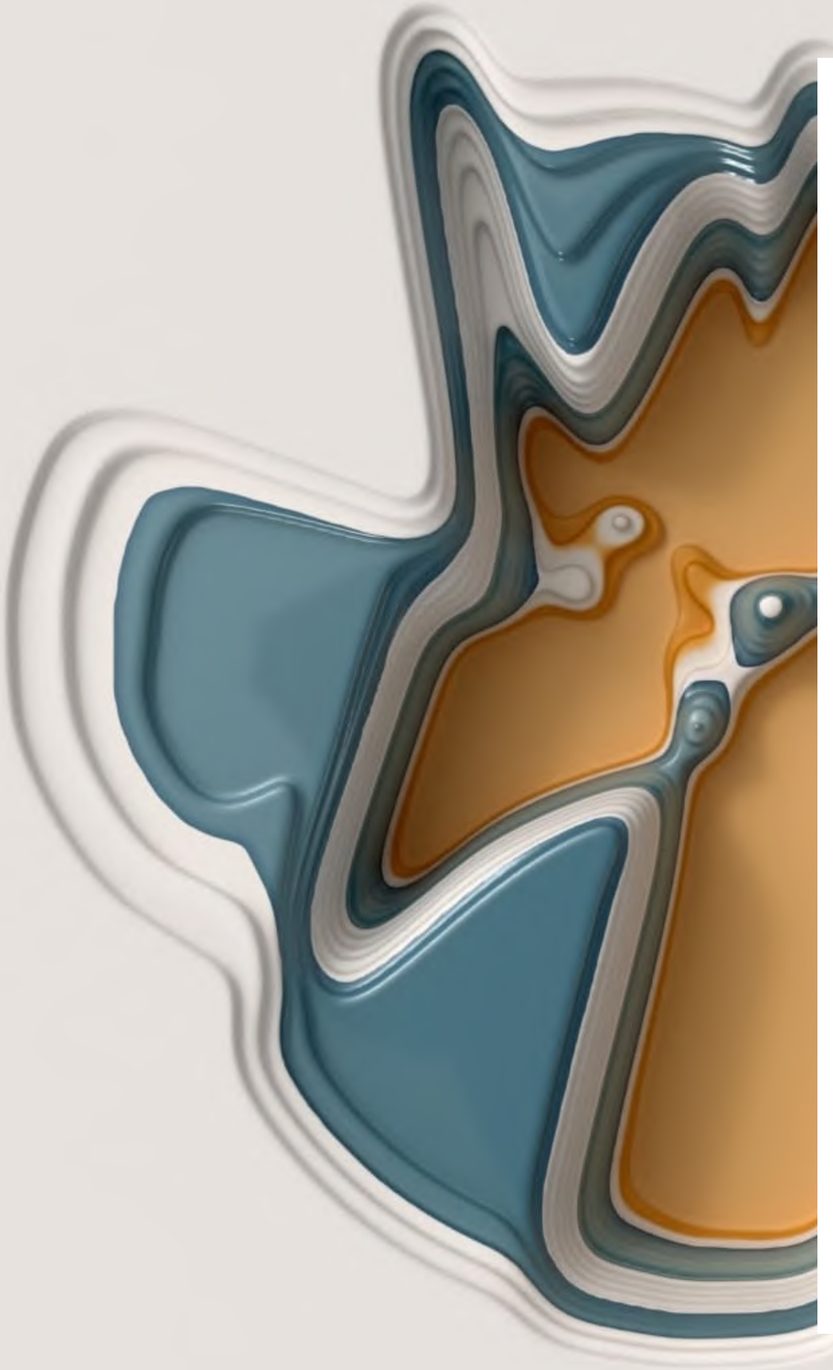


SOFTWARE ARCHITECTURE GATHERING / 17.11.2022

The Role of DSLs in Architecture Design

@active group

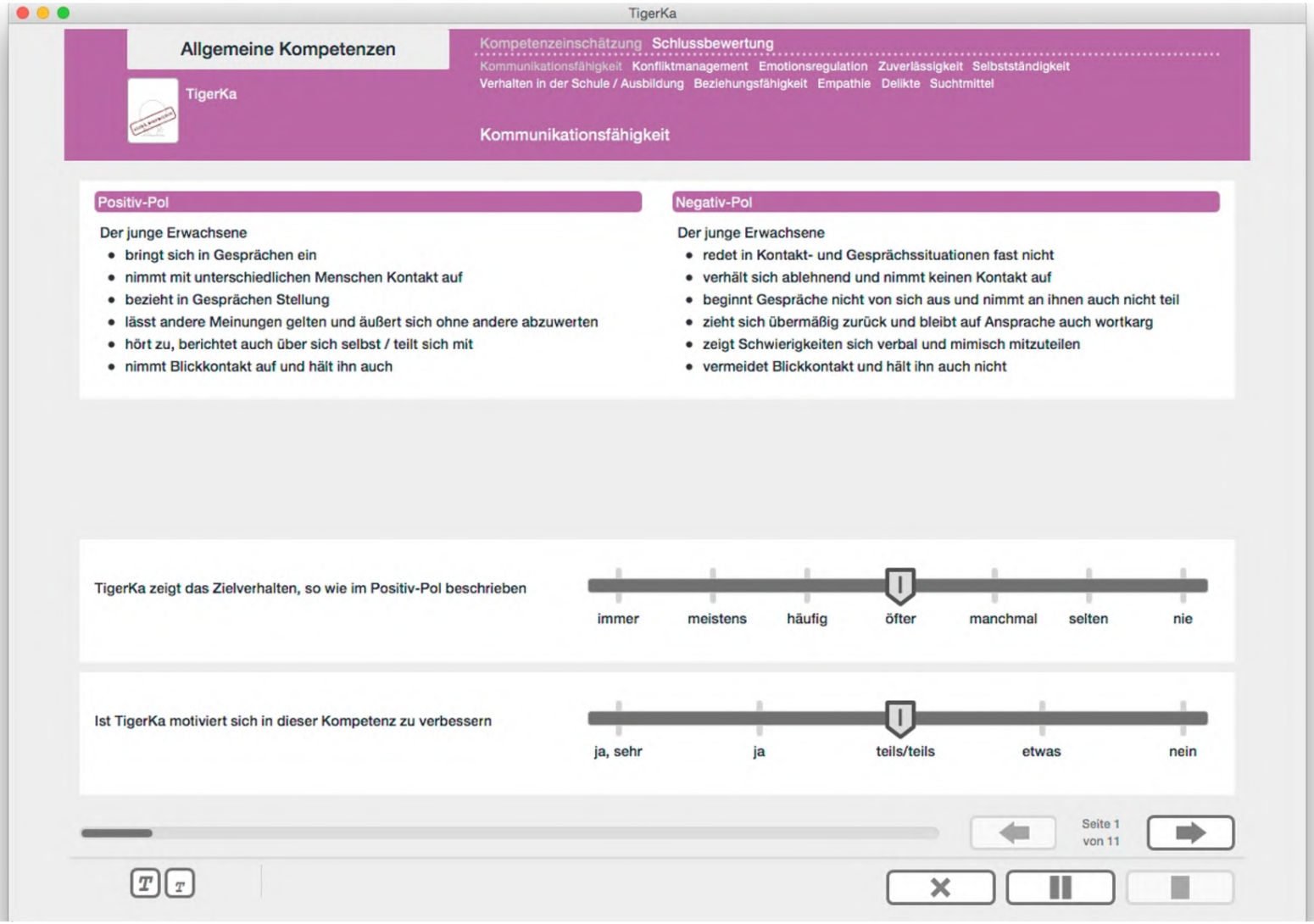
INNOQ

MIKE SPERBER
@sperbsen · he/him

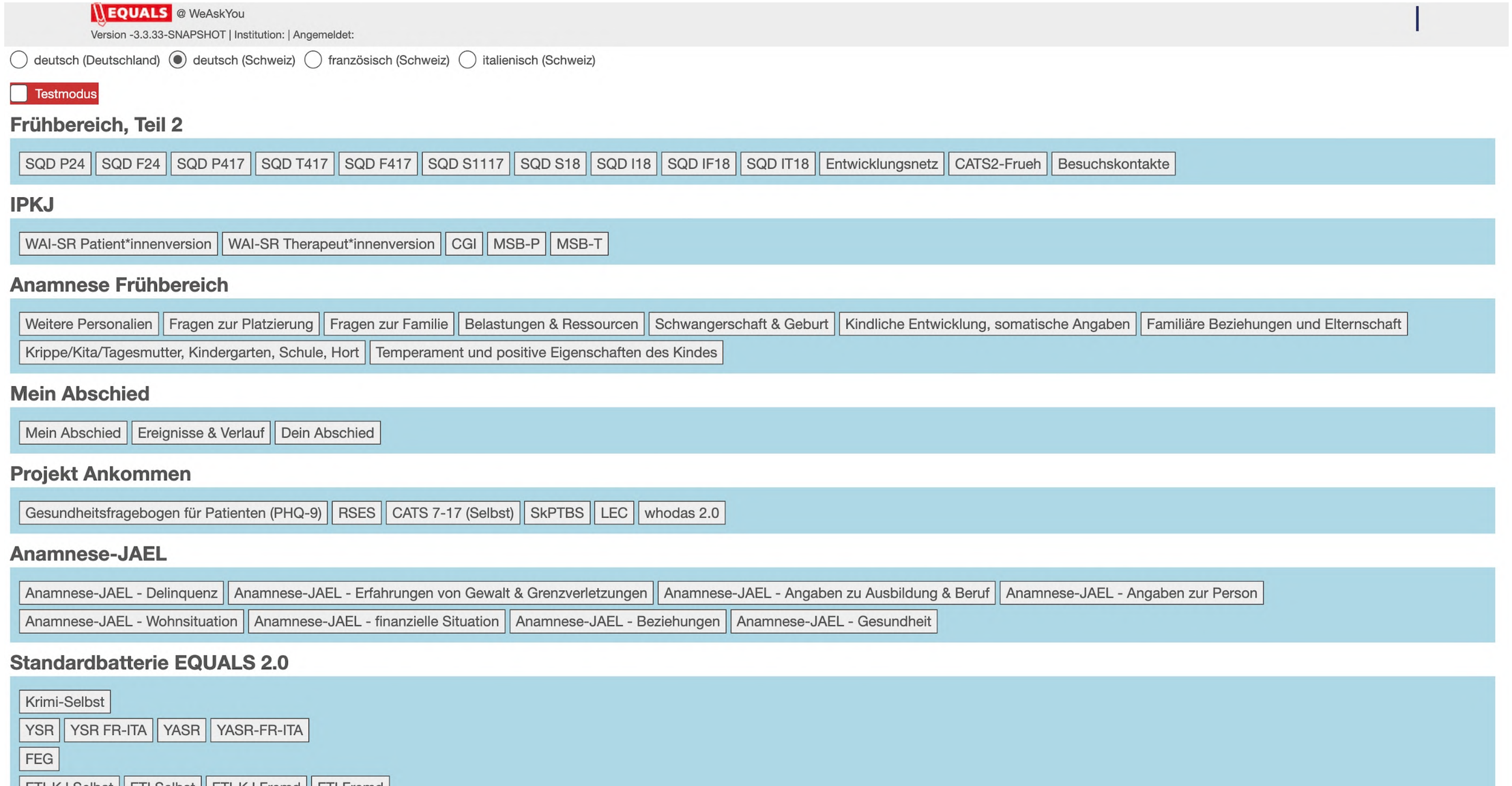
LARS HUPEL
@larsr_h · they/them

Domain-Specific Languages in the Wild

Questionnaires for Social Pedagogy



Questionnaires for Social Pedagogy

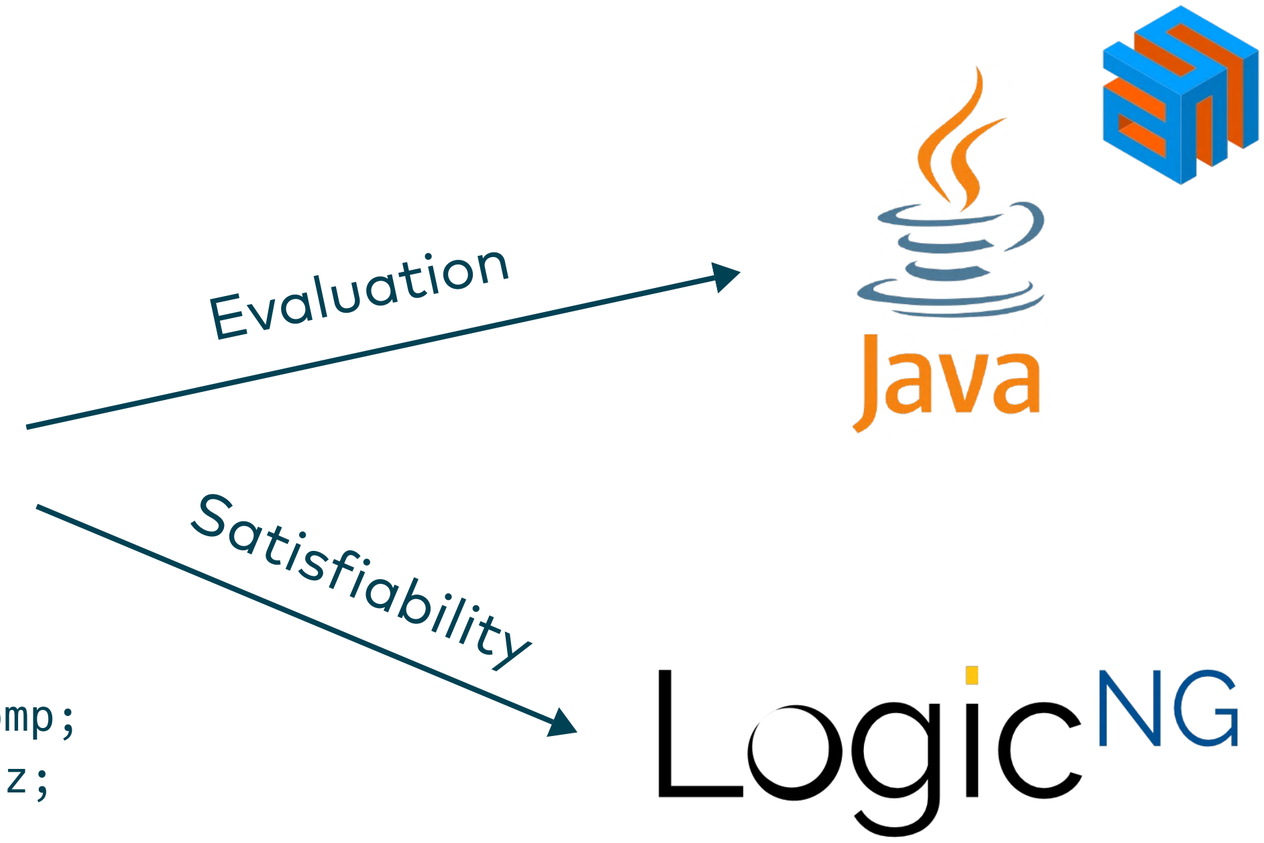


Questionnaire DSL

```
(defn verstoesse-editor
   [verstoesse date]
   (let [edit-verstoss (edit-verstoss-x verstoesse date)
         verstoesse-map (apply conj {} verstoesse)]
     (elements/within :verstoesse
                      (lists/list-editor
                       (lists/list-item-editor
                         (lists/with-add-button-editor "28px" edit-verstoss))
                       (lists/list-item (lists/list-column (loc/L "Gesetzesverstoss" loc/fra "Infractions à la loi"
                                                                                     loc/ita "Infrazione alla legge")
                                                           (lists/list-item-field
                                                            (elements/value-of (lens/++ (lens/pos 0) (lens/pos 2))
                                                                               (fn [[id sonstiges] options]
                                                                                 (dom/div
                                                                                  (if (= id sonstiger-verstoss-id)
                                                                                    sonstiges
                                                                                    (utils/spec options (verstoesse-map id)))))))
                                                           :width "auto")
                                        (lists/list-column txt/Datum
                                                           (lists/list-item-field (show-udate (lens/pos 1)))
                                                           :width "20%")
                                        bearbeiten-column
                                        löschen-column
                                        )))))
```

Car Configuration

```
declare x;
scope A [ y, z ] {
    l100 := 100;
    ltrue := true;
    lstra := "a";
    lstrb := "b";
    str_comp := lstra < lstrb;
    num_comp := l100 + 1 <= l100 + 1;
    f := x && !y;
    g := f && (z && ltrue);
    always_true := num_comp || !num_comp;
    g_equiv := x && (!y || !ltrue) && z;
}
```



AUTOSAR

```
<SWC-TO-ECU-MAPPING
   UUID="dc6b203d-f3b8-48a1-b1bb-ebe611639946”>
   <SHORT-NAME>abd8b270ad2074978a759aa82135687</SHORT-NAME>
    <COMPONENT-IREFS>
     <COMPONENT-IREF>
       <SOFTWARE-COMPOSITION-REF DEST="SOFTWARE-COMPOSITION">
         /TutorialProject/SYS_Indicator/SWA_Indicator
        </SOFTWARE-COMPOSITION-REF>
        <TARGET-COMPONENT-PROTOTYPE-REF
         DEST="COMPONENT-PROTOTYPE">
          /TutorialProject/SWA_Indicator/FrontLeftActuator
       </TARGET-COMPONENT-PROTOTYPE-REF>
      </COMPONENT-IREF>
   </COMPONENT-IREFS>
   <ECU-INSTANCE-REF DEST="ECU-INSTANCE">
     /TutorialProject/HWT_Indicator/FrontLeftIndicatorEcu
   </ECU-INSTANCE-REF>
 </SWC-TO-ECU-MAPPING>
```

AUTOSAR Bulk Importer

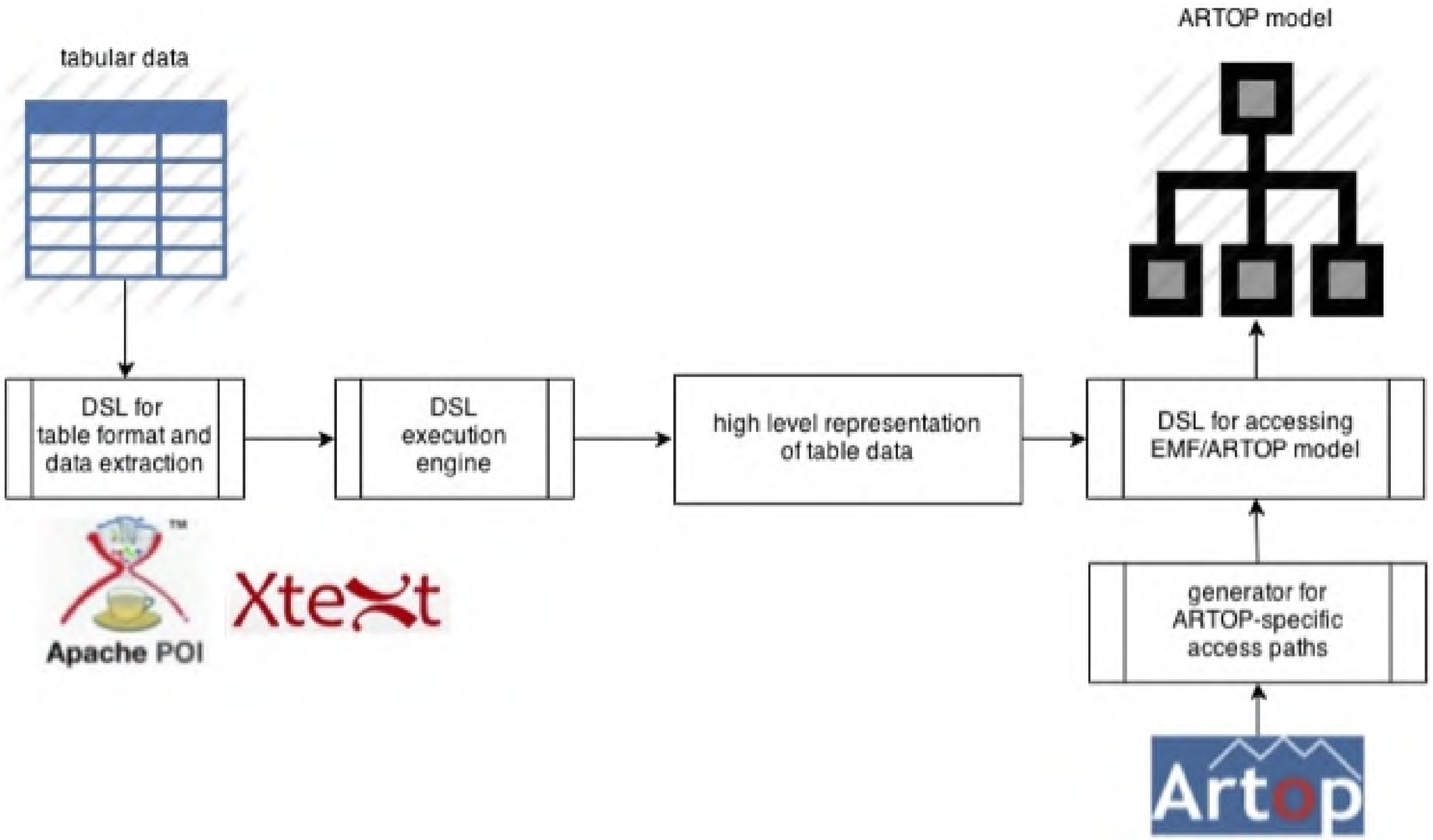


Table DSL

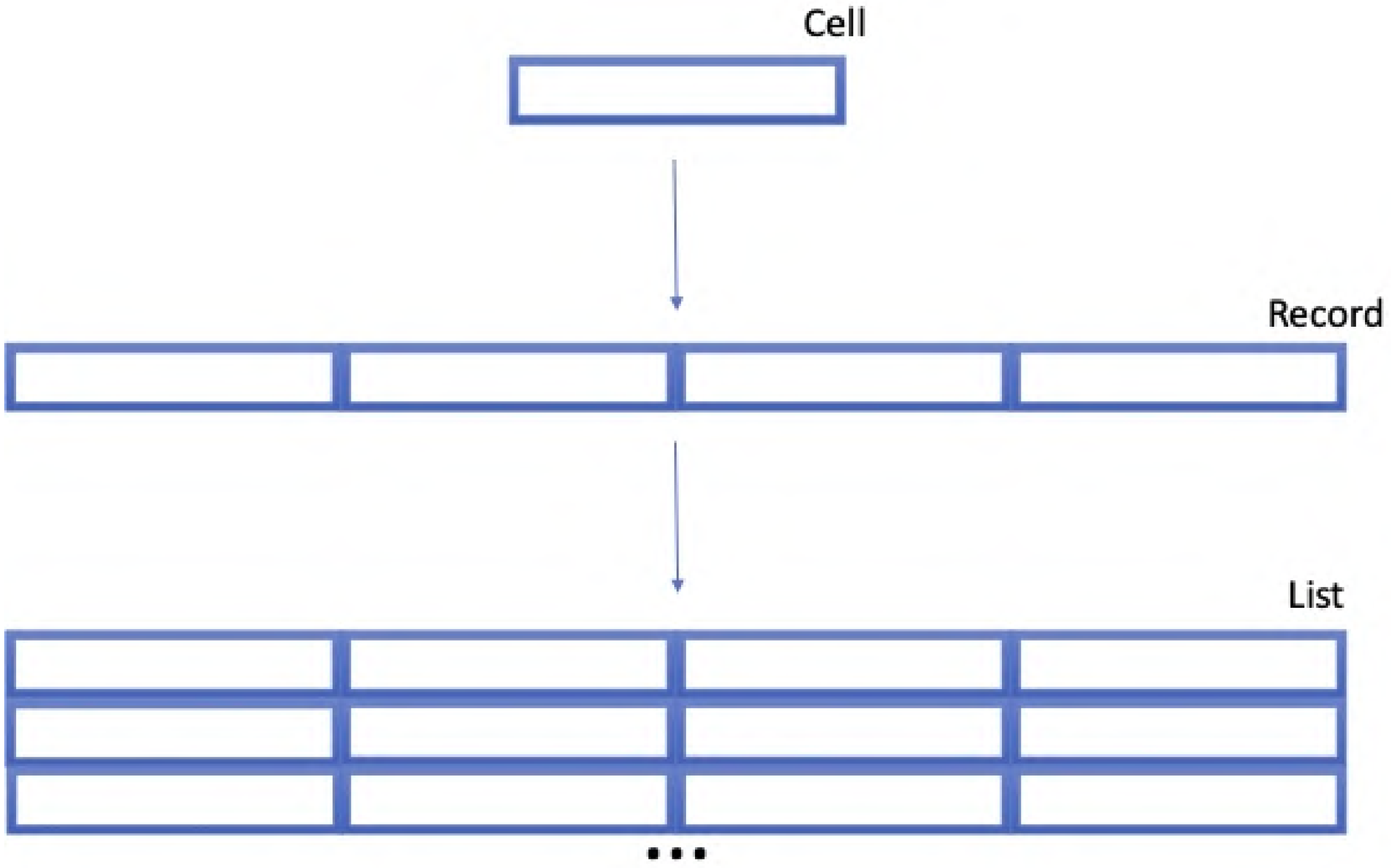
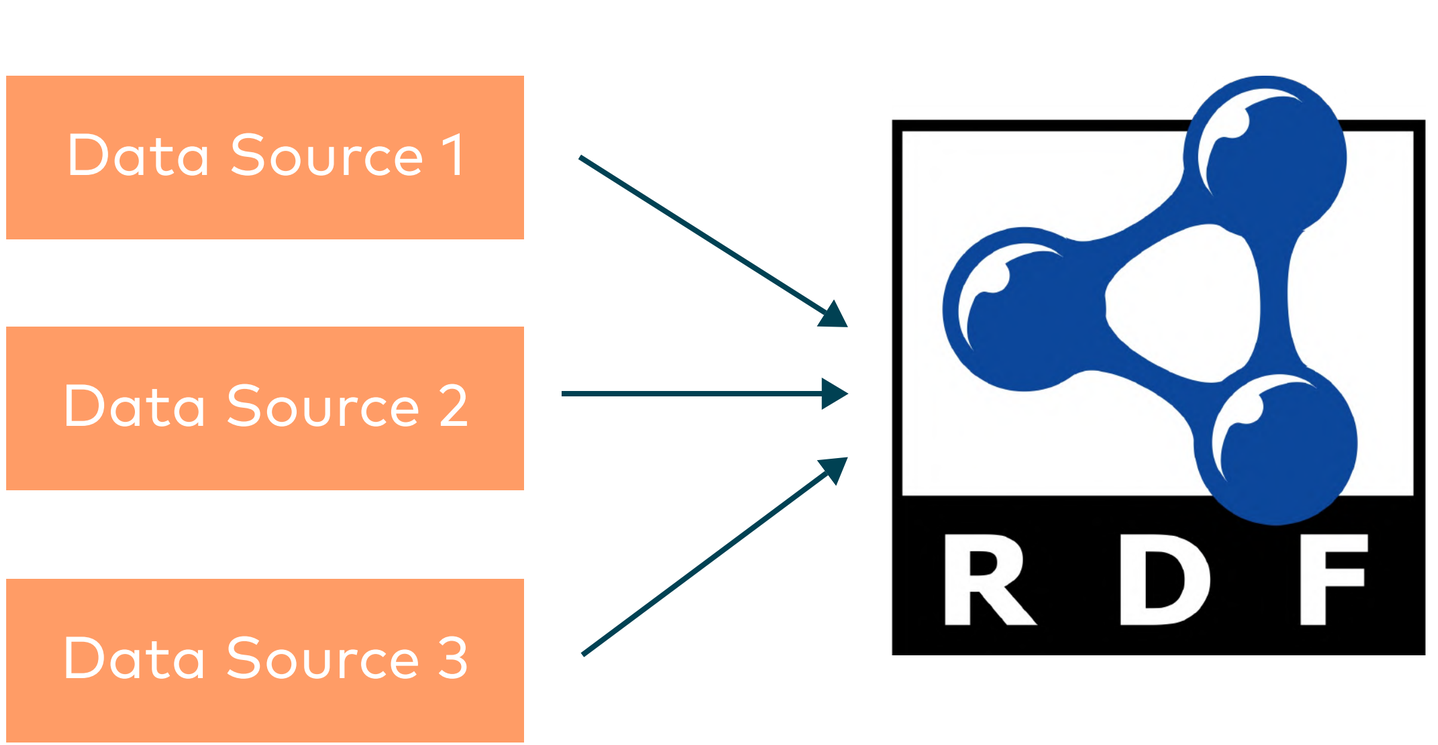


Table DSL

```
main = list-with-headers down
  [ @(0,0); @(0,1); @(0,2); @(0,3); @(0,4); @(0,5); ]
  [ osTask;
    positionInTask;
    activationOffset;
    kind;
    cpti;
    event;
  ]
  { Event2Task; };
```

Semantic Web Application



JS

- custom code (“plugins”) operating on data
- data container checks permissions
- dynamic tracing of information flows

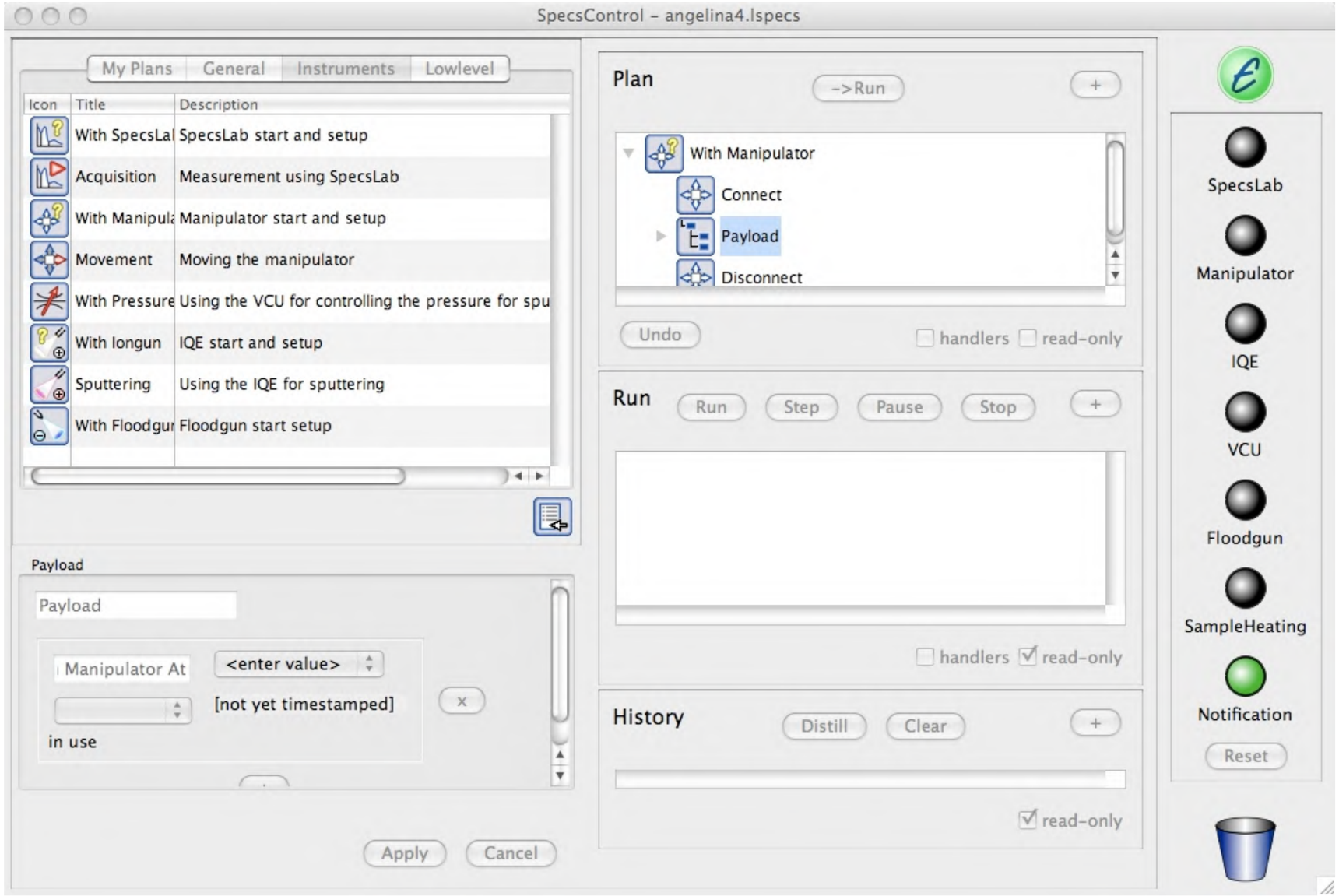
Semantic Web Application

```
const response = await this.pod.out.fetch("...");
// ...
const tweets = user.tweets.map(tweet => ({
    [Models.entityID]: { uri: `https://twitter.com/${user.screenName}/status/${tweet.id}` },
    datePublished: new Date(tweet.timestamp * 1000),
    author: personID,
    text: tweet.text
}));

await this.pod.in.add(
    ...Models.personModel.toQuads(person, this.pod.dataFactory),
    ...tweets.flatMap(tweet =>
        Models.socialMediaPostingModel.toQuads(tweet, this.props.pod.dataFactory)
    )
)
```

Surface Analysis

Surface Analysis Process DSL



You are already using DSLs!

- SQL
- HTML
- CSS
- PostScript
- PHP
- YAML
- Emacs Lisp
- LaTeX
- VBA
- DOT
- PlantUML
- Gherkin
- BPMN
- Frink

Where do DSLs fit?

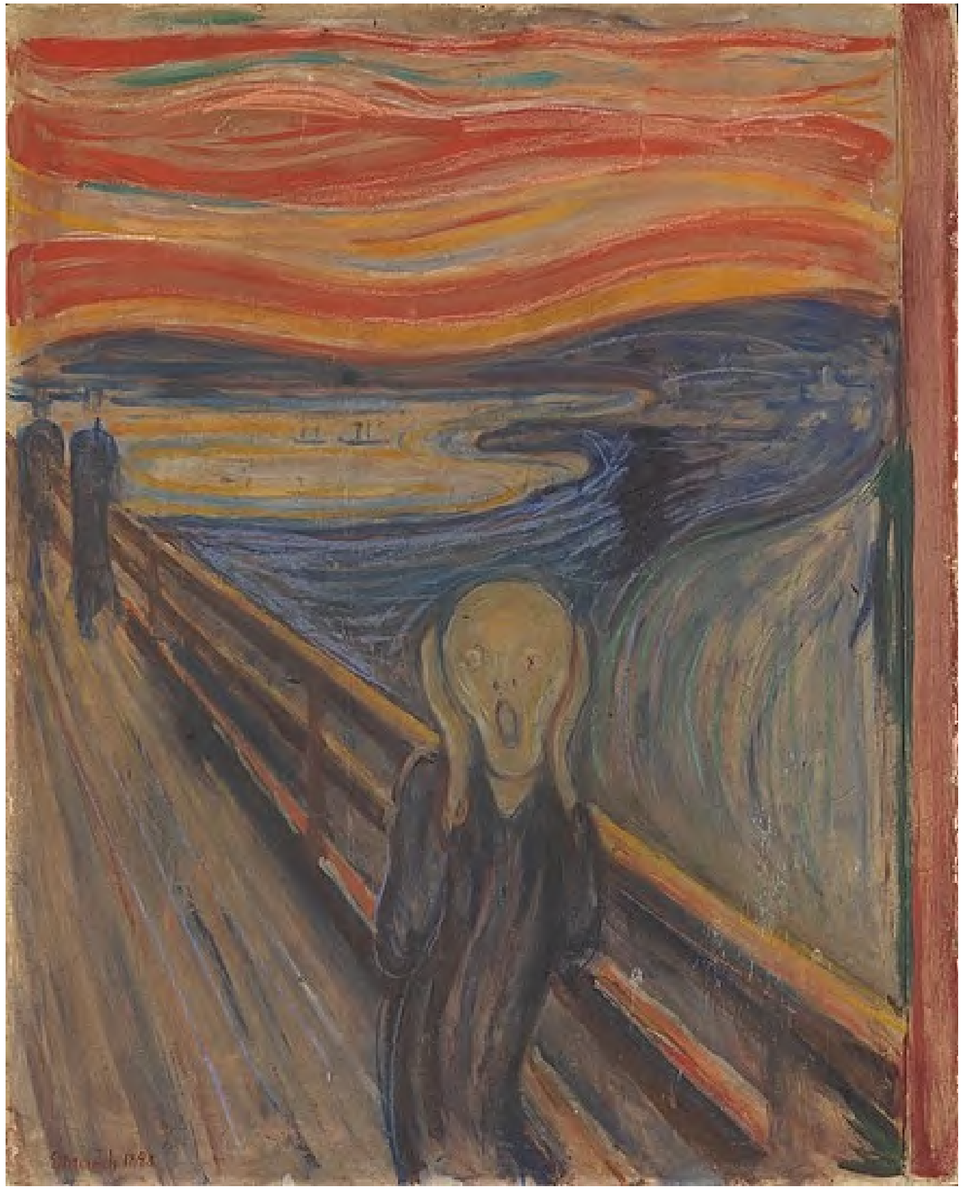
DSLs may be useful …

- for complex, user-defined flows
- when the “normal” implementation language is too flexible for analyzability
- … or not expressive enough to achieve readability
- in situations where common domain rules need to be understood across components

Designing & Implementing DSLs

DSLs are PLs

- syntax: lexing, parsing
- ASTs
- semantics
- code generation
- interpretation
- integration
- standard library
- …



Xtext to the Rescue!?

- Xtext is a stack of (Eclipse) tools
- takes care of a lot of the "hard parts"
  - lexer/parser generator based on ANTLR
  - AST based on Eclipse Modeling Framework
  - IDE support via Language Server Protocol
- general idea: define your grammar, get everything else ("batteries included")

Xtext: good, bad & ugly

- easy to get started … if you use Eclipse
- naming, scoping, syntax highlighting, … already taken care of
- write an ANTLR grammar and get everything for free … except
  - customizing the AST is hard!
  - automating the build (e.g. in Gradle) is even harder
  - apparently, there is no free lunch
- still have to write an interpreter or compiler!

Domain-Specific Languages Made Easy

Just one Feature ...

```
triangles =
        [ (a,b,c) | c <- [1..10], b <- [1..10], a <- [1..10] ]
rightTriangles =
        [ (a,b,c) |
          c <- [1..10],
          b <- [1..c],
          a <- [1..b],
          a^2 + b^2 == c^2 ]
```

Haskell

Syntax-as-a-Library

```
(require list-comprehensions)

(define triangles
  (|| (list a b c)
      (<- c (from-to 1 10))
      (<- b (from-to 1 10))
      (<- a (from-to 1 10))))

(define right-triangles
  (|| (list a b c)
      (<- c (from-to 1 10))
      (<- b (from-to 1 c))
      (<- a (from-to 1 b))
      (= (+ (sqr a) (sqr b)) (sqr c))))
```

Racket

Macros

```
(define-syntax ||
  (syntax-rules (<- let)
    ((|| e #t) (list e))
    ((|| e q) (|| e q #t))
    ((|| e (<- p l) Q ...)
     (let ((ok
            (lambda (p)
              (|| e Q ...))))
       (concat-map ok l)))
    ((|| e (let decls) Q ...)
     (let decls
       (|| e Q ...)))
    ((|| e b Q ...)
     (if b
         (|| e Q ...)
         '()))))
```

Alternative Execution Model

```
(define (grass-is-wet? rain sprinkler)
  (or (and (flip 0.9) rain)
      (and (flip 0.8) sprinkler)
      (flip 0.1)))

(define (grass-model)
  (let ((rain (flip 0.3))
        (sprinkler (flip 0.5)))
    (if (grass-is-wet? rain sprinkler)
        rain
        (fail))))
```

Again with Monads

```
grassIsWet rain sprinkler =
  do r <- flip 0.9
     s <- flip 0.8
     o <- flip 0.1
     return (r && rain || s && sprinkler || o)

grassModel =
  do rain <- flip 0.3
     sprinkler <- flip 0.5
     if grassIsWet rain sprinkler
     then return rain
     else fail
```

Host Languages for Embedded DSLs

- macros
  (Racket, Clojure, Elixir)
- monads
  (Haskell, Scala, F#, Kotlin)
- control abstraction
  (Racket, OCaml)
- overloading
  (Haskell, Scala, Kotlin)

Paths to DSL Design & Implementation

- embedded DSL
- DSL-as-a-library
- use prototyping tools (PLT Redex, Ott, ...)
- implementation tooling (Xtext, Spoofax, MPS, ...)
- generally: use existing concepts

Wadler's Law

Wadler's Law states that:

> In any language design, the total time spent discussing a feature in this list is proportional to two raised to the power of its position.
>
> 0. Semantics
> 1. Syntax
> 2. Lexical syntax
> 3. Lexical syntax of comments

https://wiki.haskell.org/Wadler%27s_Law

Greenspuns Tenth Rule Of Programming

This is a humorous observation once made by PhilipGreenspun:

> *Any sufficiently complicated C or Fortran program contains an ad-hoc, informally-specified, bug-ridden, slow implementation of half of CommonLisp.*

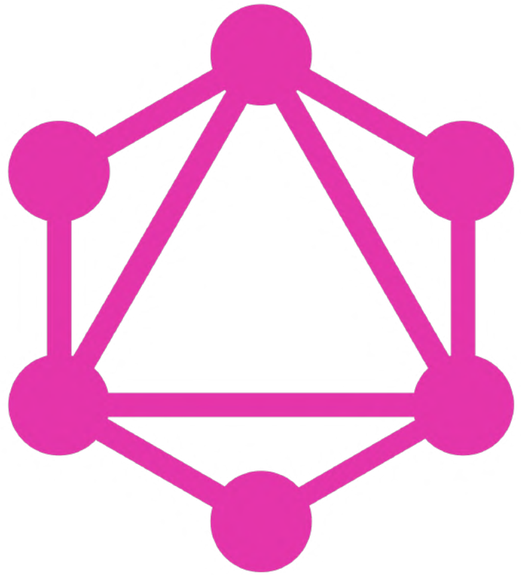
http://philip.greenspun.com/research/

http://wiki.c2.com/?GreenspunsTenthRuleOfProgramming

Domain-Specific Languages and Software Architecture

DSLs are really architecture

- example: GraphQL
- completely changes the way web-based applications are designed
  - REST model: exhibit resources with fixed semantics
  - GraphQL model: exhibit query language (~ SQL)



Why is this architecture?

- first and foremost a tool to manage complexity
- extremely useful together with domain-driven design
- empowers users
- foundation for quality goals such as adaptability and modifiability

DSL is a Major Decision

- lives for a long time
- mistakes in DSL design replicate indefinitely
- impacts building-block structure & deployment

```
#-------------------------------------------#
# Phase 9: switch to next logfile table     #
#-------------------------------------------#
IF del_logf_phase = 9 THEN

  del_logf_timer = -1  # Action done, stop timer
  _next = 0
  WHILE del_logf_no < 20 AND NOT _next
    del_logf_no = del_logf_no + 1
    IF del_logf_no < 20 THEN
      _next = (del_logf_avail [del_logf_area][del_
    ENDIF
  WEND
  IF _next THEN
    del_logf_phase = 3
  ELSE
    del_logf_no = 0
     next = 0
```

iSAQB curriculum "DSL"

Work in progress (not yet approved):

https://github.com/isaqb-org/curriculum-dsl

A data structure is just a stupid programming language.

—Bill Gosper

Q&A

Mike Sperber
michael.sperber@active-group.de
@sperbsen

Lars Hupel
lars.hupel@innoq.com
@larsr_h

www.active-group.de

INNOQ

www.innoq.com