iSAQB Advanced DSL - Case Study

Michael Sperber

Created: 2024-07-02 Tue 16:35

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>
```

Spreadsheet Tables

Segment	Country	Units Sold	Manuf. Price	Sale Price	Sales	P
Government	Canada	1618	$3,00	$20,00	$32.370,00	$
Government	Germany	1321	$3,00	$20,00	$26.420,00	$
Midmarket	France	2178	$3,00	$15,00	$32.670,00	$
Midmarket	Germany	888	$3,00	$15,00	$13.320,00	$
Midmarket	Mexico	2470	$3,00	$15,00	$37.050,00	$

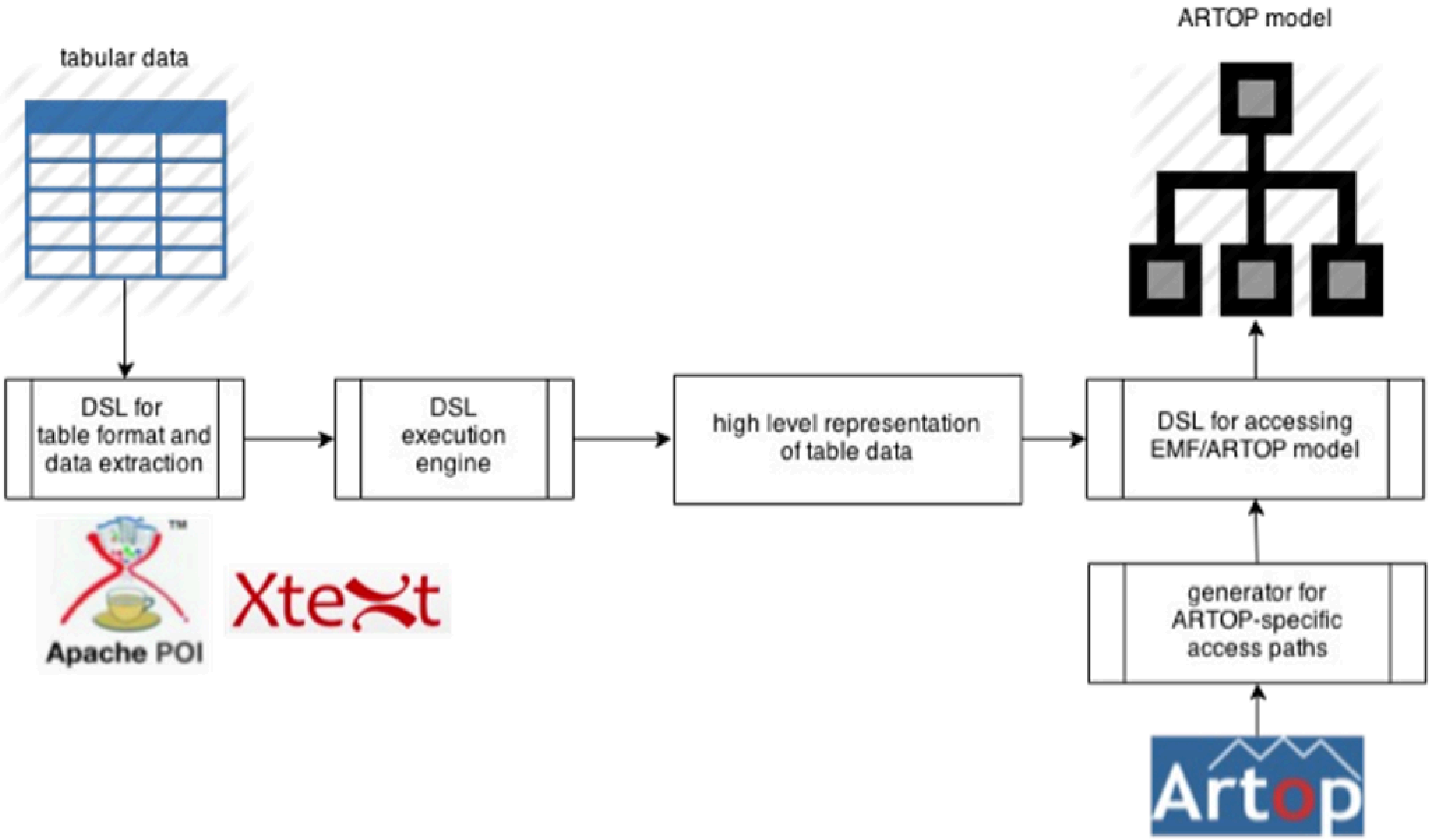
AUTOSAR Bulk Importer

An large automotive OEM maintains their own AUTOSAR tooling. Frequently, AUTOSAR models contain data originally maintained as an Excel table. The formats of these tables are custom, and new formats keep being added, but many tables with a common format are imported into AUTOSAR models.

(LG 1-3) Exercise: Domain Analysis

- Perform basic context mapping for the AUTOSAR bulk importer.
- Identify bounded contexts that may be amenable for a DSL.
- What are criteria that make a bounded context amenable for a DSL?

AUTOSAR Bulk Importer


tabular data
ARTOP model
DSL for table format and data extraction
DSL execution engine
high level representation of table data
DSL for accessing EMF/ARTOP model
generator for ARTOP-specific access paths
Apache POI
Xtext
Artop

Apache POI

```
Sheet sheet = workbook.getSheetAt(0);

for (Row row : sheet) {
    for (Cell cell : row) {
        switch (cell.getCellType()) {
            case STRING: ... break;
            case NUMERIC: ... break;
            case BOOLEAN: ... break;
            case FORMULA: ... break;
            default: ...
        }
    }
}
```

Exercise: Table Structure

How would you describe the information in the example table?

Design data types for representing that information!

Case Study

Overall Goal:

Design a DSL for describing the structure of a spreadsheet table, along with an implementation that parses that structure into appropriate data types.

Exercise: DSL Approaches

- How would you design and implement such a DSL?
- What general approaches could you formulate for DSL design and implementation?
- What skills do these approaches require, not found among general software developers / architects?

(LG 1-2) Embedded vs. Stand-Alone

Embedded: create library of functions, functions, macros, reuse *host language*

Stand-alone: read program into separate program that reads and executes

Exercise: What are the trade-offs of the respective approaches?

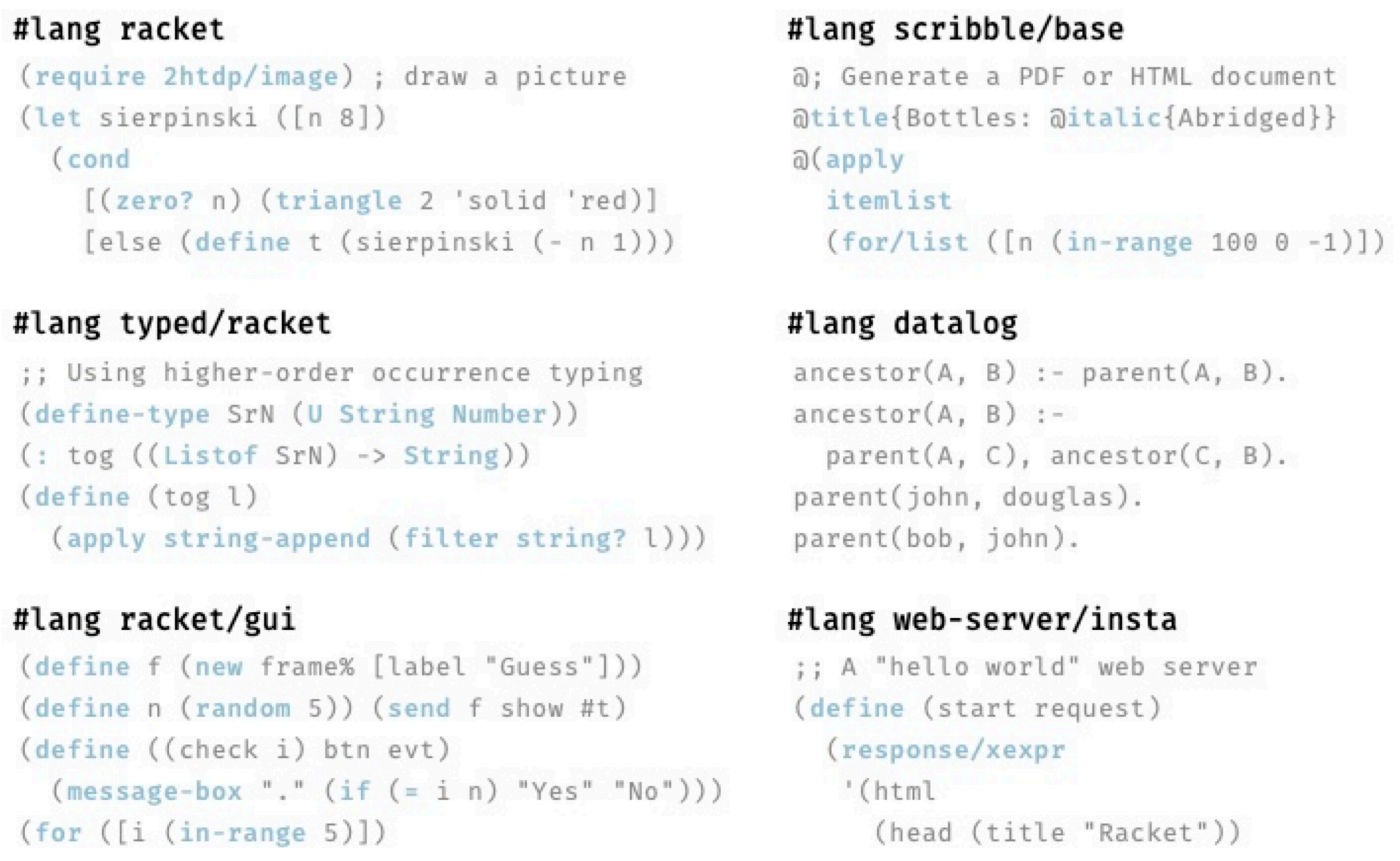
(LG 1-2, LG 1-4, LG 3-1) Interpreted vs. Compiled

Interpreter: represent DSL program as data, execute by traversing the data structure

Compiler: translate DSL program to a *target language*, use existing implementation of that

Exercise: Can you see these two approaches as endpoints of a spectrum?

(LG 5-2) Approach: Embedded DSL in Racket



```
#lang racket
(require 2htdp/image) ; draw a picture
(let sierpinski ([n 8])
  (cond
    [(zero? n) (triangle 2 'solid 'red)]
    [else (define t (sierpinski (- n 1)))
```

```
#lang scribble/base
@; Generate a PDF or HTML document
@title{Bottles: @italic{Abridged}}
@(apply
  itemlist
  (for/list ([n (in-range 100 0 -1)])
```

```
#lang typed/racket
;; Using higher-order occurrence typing
(define-type SrN (U String Number))
(: tog ((Listof SrN) -> String))
(define (tog l)
  (apply string-append (filter string? l)))
```

```
#lang datalog
ancestor(A, B) :- parent(A, B).
ancestor(A, B) :-
  parent(A, C), ancestor(C, B).
parent(john, douglas).
parent(bob, john).
```

```
#lang racket/gui
(define f (new frame% [label "Guess"]))
(define n (random 5)) (send f show #t)
(define ((check i) btn evt)
  (message-box "." (if (= i n) "Yes" "No")))
(for ([i (in-range 5)])
```

```
#lang web-server/insta
;; A "hello world" web server
(define (start request)
  (response/xexpr
   '(html
     (head (title "Racket"))
```

Racket

- development system, since 1995
- orginally made for teaching beginners
- basis: Scheme, functional programming
- "language-oriented programming"

(LG 4-3) Exercise: Combinators

Design a data representation for shower products, along with a function that computes the proportion of soap in a shower product!

A shower product can be one of the following:

- soap
- shampoo
- shower gel consisting of soap and shampoo, in equal parts

Generalize shower gels to arbitrary mixtures!

(LG 4-3) Closure of Operations

"**Where it fits, define an operation whose return type is the same as the type of its argument(s).** If the implementer has state that is used in the computation, then the implementer is effectively an argument of the operation, so the argument(s) and return value should be of the same type as the implementer. Such an operation is closed under the set of instances of that type. A closed operation provides a high-level interface without introducing any dependency on other concepts."

Eric Evans, *Domain-Driven-Design*, 2003.

(LG 4-3) Compositionality

"The meaning of a complex expression is determined by the meanings of its constituent expressions and the rules used to combine them."

Wikipedia: Principle of compositionality

(LG 4-3) Language Design and Compositionality

"Every powerful language has three mechanisms for accomplishing this:

- primitive expressions, which represent the simplest entities the language is concerned with,
- means of combination, by which compound elements are built from simpler ones, and
- means of abstraction, by which compound elements can be named and manipulated as units"

Abelson, Sussman: Structure and Interpretation of Computer Programs, second edition

Assisted Exercise: Combinators for Spreadsheet Tables

Design a combinator library for describing the format of spreadsheet tables and implement it in Racket or Java!

(LG 1-4, LG 2-1) DSL or Just a Library?

```
(struct entry (name address phone)
  #:transparent)

(define right (direction 1 0))
(define down (direction 0 1))

(define entry-row
  (record-format entry right (list text text number)))
(define entries
  (list-format down entry-row))
(define heading
  (ignore right
          (list (constant-format "Name")
                (constant-format "Address")
                (constant-format "Phone"))))
(define addressbook
  (choose 2 down (list heading entries)))
```

Assisted Exercise: Parsing Spreadsheet Tables

Implement a Racket or Java parser for spreadsheet tables that, given a table format and a table, returns a structured representation of the table contents.

(Represent the table by a function that accepts coordinates and returns the cell contents as a string.)

(LG 1-2, LG 3-2) Static vs. Dynamic

Which parts of a Tim program could be executed at compile time (*statically*), which ones at run time (*dynamically*)?

(LG 3-1) Partial Evaluation

$$\begin{array}{rclll}
\texttt{program} & : & \texttt{input}_1 & \times\,\texttt{input}_2 & \rightarrow \texttt{output} \\
\texttt{mix} & : & \texttt{program} & \times\,\texttt{input}_1 & \rightarrow (\texttt{input}_2 \rightarrow \texttt{output}) \\
\texttt{interpret} & : & \texttt{program} & \times\,\texttt{input} & \rightarrow \texttt{output}
\end{array}$$

(LG 3-1) Futamura Projections

$$\begin{array}{lrcl}
(1) & \texttt{mix}(\texttt{interpret}, \texttt{program}) & = & \texttt{program}\ \text{(compiled)} \\
(2) & \texttt{mix}(\texttt{mix}, \texttt{interpret}) & = & \texttt{compiler} \\
(3) & \texttt{mix}(\texttt{mix}, \texttt{mix}) & = & \texttt{compiler generator}
\end{array}$$