Domain-Specific Languages

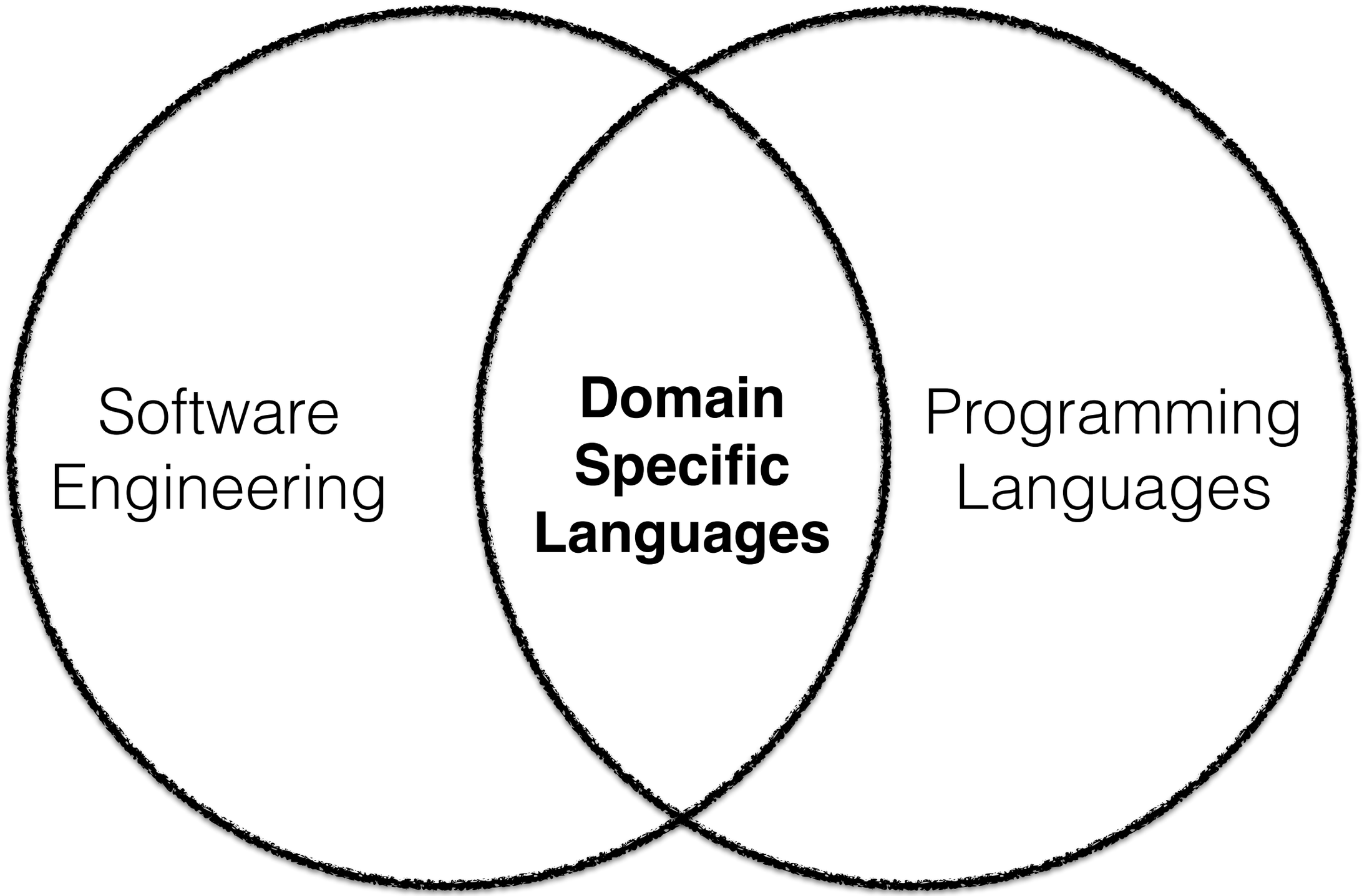
Tijs van der Storm

CWI
Centrum Wiskunde & Informatica

university of groningen

Domain-specific languages

- Formal representations of domain knowledge
- AKA: Executable specification languages
- Write down “what”, generate the “how”
- Tailored to particular problem domain (e.g. Tax administration, digital forensics, etc.)
- Easier to validate by domain experts


Software Engineering
Domain Specific Languages
Programming Languages

Programming

```
.text:131411EF                  ; FUNCTION CHUNK AT .text:13141239 SIZE 000
.text:131411EF                  ; FUNCTION CHUNK AT .vmp0:13143000 SIZE 000
.text:131411EF                  ; FUNCTION CHUNK AT .vmp0:13143522 SIZE 000
.text:131411EF                  ; FUNCTION CHUNK AT .vmp0:13145C99 SIZE 000
.text:131411EF
.text:131411EF 68 F1 4F 5B FF                   push    0FF5B4FF1h
.text:131411F4 E9 A0 4A 00 00                   jmp     loc_13145C99
.text:131411F4                  start           endp
.text:131411F4
.text:131411F4                  ; ---------------------------------------
.text:131411F9 64 69 31         byte_131411F9   db 64h, 69h, 31h        ; D
.text:131411FC 60 16 42 75 9A C4+               dd 75421660h, 0CACDC49Ah, 7
.text:131411FC CD CA FA 7B 9D 7B+               dd 6EEE544Dh, 10D69610h, 0F
.text:131411FC D5 D4 48 2B 85 4B+               dd 0EB6323E4h, 82B58465h, 0
.text:13141238                  ; ---------------------------------------
.text:13141238 58                               pop     eax
.text:13141239                  ; START OF FUNCTION CHUNK FOR start
.text:13141239
.text:13141239                  loc_13141239:                           ; C
.text:13141239 52                               push    edx
.text:1314123A 55                               push    ebp
.text:1314123B 53                               push    ebx
.text:1314123C 51                               push    ecx
.text:1314123D 9C                               pushf
.text:1314123E 57                               push    edi
.text:1314123F 50                               push    eax
.text:13141240 56                               push    esi
.text:13141241 51                               push    ecx
.text:13141242 68 00 00 00 00                   push    0
.text:13141247 8B 74 24 28                      mov     esi, [esp+2Ch+var_4
.text:1314124B BF F9 11 14 13                   mov     edi, offset byte_13
.text:13141250
.text:13141250                  loc_13141250:                           ; C
.text:13141250 89 F3                            mov     ebx, esi
.text:13141252 03 34 24                         add     esi, [esp+2Ch+var_2
.text:13141255
.text:13141255                  loc_13141255:                           ; C
.text:13141255                                                          ; s
```

Domain Knowledge

Hard labour

Code

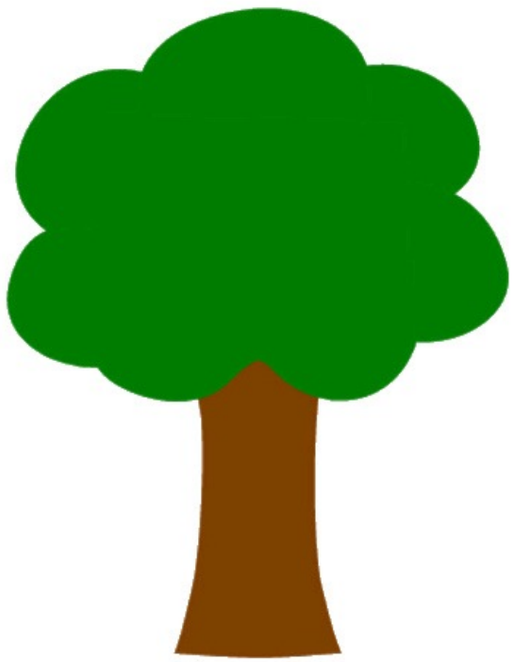
Change

???

Domain-specific Language

Automation

Code



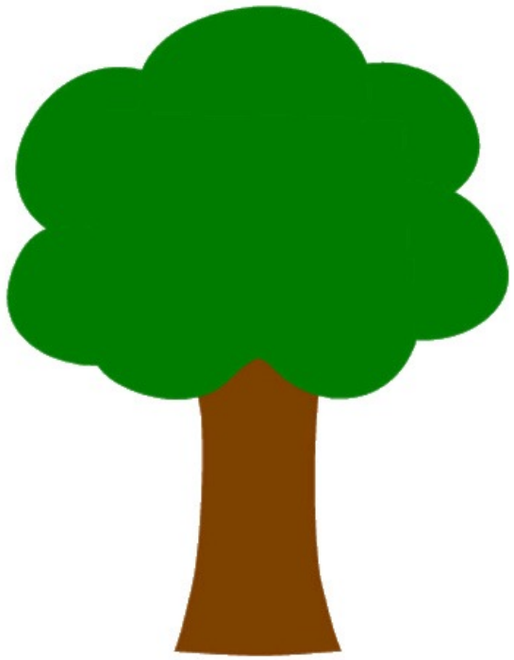
“What”
formalized
requirements

“How”
formalized design
and architecture

Domain-specific Language

Automation

Code

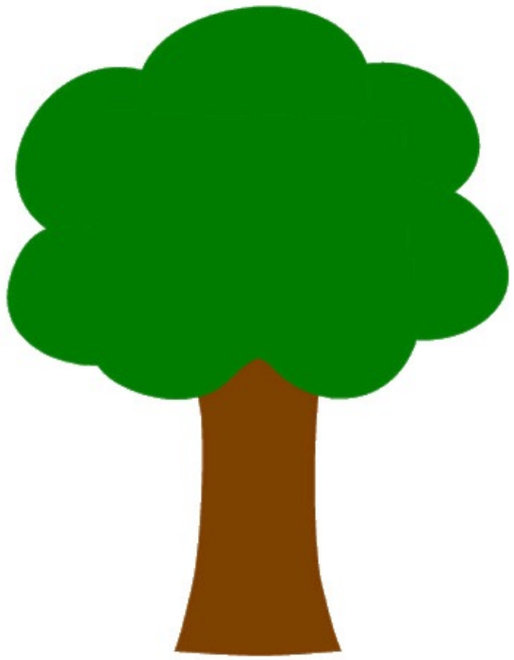


Change

Domain-specific Language

Automation

Code



Change

Change

Programming is lossy

- encoding
- obfuscating
- encrypting
- dispersing
- tangling
- distorting

Time consuming

SCRIPTORIUM MONK AT WORK. (From *Lacroix.*)

Change is error-prone

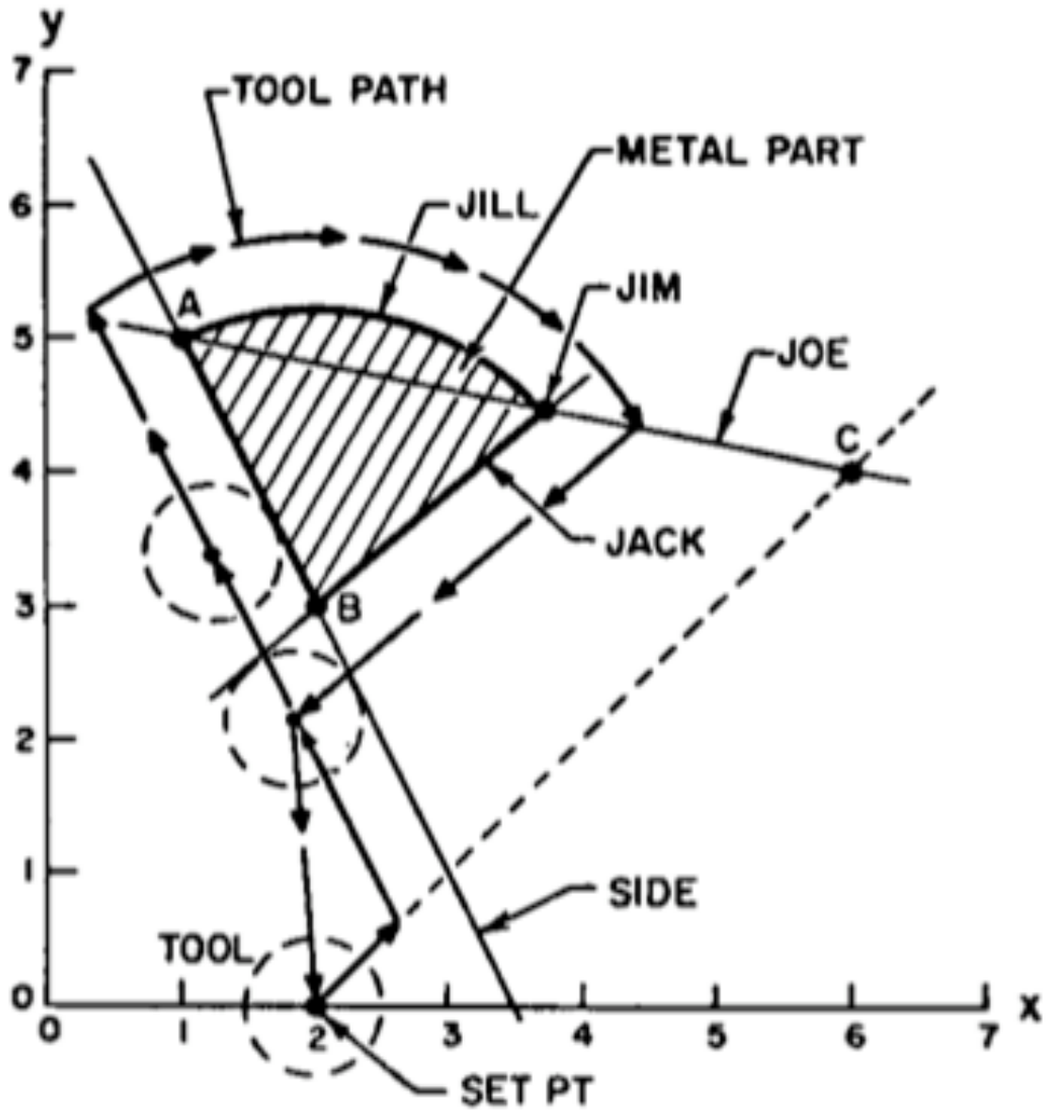
Cognitive distance

?

General purpose languages

Domain-specific languages

APT: robotic control




```
A    =  POINT / 1, 5
B    =  POINT / 2, 3
C    =  POINT / 6, 4
        TL DIA / +1.0, INCH
        FEDRAT / 30, IPM
SET PT =  FROM, POINT / 2, 0
        IN DIR, POINT / C
SIDE   =  GO TO, LINE / THRU, A, AND, B
        WITH, TL LFT, GO LFT, ALONG / SIDE
JILL   =  GO RGT, ALONG, CIRCLE / WITH, CTR AT, B, THRU, A
JOE    =  LINE / THRU, A, AND, C
JIM    =  POINT / X LARGE, INT OF, JOE, WITH, JILL
JACK   =  LINE / THRU, JIM, AND, B
        GO RGT, ALONG / JACK, UNTIL, TOOL, PAST, SIDE
        GO TO / SET PT
        STOP, END, FINI
```

from the ’50s (!)

Risla: financial products

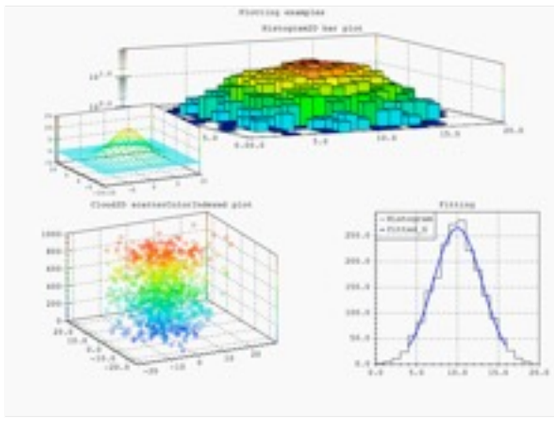
Developed at CWI

```
product LOAN

declaration
   contract data
      PAMOUNT : amount                    %% Principal Amount
      STARTDATE : date                    %% Starting date
      MATURDATE : date                    %% Maturity data
      INTRATE : int-rate                  %% Interest rate
      RDMLIST := [] : cashflow-list       %% List of redemptions.

information
      PAF : cashflow-list                 %% Principal Amount Flow
      IAF : cashflow-list                 %% Interest Amount Flow

registration
      %% Register one redemption.
      RDM(AMOUNT : amount, DATE : date)
```

Case 1: digital forensics

Acquisition

Recovery



Analysis

File carving tools

- Carving: recovering files without using file system
- Hard to change because of tangling
- Hard to change because highly optimized
- Complicating forces:
  - high variability in the domain (formats, devices)
  - deadline pressure during pre-charge detainment

Derric

```
format jpeg
extension jpeg jpg jfif

unit byte
size 1
sign false
type integer
endian big
strings ascii

sequence
  SOI
  ([APP0JFIF APP0JFXX?] [APP1 APP2?])
  !(SOI APP0JFIF APP0JFXX EOI)*
  EOI
```

- Separate file format description from carving algorithms
- File formats can be understood by forensic experts
- Generate high performance code.

Lessons learned

- Derric carvers perform at least as good as existing state-of-the-art carvers
  - Accuracy and run-time performance
- Derric file carver amounts to ±1000 SLOC
  - Including: specification, code generator, runtime
- Compare: PhotoRec, ReviveIt are huge C programs with only builtin file format support.

Case 2: tax administration

Gegevensspecificatie

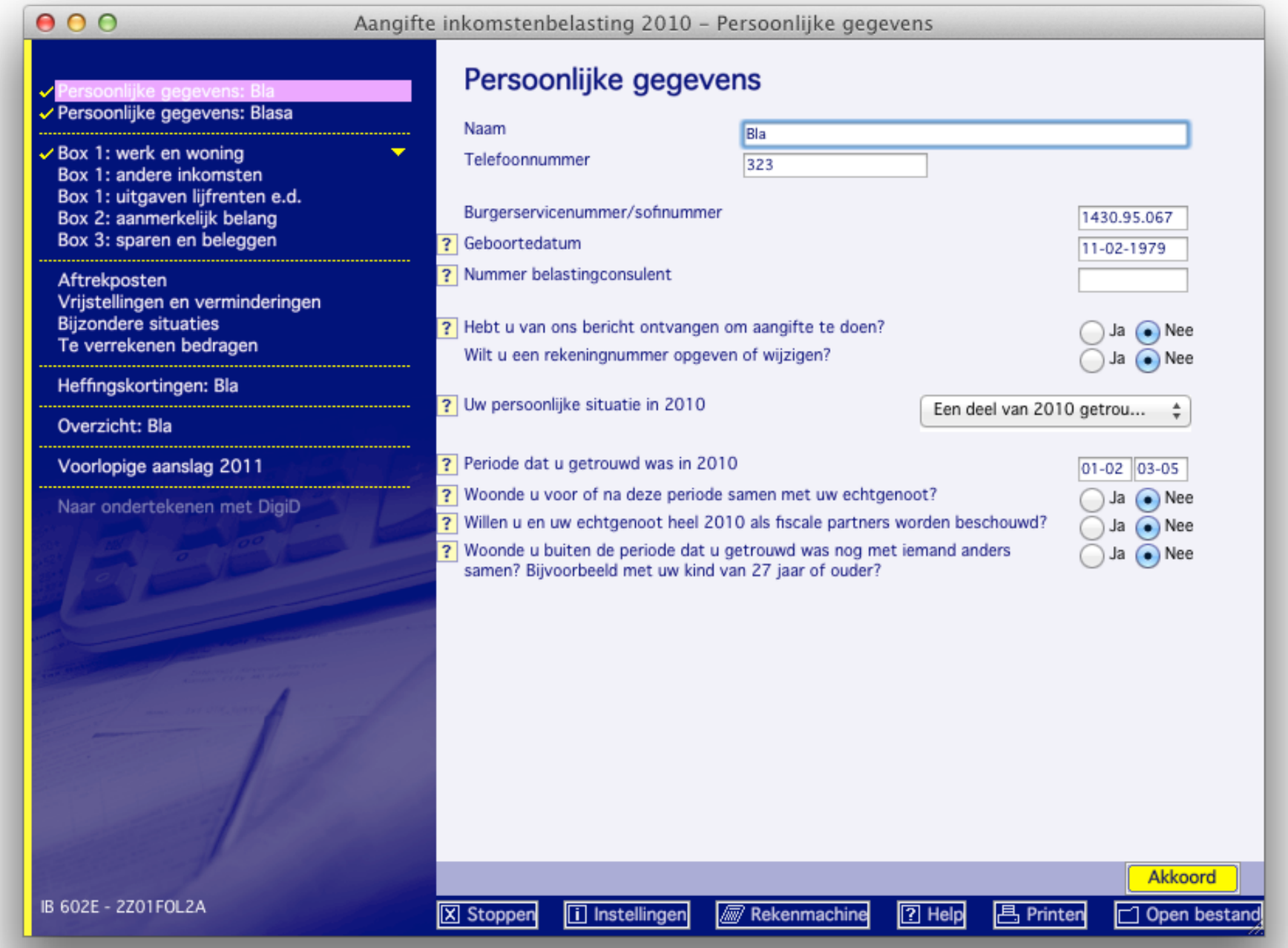
Behorende bij de specificatie
van de berichtstructuur
DA-2015

Versie 3.0 van 17/8/2015

Versie 3.0
Definitief
17/8/2015

1147 page PDF document

The running program



Consequences

- Enormous amount of work: programmers manually implement ±1500 data items + interaction logic

- Gargantuan testing effort: lots of data, lot of choices in the UI, — no automated support.

- Validation: how to know if the code is correctly implementing the spec?

- Every year, everything changes again, but unit of change != unit of code.

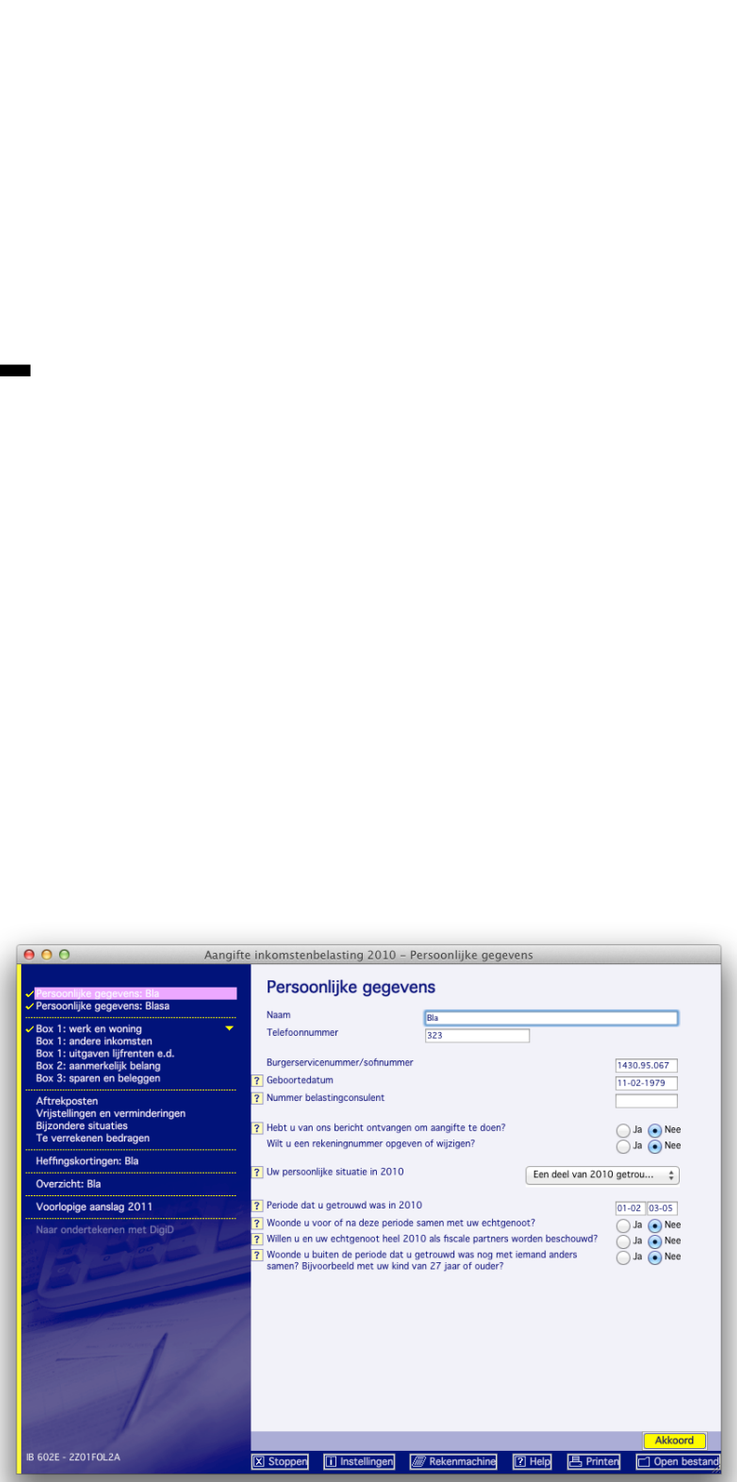
Using a DSL

extract

formal

specification

generate



```
gegeven periode samenwonen in belastingjaar [117315]: DDMMEEJJ {
  alias EPerSamenwonen
  minimaal voorkomen 0
  maximaal voorkomen 1
  gebruikt in
    117308
  pagina 73
  definitie {
    De periode in het belastingjaar waarin de
    aangever en de partner onafgebroken een
    gezamenlijke huishouding hebben gevoerd
    en op hetzelfde woonadres stonden ingeschreven.
  }
  bron {
    Wet IB2001, art 1.2
  }
  conditie
    [117315.B]<periode samenwonen in belastingjaar.Begindatum>
       <= [117315.E]<periode samenwonen in belastingjaar.Einddatum>
}
```

EasierTax

- DA-2015
  - 3.0
    - Inkomensheffingen, Zorgverzekeringswet, Ve
      - Zorgverzekeringswet
      - Inkomstenbelasting en premies volksverzek
        - Inkomensopstelling kwalificerende buiten
          - Inkomensopstelling kwalificerende buit
            - Vermeerdering-vermindering inkome
            - Grondslag inkomsten verhouding Ned
              - Inkomen belast Nederland samenst
              - Inkomen in en buiten Nederland sa
          - Inkomensopstelling partner kwalificerer
            - Vermeerdering-vermindering inkome
            - Grondslag inkomsten verhouding Ned
              - Inkomen in en buiten Nederland pa
              - Inkomen belast Nederland partner
      - Persoonsgebonden aftrek
        - Totaal aftrekbare giften
          - Andere giften
            - Andere giften aan culturele instellir
              - Andere giften culturele instelling
            - Overige andere giften
              - Overige andere giften spec
            - Andere giften aan steunstichtingen
              - Andere giften steunstichtingen S
          - Periodieke giften
            - Overige periodieke giften
              - Overige periodieke giften spec

Giften aftrekbaar aangever migratiejaar Nederl periode

Het deel van de aftrekbare giften van de aangever dat betrekking heeft op de periode wone Nederland in het migratiejaar.

Bronnen: Wet IB 2001, art 6.32

n..13

Andere giften culturele instelling

De giften die betaald, verrekend of ter beschikking zijn gesteld aan een culturele instelling.

Bronnen: Wet IB 2001, art 6.32 Wet IB 2001, art 6.39a

n..13

Omschrijving andere giften culturele instelling

De naam van de culturele instelling of omschrijving waaraan de gift is gedaan.

Bronnen: Wet IB 2001, art 6.33Algemene Wet Inzake Rijksbelastingen, artikel 5b

an..70

\+ -

Verhoging aftrek andere giften culturele instellingen

De verhoging van de aftrek van andere giften gedaan aan culturele instellingen.

Bronnen: Wet IB 2001, art 6.39a, lid 1

n..13

Andere giften culturele instellingen totaal

Het totaal van de giften aan culturele instellingen.

Bronnen: Wet IB 2001, art 6.32 Wet IB 2001, art 6.39a

n..13

Aftrekbare andere giften

Het bedrag van de andere giften aan instellingen of aan steunstichtingen SBBI

Lessons we are learning

- Formal specification of data items and rules
- Enables:
  - consistency checking
  - generation of code
  - impact analysis upon new versions
  - …

Shorter programs

- Commonality is in the language implementation
- Only specify relevant variation points
- Remove repetitive patterns / “boilerplate” which are error-prone
- “Higher level of abstraction”

Fewer degrees of freedom

- Less room to make mistakes
- Guaranteed absence of certain problems:
  - termination, resource usage, etc.
- Better checking / domain-specific error messages

Notation close to problem domain

- Facilitates validation / stake holder communication
- Towards “formalized” requirements

Untangling concerns

- Coupling/scattering issue resolved in code generator

- Think: security, optimization, contracts, persistence, etc.

Domain-specific AVOPT

- Limited expressivity + encoded domain knowledge:
  - better error messages
  - better performance
  - parallelization

Design reuse

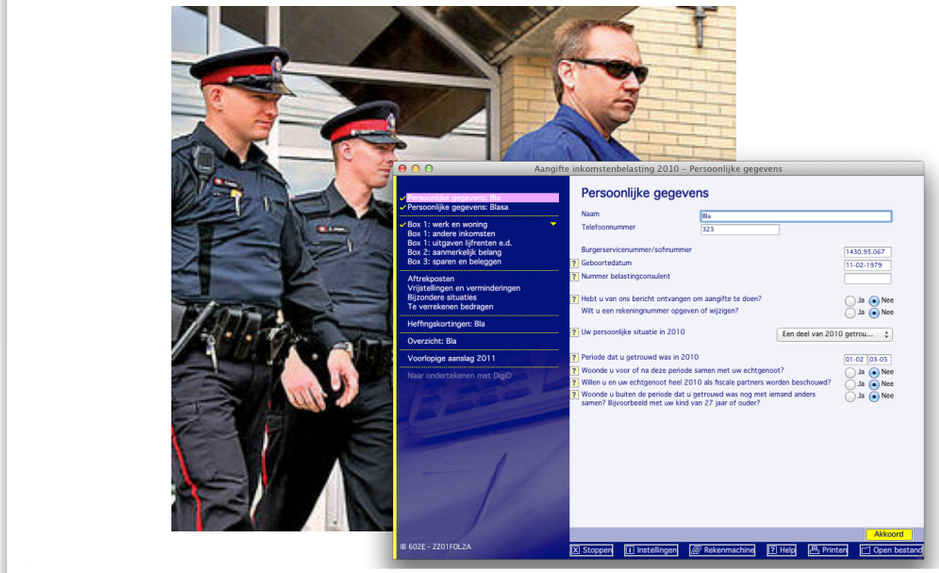
- Code generator / interpreter encodes design knowledge
- Leveraged across multiple systems of a family.
- Change the design without changing application logic (“what”)

DSLs

- “A programming language is low level when its programs require attention to the irrelevant.” (Perlis, Epigrams in Programming)
- Ensure quality by language design & engineering
- Use the best language for the job

Little languages vs general purpose

Raise level of abstraction; generate the implementation



Examples: forensics, tax administration

Find the right language to express the problem, instead of adapting the process to the solution