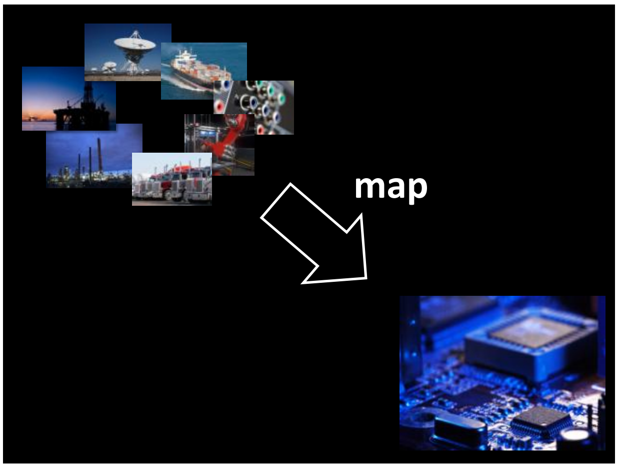
effective++

specialized, limited

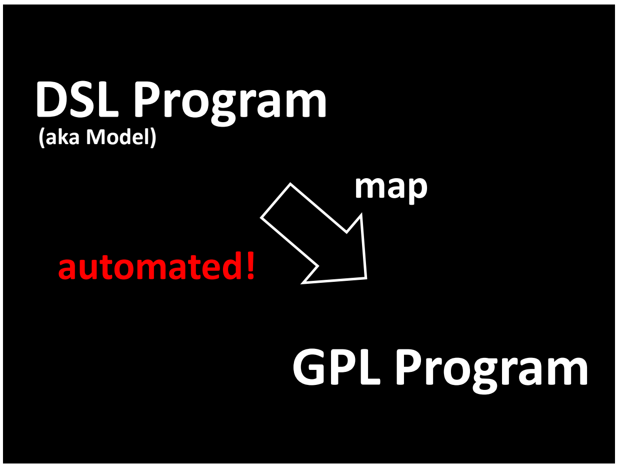
used by experts

together with other
specialized tools

execute?



map



DSL Program
(aka Model)

map

automated!

GPL Program

map

Generation

Transformation

Compilation

Interpretation

Example 1:

Embedded Protocol Handler

CONTEXT

Factory Control System

Plug-in Cards

Componentized System

PROBLEM

Cards have to communicate via predefined protocol

Protocol only specified as „plain text and pictures“

Verification tough, no automatic processing

SOLUTION

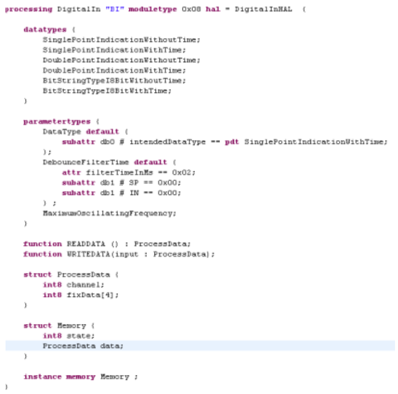
Define DSL for describing the protocol, formally define protocol with it

Generate Handler

Express test cases

SOLUTION

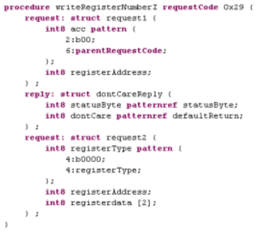
Component Specification



```
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 ;
}
```

SOLUTION

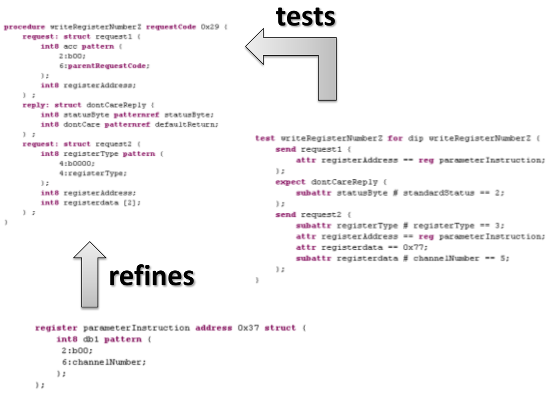
Message Format Definition



```
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];
    } ;
}
```

SOLUTION

Testing



```
procedure writeRegisterNumberZ requestCode 0x29 {
    request: struct request1 {
        int8 acc pattern {
            2:b00;
            6:parentRequestCode;
        );
        int8 registerAddress;
    } ;
    reply: struct dontCareReply {
        int8 statusByte patternref statusByte;
        int8 dontCare patternref defaultReturn;
    } ;
    request: struct request2 {
        int8 registerType pattern {
            4:b0000;
            4:registerType;
        );
        int8 registerAddress;
        int8 registerdata [2];
    } ;
}
```

```
test writeRegisterNumberZ for dip writeRegisterNumberZ {
    send request1 {
        attr registerAddress == reg parameterInstruction;
    );
    expect dontCareReply {
        subattr statusByte # standardStatus == 2;
    );
    send request2 {
        subattr registerType # registerType == 3;
        attr registerAddress == reg parameterInstruction;
        attr registerdata == 0x77;
        subattr registerdata # channelNumber == 5;
    );
}
```

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

BENEFITS

Testing could be simplified and automated

Handler could be generated

Mistakes in Spec could be found automatically

Second „version“ could be done trivially

TOOLS

Eclipse Modeling
Eclipse Xtext

Example 2:

Pension Fund Specification

CONTEXT

Insurance Company

Old-age Pension Funds

Lots of plans, based on how the laws change over time

Old plans must be kept around

PROBLEM

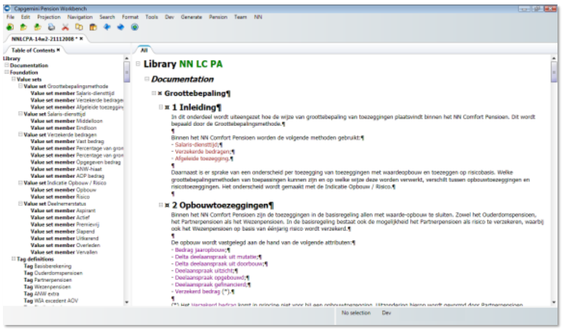
Time to implement/adapt new plans too long:
- Word documents,
- manual implementation

SOLUTION

Describe the complete plan and all calculation rules formally with a set of DSLs and then generate all of the calculation engine.

SOLUTION

Textual Documentation


Capgemini Pension Workbench
File Edit Projection Navigation Search Format Tools Dev Generate Pension Team NN
NNLCPA-14w2-21112008 *
Table of Contents
All
No selection Dev

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 ADP 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

Library NN LC PA

Documentation

¤ Groottebepaling¶

¤ 1 Inleiding¶

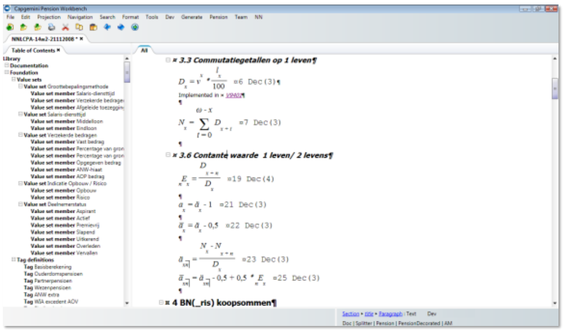
In dit onderdeel wordt uiteengezet hoe de wijze van groottebepaling van toezeggingen plaatsvindt binnen het NN Comfort Pensioen. Dit wordt bepaald door de Groottebepalingsmethode.¶
¶
Binnen het NN Comfort Pensioen worden de volgende methoden gebruikt:¶
- Salaris-diensttijd;¶
- Verzekerde bedragen;¶
- Afgeleide toezegging.¶
¶
Daarnaast is er sprake van een onderscheid per toezegging van toezeggingen met waardeopbouw en toezeggen op risicobasis. Welke groottebepalingsmethoden 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.¶
¶

¤ 2 Opbouwtoezeggingen¶

Binnen het NN Comfort Pensioen zijn de toezeggingen in de basisregeling allen met waarde-opbouw te sluiten. Zowel het Ouderdomspensioen, het Partnerpensioen als het Wezenpensioen. In de basisregeling bestaat ook de mogelijkheid het Partnerpensioen als risico te verzekeren, waarbij ook het Wezenpensioen 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 uitzicht;¶
- Deelaanspraak opgebouwd;¶
- Deelaanspraak gefinancierd;¶
- Verzekerd bedrag (*).¶
¶
(*) Het Verzekerd bedrag komt in principe niet voor bij een opbouwtoezegging. Uitzondering hierop wordt gevormd door Partnerpensioen

SOLUTION

Insurance Mathematics


Capgemini Pension Workbench
File Edit Projection Navigation Search Format Tools Dev Generate Pension Team NN
NNLCPA-14w2-21112008 *
Table of Contents
All
Section • title • Paragraph : Text Dev
Doc | Splitter | Pension | PensionDecorated | AM

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 ADP 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 ¤ VPA02¶
¶

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

¤ ***3.6 Contante waarde 1 leven/ 2 levens¶***

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

$a_x = \ddot{a}_x - 1$ ¤21 Dec(3)
¶

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

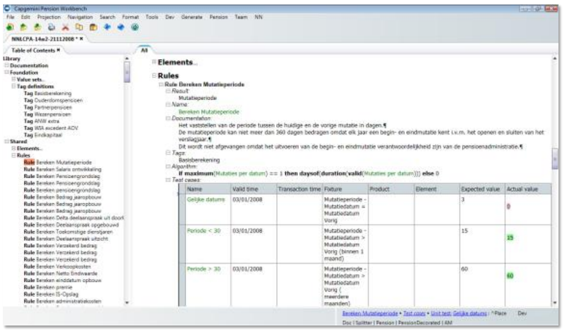
$\ddot{a}_{x\overline{n}|} = \frac{N_x - N_{x+n}}{D_x}$ ¤23 Dec(3)

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

¤ 4 BN(_ris) koopsommen¶

SOLUTION

Calculation Rules and Tests


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 doorl
    - 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 einddatum 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

BENEFITS

Domain users could implement complete plans themselves

Time-to-implement from 30 days to 3 days per plan

TOOLS

INTENTIONAL SOFTWARE

Intentional Software's Domain Workbench

Example 3:

Radar Systems Engineering

CONTEXT

Radar Systems for satellites need to be designed

Radar requirements influence sensor design influences satellite bus influences launch vehicle … circular.

PROBLEM

Requirements cannot simply be written down because tradeoffs studies need to be performed

These need to be numerical.

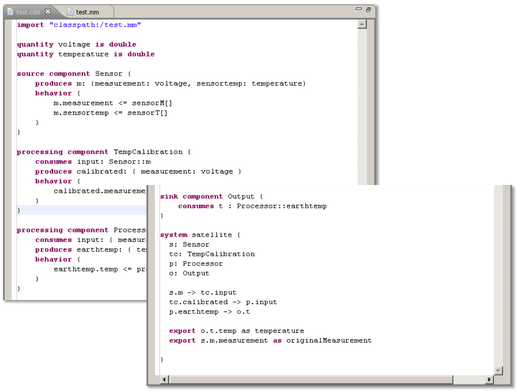
SOLUTION

Break system down into components

Use approximate numerical fomulae to how requirements and design effect other components

SOLUTION

Component Definition

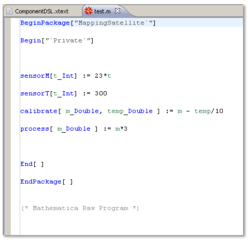


```
import "classpath:/test.mm"

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

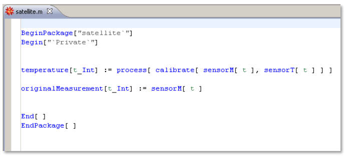
Component Behavior Specification



```
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 *)
```

SOLUTION

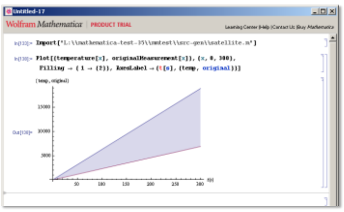
Resulting System Behaviour



```

BeginPackage["satellite`"]
Begin["`Private`"]


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

originalMeasurement[t_Int] := sensorM[ t ]


End[ ]
EndPackage[ ]
```

SOLUTION

Analysis



```
In[133]:= Import["L:\\mathematica-test-35\\mmtest\\src-gen\\satellite.m"]

In[130]:= Plot[{temperature[x], originalMeasurement[x]}, {x, 0, 300},
           Filling → {1 → {2}}, AxesLabel → {t[s], {temp, original}}]

Out[130]=
```

BENEFITS

Numerical approximate requirements/design tradeoffs can be performed

TOOLS

Eclipse Modeling
Eclipse Xtext
Wolfram Mathematica
Mathematica Workbench

Example 4:

Alarm System Menus

CONTEXT

Alarm Systems
Operator Panels

PROBLEM

The structure of the UI and menus of the operator panel was described in Word, and then manually implemented.

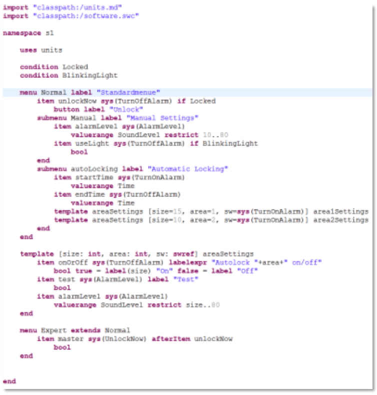
Error prone, slow, …

SOLUTION

Use a DSL to describe menu structures directly, and generate various artifacts from it:
- flash simulator data
- C code for implementation
- i18n templates

SOLUTION

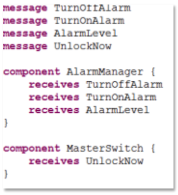
Menu Structure



```
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
```

SOLUTION

Software Components



```
message TurnOffAlarm
message TurnOnAlarm
message AlarmLevel
message UnlockNow

component AlarmManager {
    receives TurnOffAlarm
    receives TurnOnAlarm
    receives AlarmLevel
}

component MasterSwitch {
    receives UnlockNow
}
```

SOLUTION

BENEFITS

Various non-software artifacts could be generated

Integration with software structure simpler

Fewer errors, faster…

TOOLS

eclipse modeling PROJECT

xtext

Eclipse Modeling
Eclipse Xtext

Example 5:

Requirements Tracability

CONTEXT

Embedded Systems developed with a C-derivative

PROBLEM

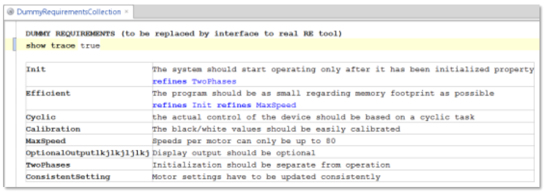
Tracability to textual requirements is necessary.

SOLUTION

Import Requirements into the tool and then use traceability annotations to refer to them from any program element.

SOLUTION

Imported Requirements

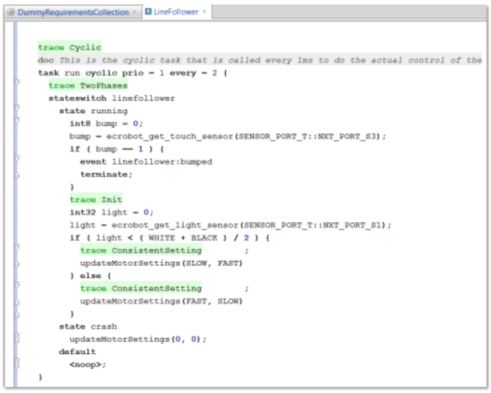


```
DUMMY REQUIREMENTS (to be replaced by interface to real RE tool)
show trace true
```

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

SOLUTION

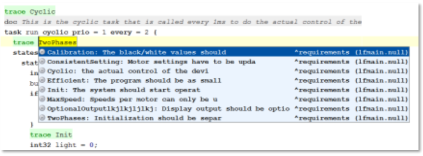
Program Code with Annotations (green)



```
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>;
}
```

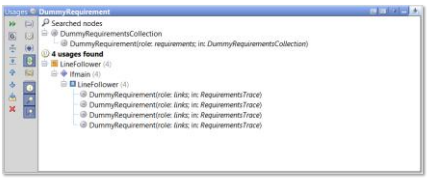
SOLUTION

Selecting from the Requirements



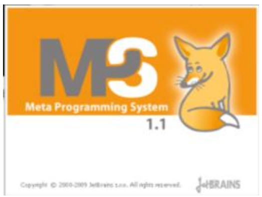
```
trace Cyclic
doc This is the cyclic task that is called every 1ms to do the actual control of the
task run cyclic prio = 1 every = 2 {
  trace TwoPhases
  states Calibration: The black/white values should                      ^requirements (lfmain.null)
    stat ConsistentSetting: Motor settings have to be upda               ^requirements (lfmain.null)
      in Cyclic: the actual control of the devi                          ^requirements (lfmain.null)
      bu Efficient: The program should be as small                       ^requirements (lfmain.null)
      if Init: The system should start operat                            ^requirements (lfmain.null)
         MaxSpeed: Speeds per motor can only be u                        ^requirements (lfmain.null)
         OptionalOutputlkjlkjljlkj: Display output should be optio       ^requirements (lfmain.null)
         TwoPhases: Initialization should be separ                       ^requirements (lfmain.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)
```

TOOLS

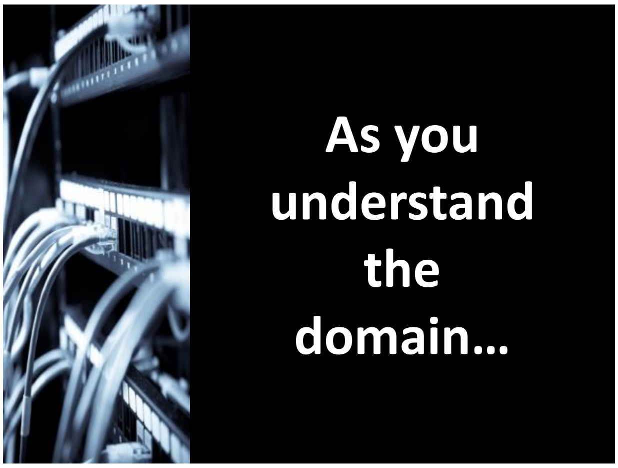


JetBrains MPS

What if I don‘t yet have a language?

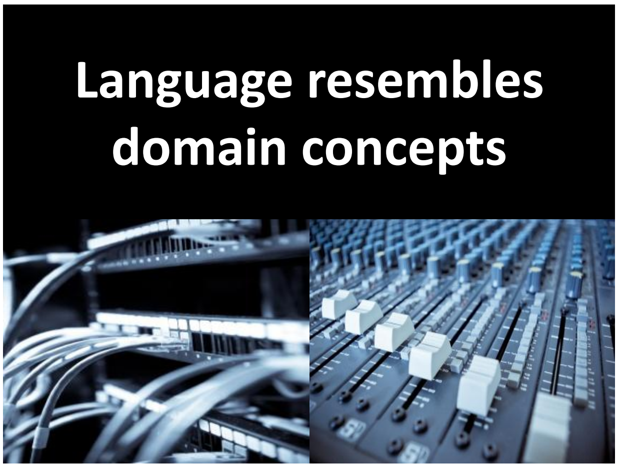
Actually, this is the normal case!
Domain Specific Language

Building Languages

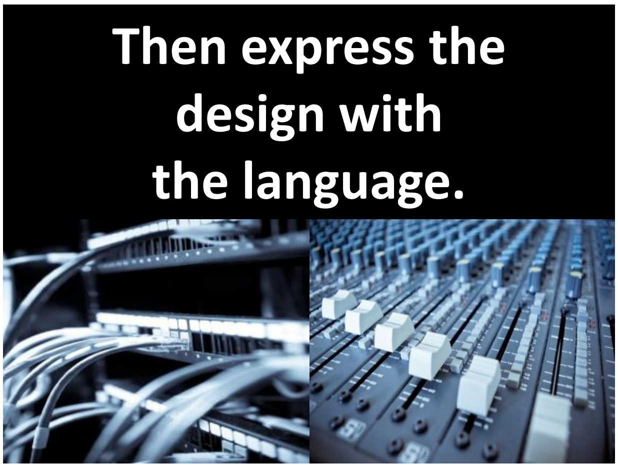


As you understand the domain…

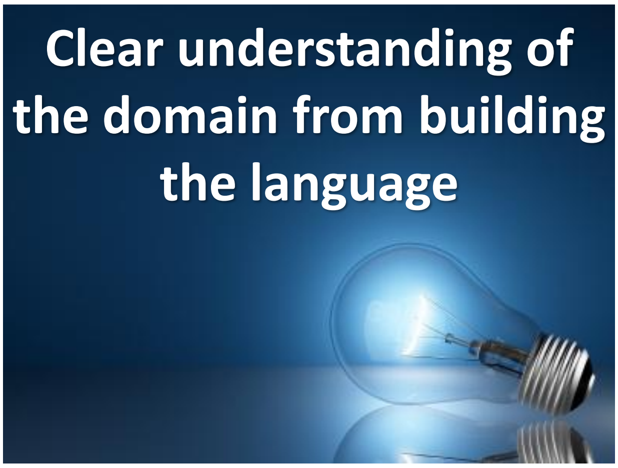
…develop a language to express it!



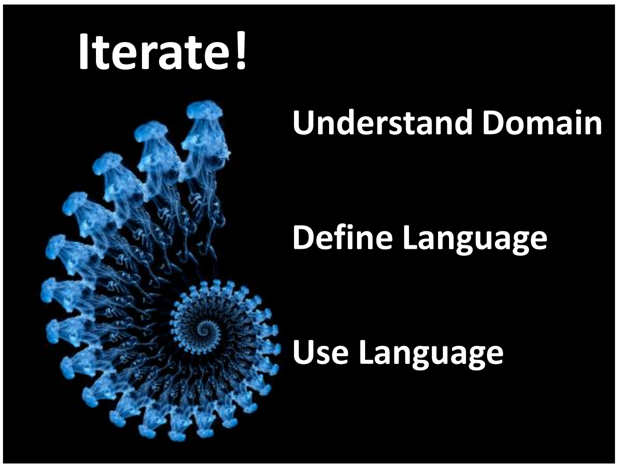
Language resembles domain concepts



Then express the design with the language.



Clear understanding of the domain from building the language

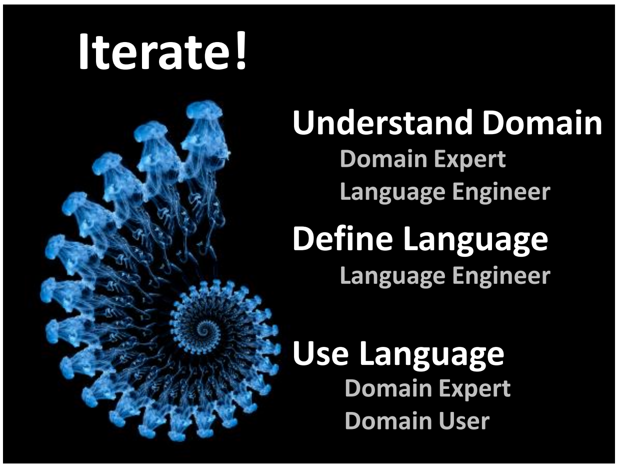


Iterate!

Understand Domain

Define Language

Use Language

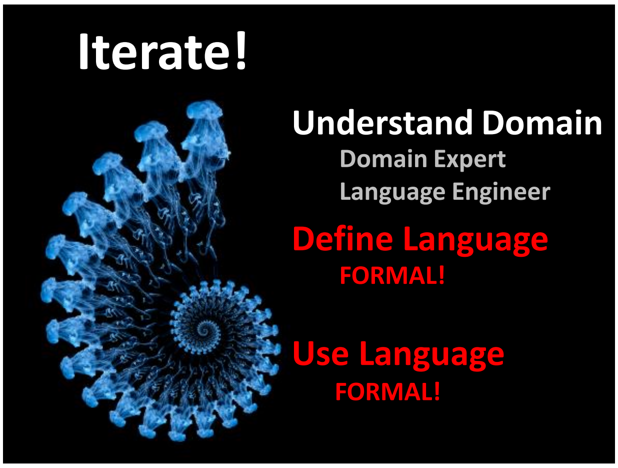


Iterate!

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!

Like
Analysis!

Like
Analysis!

… but with an
executable
result

DSL Engineering Tools

Notations

Notations
Editors

Notations
Editors
Multi-Languages

Notations
Editors
Multi-Languages
Debugger

Notations
Editors
Multi-Languages
Debugger
Testing

Notations
Editors
Multi-Languages
Debugger
Testing

Groupware

Notations
Editors
Multi-Languages
Debugger
Testing

Groupware
Scalable

Notations
Editors
Multi-Languages
Debugger
Testing

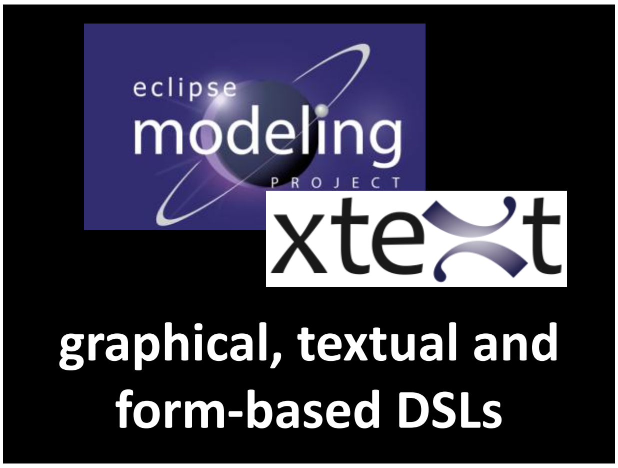
Groupware
Scalable
Integrated

eclipse modeling PROJECT
xtext

Open Source
Eclipse Public License

eclipse modeling PROJECT
xtext

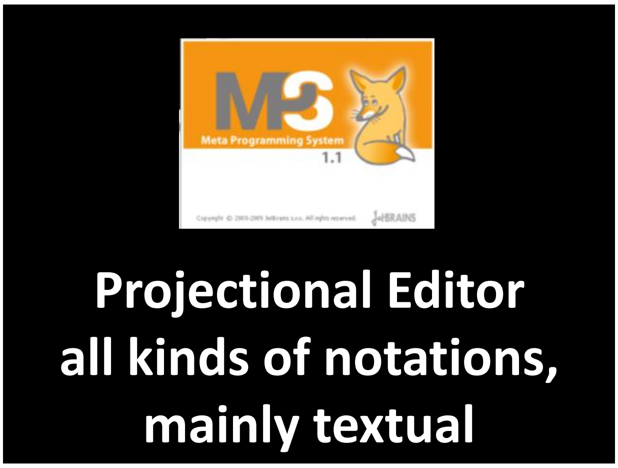
Large wold wide community


eclipse modeling PROJECT
xtext

graphical, textual and form-based DSLs

MPS
Meta Programming System
1.1
JetBRAINS

Developed by JetBrains
Open Source
Apache 2.0


MPS
Meta Programming System
1.1
JetBRAINS

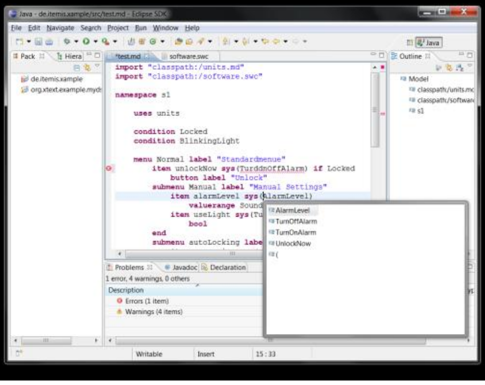
Projectional Editor
all kinds of notations, mainly textual

INTENTIONAL®
SOFTWARE

Commercial Product
Projectional Editor
Most flexible notations

Notational Flexibility?

Textual Rich IDEs



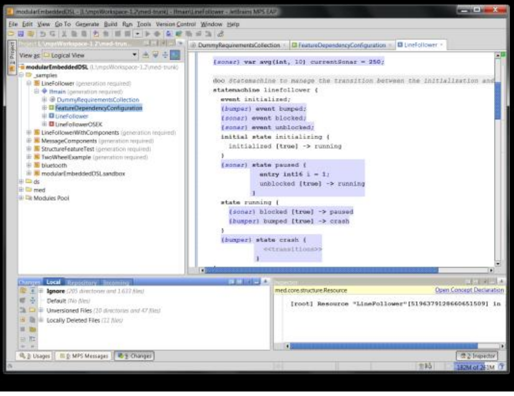
```
import "classpath:/units.md"
import "classpath:/software.swc"

namespace s1

    uses units

    condition Locked
    condition BlinkingLight

    menu Normal label "Standardmenue"
        item unlockNow sys(TurddnOffAlarm) if Locked
            button label "Unlock"
        submenu Manual label "Manual Settings"
            item alarmLevel sys(AlarmLevel)
                valuerange Sound
            item useLight sys(Tu
                bool
        end
        submenu autoLocking labe
```

Problems | Javadoc | Declaration

1 error, 4 warnings, 0 others

Description
- Errors (1 item)
- Warnings (4 items)

Textual Rich IDEs



```
(sonar) var avg(int, 10) currentSonar = 250;

doc Statemachine to manage the transition between the initialization and
statemachine linefollower {
  event initialized;
  (bumper) event bumped;
  (sonar) event blocked;
  (sonar) event unblocked;
  initial state initializing {
    initialized [true] -> running
  }
  (sonar) state paused {
            entry int16 i = 1;
            unblocked [true] -> running
          }
  state running {
    (sonar) blocked [true] -> paused
    (bumper) bumped [true] -> crash
  }
  (bumper) state crash {
             <<transitions>>
           }
```

med.core.structure.Resource   Open Concept Declaration

```
[root] Resource "LineFollower"[5196379128660651509] in
```

Textual

Languages and Editors are easier to build

Textual

Languages and Editors are easier to build

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

Textual

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

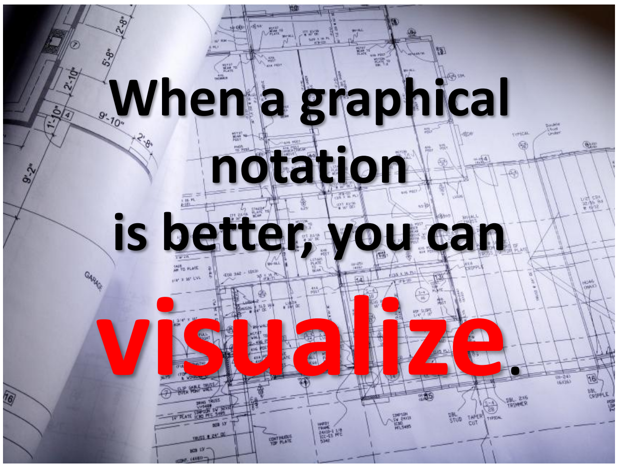
Textual

adapting existing models as the DSL evolves

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

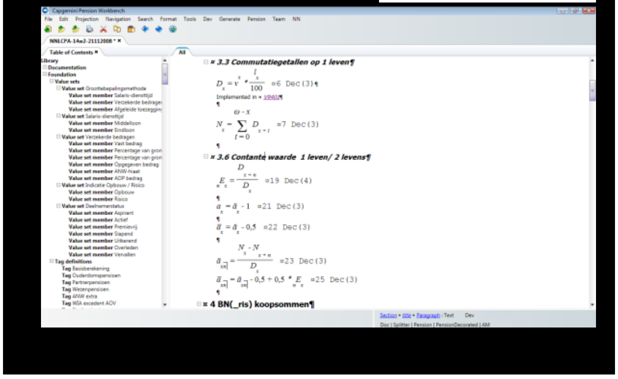
Textual

Many Developers prefer textual notations



When a graphical notation is better, you can **visualize**.

Mixed


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 bedragen
      - Value set member Afgeleide toezegging
    - 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 ¤ V9402¶

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

¶

¤ *3.6 Contante waarde 1 leven/ 2 levens*¶

${}_nE_x = \frac{D_{x+n}}{D_x}$ ¤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)

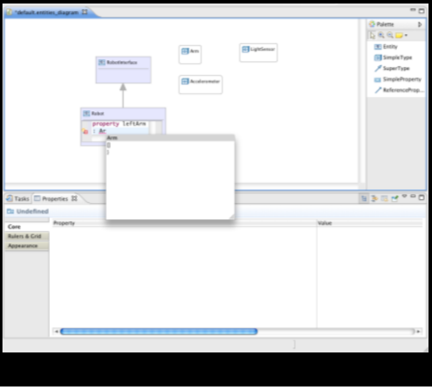
$\bar{a}_{x\overline{n}|} = \ddot{a}_{x\overline{n}|} - 0{,}5 + 0{,}5 * {}_nE_x$ ¤25 Dec(3)

¶

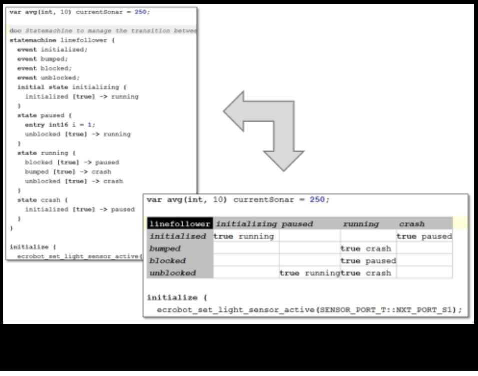
¤ **4 BN(_ris) koopsommen¶**

Section • title • Paragraph : Text Dev
Doc | Splitter | Pension | PensionDecorated | AM

Mixed



Multiple



```
var avg(int, 10) currentSonar = 250;

doc Statemachine to manage the transition betwe
statemachine linefollower {
  event initialized;
  event bumped;
  event blocked;
  event unblocked;
  initial state initializing {
    initialized [true] -> running
  }
  state paused {
    entry int16 i = 1;
    unblocked [true] -> running
  }
  state running {
    blocked [true] -> paused
    bumped [true] -> crash
    unblocked [true] -> crash
  }
  state crash {
    initialized [true] -> paused
  }
}

initialize {
  ecrobot_set_light_sensor_active(
```

```
var avg(int, 10) currentSonar = 250;
```

linefollower	initializing	paused	running	crash
initialized	true running			true paused
bumped			true crash	
blocked			true paused	
unblocked		true running	true crash	

```
initialize {
  ecrobot_set_light_sensor_active(SENSOR_PORT_T::NXT_PORT_S1);
```

Standards & UML

You can model everything with UML

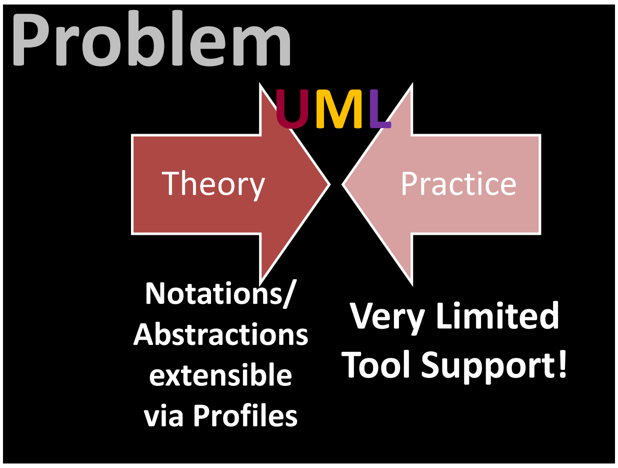
You can model everything with UML somehow!

Problem

Shoehorning domain abstractions into the generic language

Problem

Sidetracked
by existing
abstractions
and notations



Problem

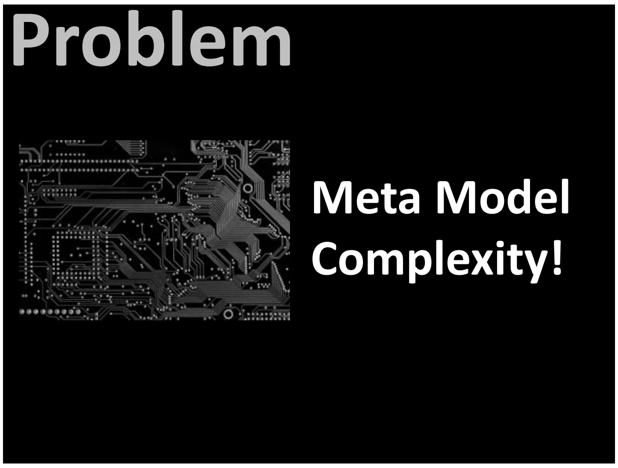
UML

Theory

Practice

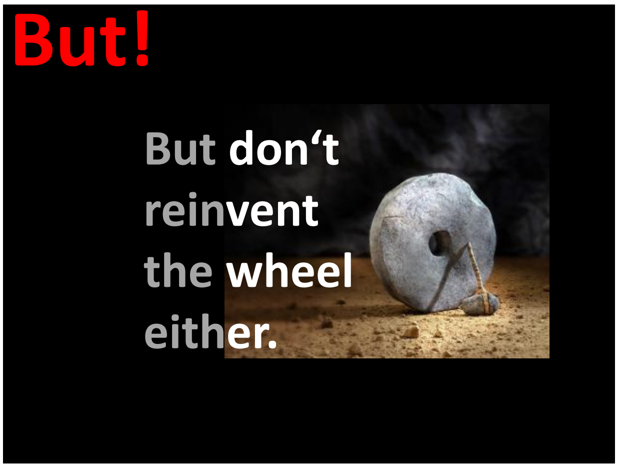
Notations/
Abstractions
extensible
via Profiles

Very Limited
Tool Support!



Problem

Meta Model
Complexity!



But!

But don‘t
reinvent
the wheel
either.

Where are
standards
useful?

People have
to learn
underlying
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?

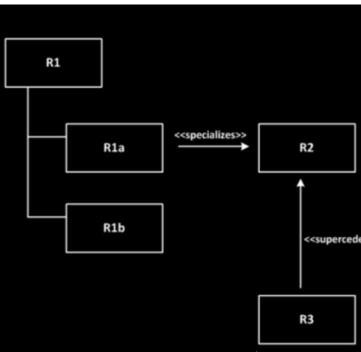
Similar statements can be made relative to

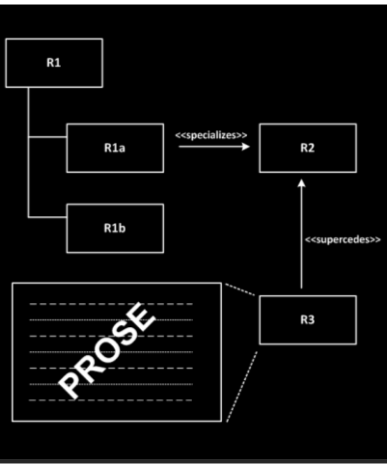
BPMN

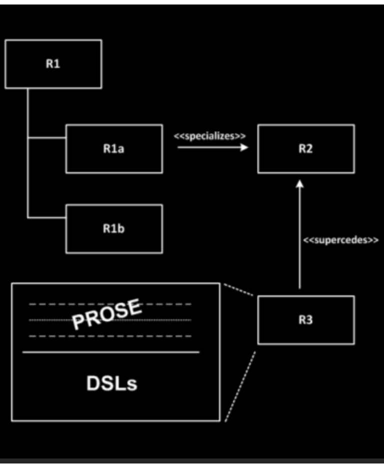
Textual Requirements?

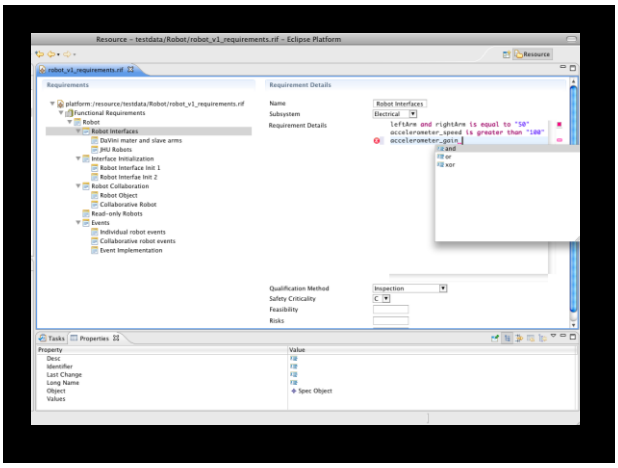
Textual Requirements

still necessary.








Resource – testdata/Robot/robot_v1_requirements.rif – Eclipse Platform

Requirements

- platform:/resource/testdata/Robot/robot_v1_requirements.rif
  - Functional Requirements
    - Robot
      - Robot Interfaces
        - DaVini 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_gain_
```

Qualification Method: Inspection

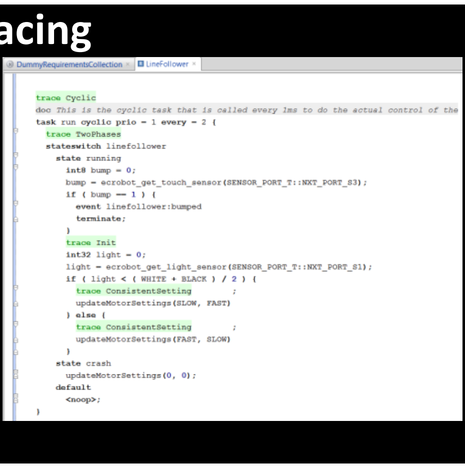
Safety Criticality: C

Feasibility:

Risks:

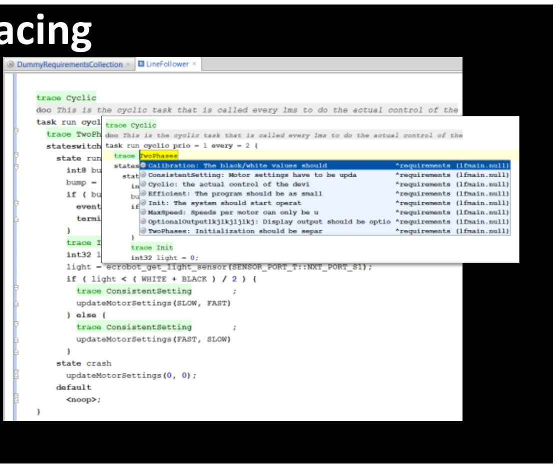
Property	Value
Desc	
Identifier	
Last Change	
Long Name	
Object	Spec Object
Values	

Tracing



```
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



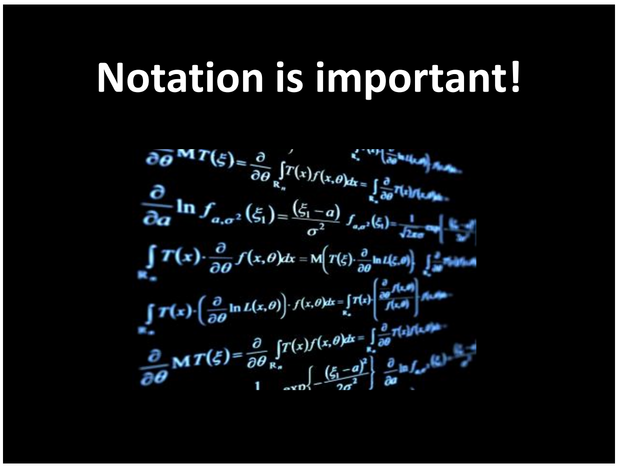
```
trace Cyclic
doc This is the cyclic task that is called every 1ms to do the actual control of the
task run cyclic prio = 1 every = 2 {
  trace TwoPhases
```

```
Calibration: The black/white values should               ^requirements (lfmain.null)
ConsistentSetting: Motor settings have to be upda        ^requirements (lfmain.null)
Cyclic: the actual control of the devi                   ^requirements (lfmain.null)
Efficient: The program should be as small                ^requirements (lfmain.null)
Init: The system should start operat                     ^requirements (lfmain.null)
MaxSpeed: Speeds per motor can only be u                 ^requirements (lfmain.null)
OptionalOutputlkjlkjljlkj: Display output should be optio ^requirements (lfmain.null)
TwoPhases: Initialization should be separ                ^requirements (lfmain.null)
```

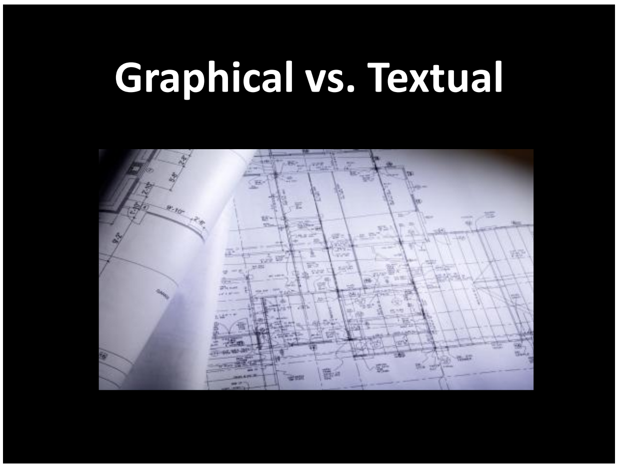
```
      }
      trace Init
      int32 light = 0;
      light = ecrobot_get_light_sensor(SENSOR_PORT_T::NXT_PORT_S1);
      if ( light < ( WHITE + BLACK ) / 2 ) {
        trace ConsistentSetting        ;
        updateMotorSettings(SLOW, FAST)
      } else {
        trace ConsistentSetting        ;
        updateMotorSettings(FAST, SLOW)
      }
    state crash
      updateMotorSettings(0, 0);
    default
      <noop>;
}
```

Things to keep in mind

Limit Expressiveness



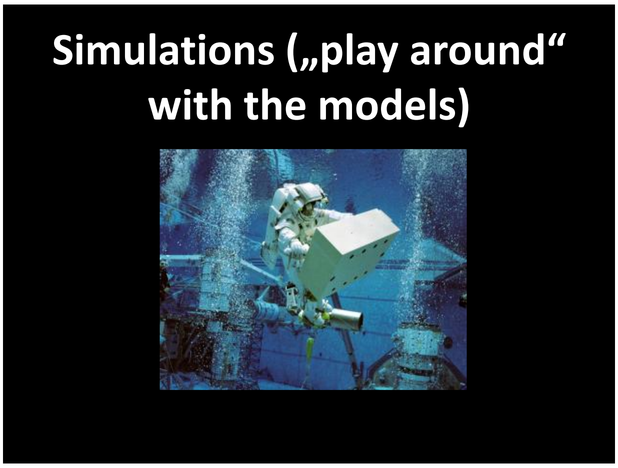
Notation is important!



Graphical vs. Textual

Invest in good constraints

Testing may be an important benefit of your DSL



Simulations („play around“ with the models)

Who are 1st Class Citizens?

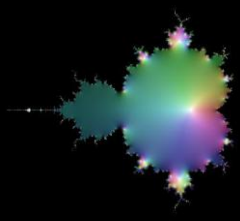
Learn from - but don‘t copy – programming languages

Support for Reuse and Variations

Tooling Matters!

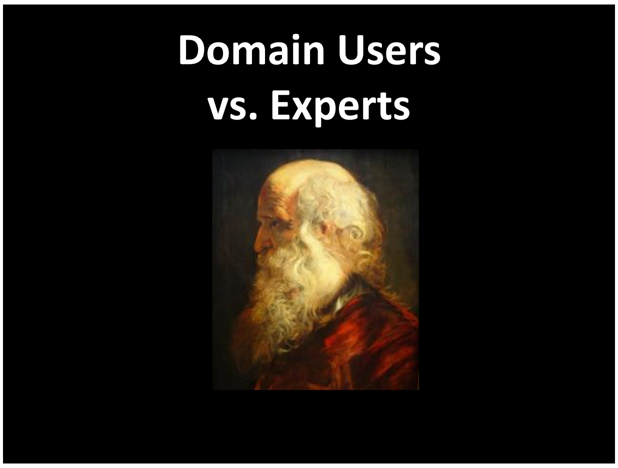
Annotation Models to add technical details

Develop the language iteratively!

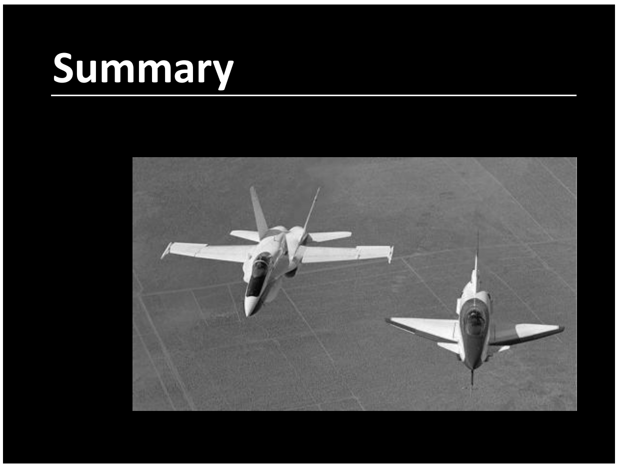


Co-Evolve Language and Concepts

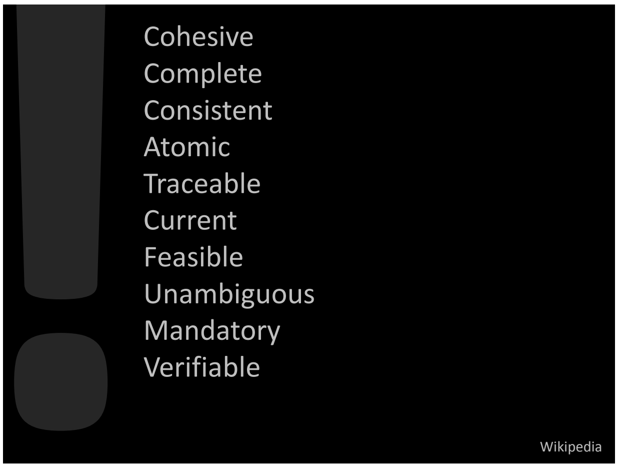
Can domain users actually „program“?



Domain Users vs. Experts

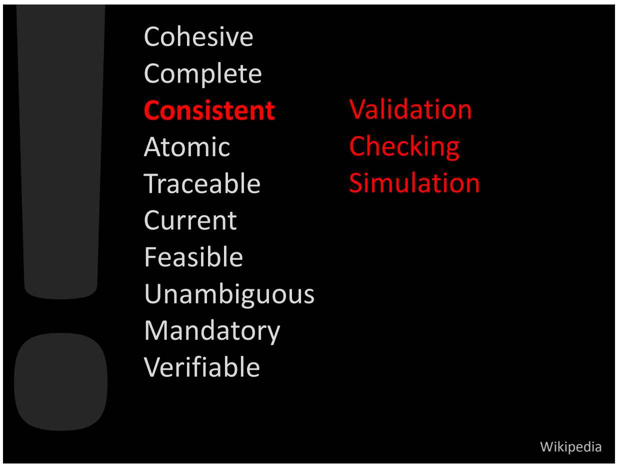


Summary



Cohesive
Complete
Consistent
Atomic
Traceable
Current
Feasible
Unambiguous
Mandatory
Verifiable

Wikipedia



Cohesive
Complete
Consistent Validation
Atomic Checking
Traceable Simulation
Current
Feasible
Unambiguous
Mandatory
Verifiable

Wikipedia

Cohesive
Complete
Consistent
Atomic
Traceable Everything is a model tracing links simple
Current
Feasible
Unambiguous
Mandatory
Verifiable

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

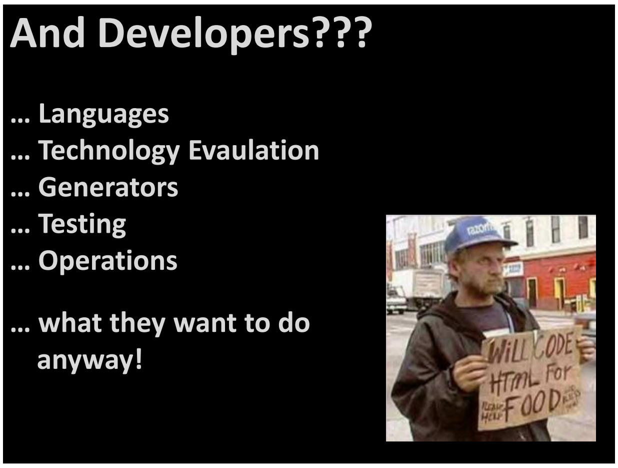
Cohesive
Complete
Consistent
Atomic
Traceable
Current
Feasible
Unambiguous
Mandatory
Verifiable
Executable Automatic Refinement downstream, Code Gen.

Cohesive
Complete
Consistent
Atomic
Traceable
Current
Feasible
Unambiguous
Mandatory
Verifiable
Executable

Reward

for the additional effort of formalization!

And Developers???

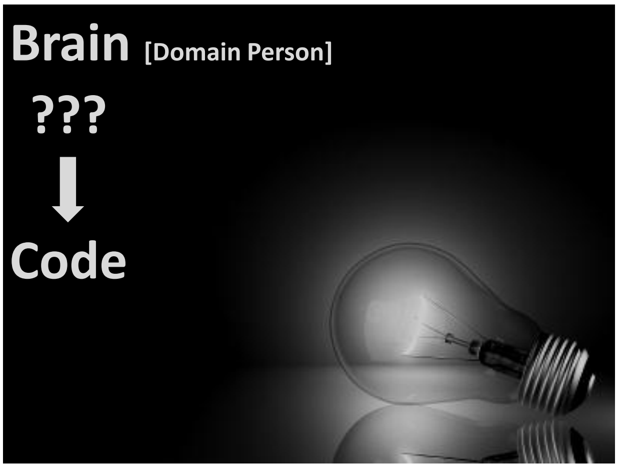


And Developers???

… Languages
… Technology Evaulation
… Generators
… Testing
… Operations

… what they want to do anyway!

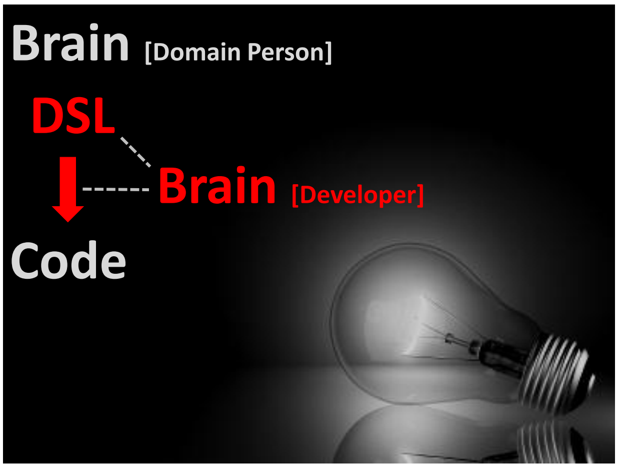
Will code HTML for FOOD



Brain [Domain Person]

???

Code



Brain [Domain Person]

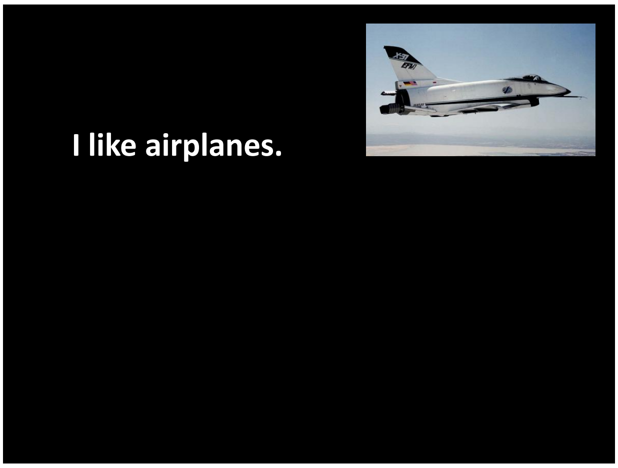
DSL

Brain [Developer]

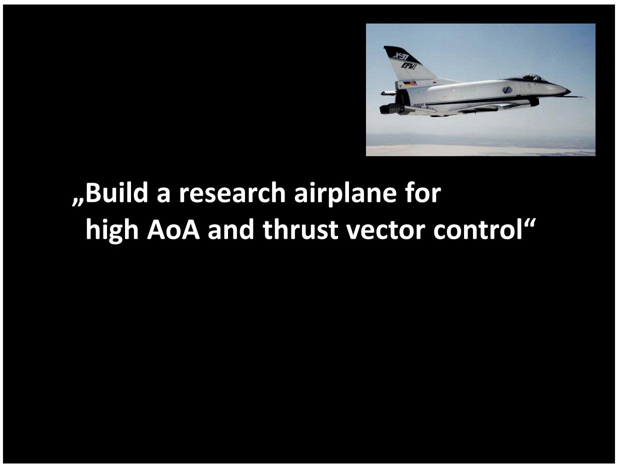
Code

One more thing:

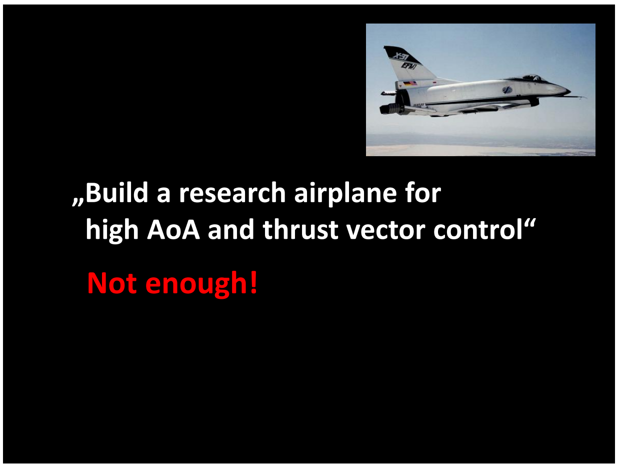
One more thing:
Why all these pictures?



I like airplanes.

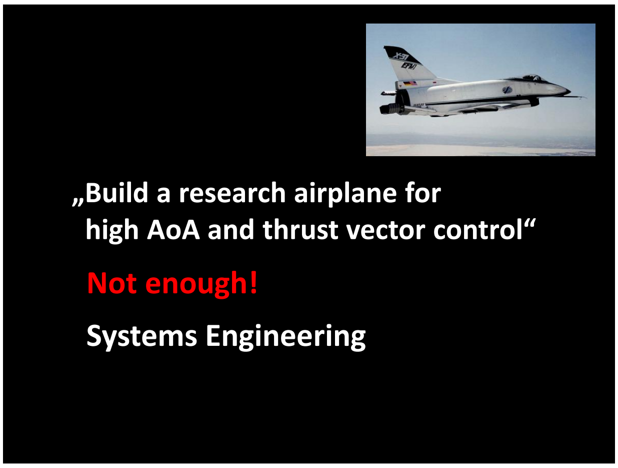


„Build a research airplane for
high AoA and thrust vector control“



„Build a research airplane for
high AoA and thrust vector control“

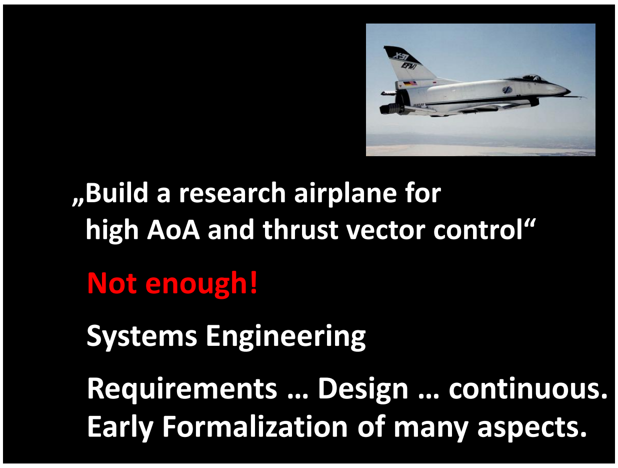
Not enough!



„Build a research airplane for
high AoA and thrust vector control“

Not enough!

Systems Engineering

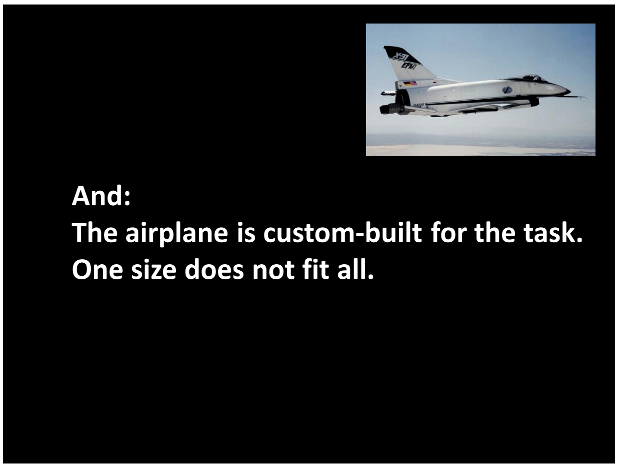


„Build a research airplane for
high AoA and thrust vector control“

Not enough!

Systems Engineering

Requirements … Design … continuous.
Early Formalization of many aspects.



And:
The airplane is custom-built for the task.
One size does not fit all.

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