goto; conference 2012

- made for developers by developers...proudly presenting the best speakers & brightest attendees

Thoughts on Polyglotism

Stefan Tilkov | @stilkov | stefan.tilkov@innoq.com

innoQ

7 Theses

1. Language Equality

Languages are Not Equal

Machine Code

Assembler

C

C++

Java

Scala

Python

Ruby

Scheme/Lisp

Haskell

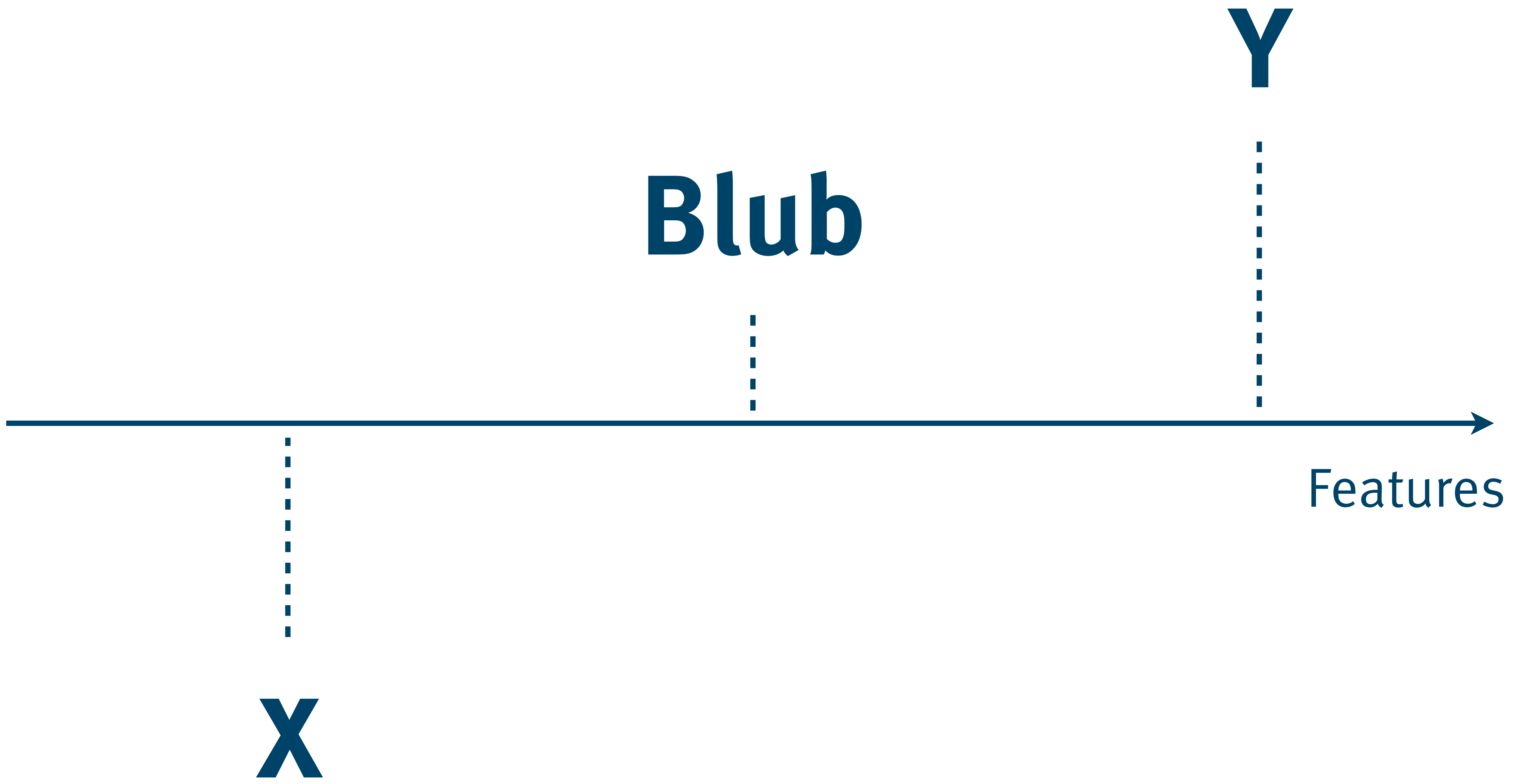
Sapir-Whorf

> “We cut nature up, organize it into concepts, and ascribe significances as we do, largely because we are parties to an agreement to organize it in this way — an agreement that holds throughout our speech community and is codified in the patterns of our language.

Whorf, Benjamin (John Carroll, Editor) (1956). Language, Thought, and Reality: Selected Writings of Benjamin Lee Whorf. MIT Press. “Sapir-Whorf Hypothesis” (note: now disputed); see also http://en.wikipedia.org/wiki/Sapir-Whorf_hypothesis

“Blub”

(Full of unnecessary complexity)



Y

Blub

Features

X

(Hopelessly limited)

> Blub falls right in the middle of the abstractness continuum... As long as our hypothetical Blub programmer is looking down the power continuum, he knows he's looking down. Languages less powerful than Blub are obviously less powerful, because they're missing some feature he's used to.
>
> But when our hypothetical Blub programmer looks in the other direction, up the power continuum, he doesn't realize he's looking up. What he sees are merely weird languages... Blub is good enough for him, because he *thinks* in Blub.

Paul Graham, “Beating the Averages”
http://www.paulgraham.com/avg.html

Differences

Paradigm

Speed

Verbosity

Ceremony

Stability

Learning Curve

Type System

> To be quite honest, most Javalanders are blissfully unaware of the existence of the other side of the world.

Steve Yegge

http://steve-yegge.blogspot.com/2006/03/execution-in-kingdom-of-anouns.html

A Little Bit of Java ...

```java
package com.example;

import java.util.List;
import java.util.Arrays;
import java.util.Collections;
import java.util.Comparator;

public class SortList {
    public static void main(String[] args) {
        List<String> list = Arrays.asList("Shamelessly", "Stolen",
                                          "From", "Ola", "Bini");
         Collections.sort(list, new Comparator<String>() {
            public int compare(String first, String second) {
                return first.length() - second.length();
            }
        });

        String sep = "";
        for (String name : list) {
            System.out.print(sep);
            System.out.print(name);
            sep = ", ";
        }

        System.out.println();
    }
}
```

... vs. Ruby

```
list = ["Shamelessly", "Stolen", "From", "Ola", "Bini"]
puts list.sort_by(&:length).join(', ')
```

http://youtube.com/watch?v=PfnP-8XbJao

```
class Project < ActiveRecord::Base
    belongs_to :portfolio
    has_one :project_manager
    has_many :milestones
    has_and_belongs_to_many :categories
end
```

JavaScript/Node.js

```
var sys = require("sys"), http = require("http"), url = require("url"),
    path = require("path"),  fs = require("fs");

var dir = process.argv[2] || './public';
var port = parseFloat(process.argv[3]) || 8080;
sys.log('Serving files from ' + dir + ', port is ' + port);

http.createServer(function(request, response) {
    var uri = url.parse(request.url).pathname;
    var filename = path.join(process.cwd(), dir, uri);
    path.exists(filename, function(exists) {
        if(exists) {
            fs.readFile(filename, function(err, data) {
                response.writeHead(200);
                response.end(data);
            });
        } else {
            sys.log('File not found: ' + filename);
            response.writeHead(404);
            response.end();
        }
    });
}).listen(port);
```

Clojure

```
(ns sample.grep
  "A simple complete Clojure program."
  (:use [clojure.contrib.io :only [read-lines]])
  (:gen-class))

(defn numbered-lines [lines]
  (map vector (iterate inc 0) lines))

(defn grep-in-file [pattern file]
  {file (filter #(re-find pattern (second %)) (numbered-lines (read-lines file)))})

(defn grep-in-files [pattern files]
  (apply merge (map #(grep-in-file pattern %) files)))

(defn print-matches [matches]
  (doseq [[fname submatches] matches, [line-no, match] submatches]
    (println (str fname ":" line-no ":" match))))

(defn -main [pattern & files]
  (if (or (nil? pattern) (empty? files))
    (println "Usage: grep <pattern> <file...>")
    (do
      (println (format "grep started with pattern %s and file(s) %s"
                         pattern (apply str (interpose ", " files))))
      (print-matches (grep-in-files (re-pattern pattern) files))
      (println "Done."))))
```

There's more to life than objects

Data structures vs. objects

```
public class Point {
    private final double x;
    private final double y;

    public Point(double x, double y) {
        this.x = x;
        this.y = y;
    }
}

Point p1 = new Point(3, 4);
```

```
(def p1 [3 4])
```

Data structures vs. objects

```
(def p1 [3 4])
```

Immutable

Reusable

Compatible

Data structures vs. objects

```java
import static java.lang.Math.sqrt;

public class Point {
    private final double x;
    private final double y;

    public Point(double x, double y) {
        this.x = x;
        this.y = y;
    }

    public double distanceTo(Point other) {
        double c1 = other.x - this.x;
        double c2 = other.y - this.y;
        return sqrt(c1 * c1 + c2 * c2);
    }
}
```

Data structures vs. objects

```
(import-static java.lang.Math sqrt)

(defn distance
  [[x1 y1] [x2 y2]]
  (let [c1 (- x2 x1)
        c2 (- y2 y1)]
    (sqrt (+ (* c1 c1) (* c2 c2)))))
```

Data structures vs. objects

```
(defn rand-seq [limit]
  (repeatedly #(rand-int limit)))
```

infinite randoms

```
(take 10 (partition 2 (rand-seq 10)))
```

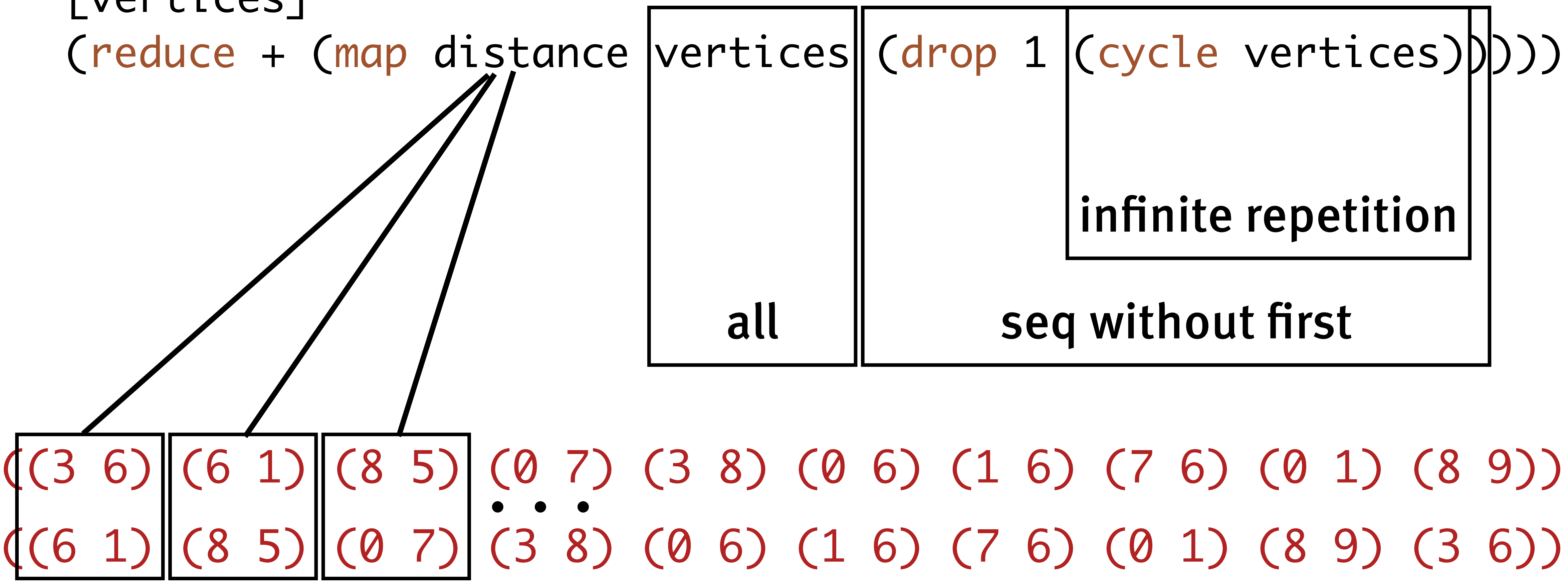
pairs of random ints

10 random points

```
;((3 6) (6 1) (8 5) (0 7) (3 8) (0 6) (1 6) (7 6) (0 1) (8 9))
```

Data structures vs. objects

```
(defn circumference
  [vertices]
  (reduce + (map distance vertices (drop 1 (cycle vertices)))))
```



all

seq without first

infinite repetition

```
;((3 6) (6 1) (8 5) (0 7) (3 8) (0 6) (1 6) (7 6) (0 1) (8 9))
                          . . .
;((6 1) (8 5) (0 7) (3 8) (0 6) (1 6) (7 6) (0 1) (8 9) (3 6))


;58.06411369758525
```

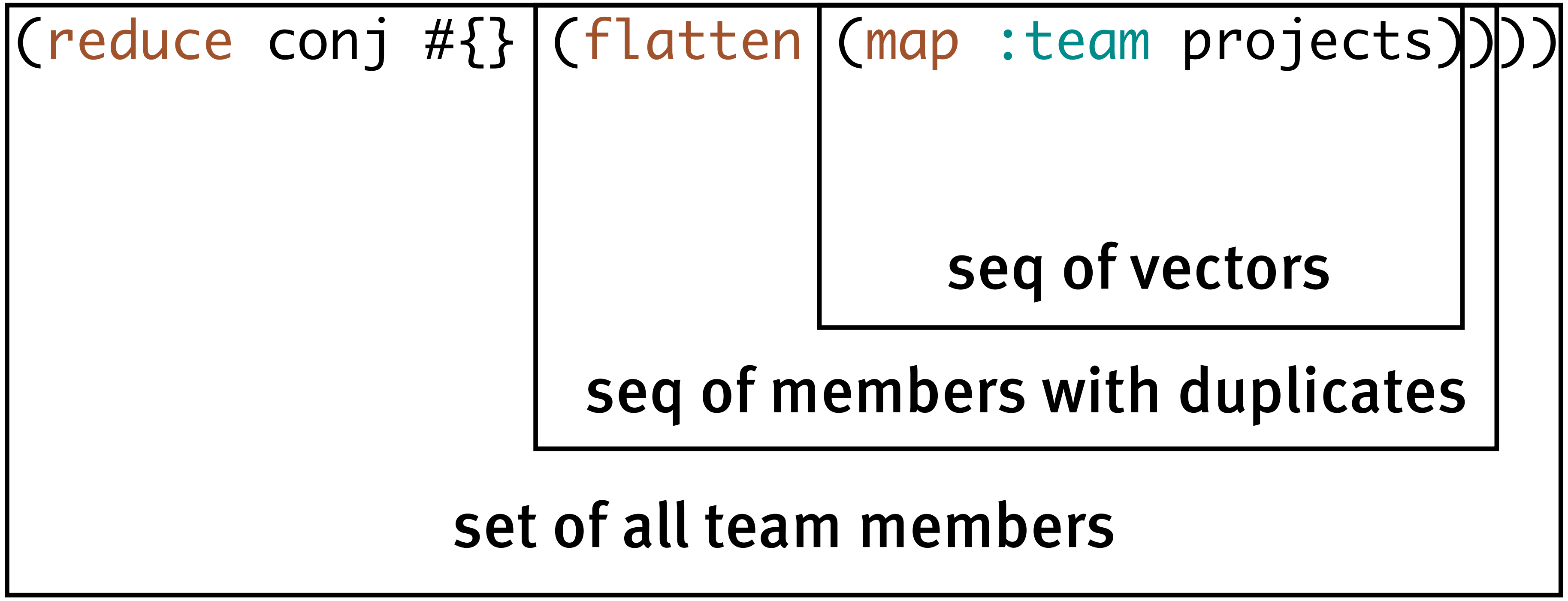
```
assoc
assoc-in
butlast
concat
conj
cons
count
cycle
difference
dissoc
distinct
distinct?
drop-last
empty
empty?
every?
filter
first
flatten
group-by
interleave
interpose
intersection
into
join
lazy-cat
mapcat
merge
merge-with
not-any?
not-empty?
not-every?
nth
partition
partition-all
partition-by
peek
pop
popy
project
remove
replace
rest
rseq
select
select-keys
shuffle
some
split-at
split-with
subvec
take
take-last
take-nth
take-while
union
update-in
```

Maps

```
(def projects #{{:id "1",
                 :kind :time-material,
                 :description "Consulting for BigCo",
                 :budget 25000,
                 :team [:joe, :chuck, :james]}
                {:id "2",
                 :kind :fixed-price,
                 :description "Development for Startup",
                 :budget 100000,
                 :team [:john, :chuck, :james, :bill]}
                {:id "3",
                 :kind :fixed-price,
                 :description "Clojure Training",
                 :budget 3000,
                 :team [:joe, :john]}})
```

Map access

```
(defn all-members
  [projects]
```



```
  (reduce conj #{} (flatten (map :team projects))))
```

seq of vectors

seq of members with duplicates

set of all team members

```
(all-members projects)
;#{:chuck :joe :james :john :bill}
```

Map access & coupling

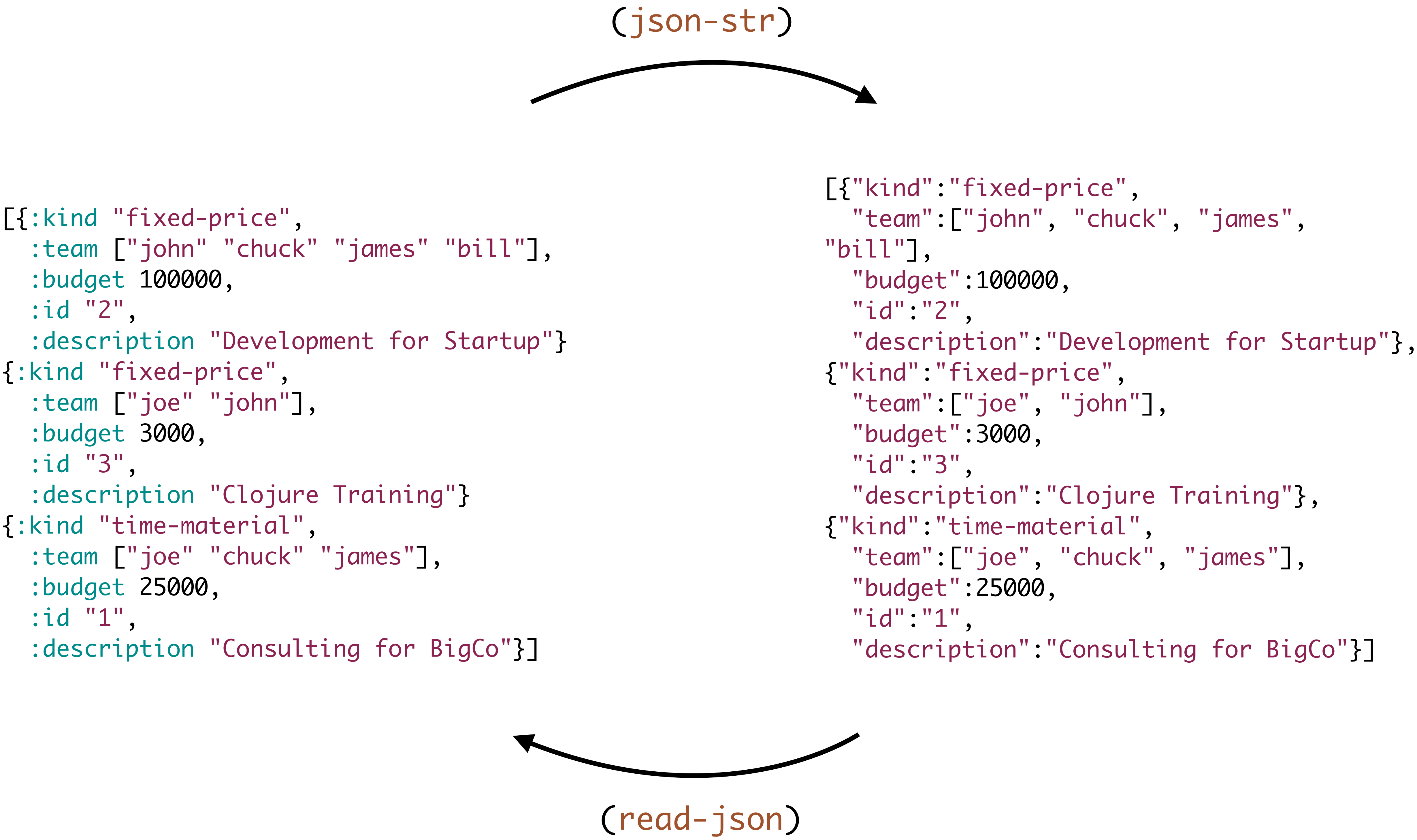
```
(defn all-members
  [projects]
  (reduce conj #{} (flatten (map :team projects))))
```

```
#{{:id "2",
   :kind :fixed-price,
   :description "Development for Startup",
   :budget 100000,
   :team [:john, :chuck, :james, :bill]}}
```

Map access & coupling

```
(defn all-members
  [projects]
  (reduce conj #{} (flatten (map :team projects))))
```

```
 #{{:id "2",
    :kind :fixed-price,
    :description "Development for Startup",
    :budget 100000,
    :team [:john, :chuck, :james, :bill]}}
```



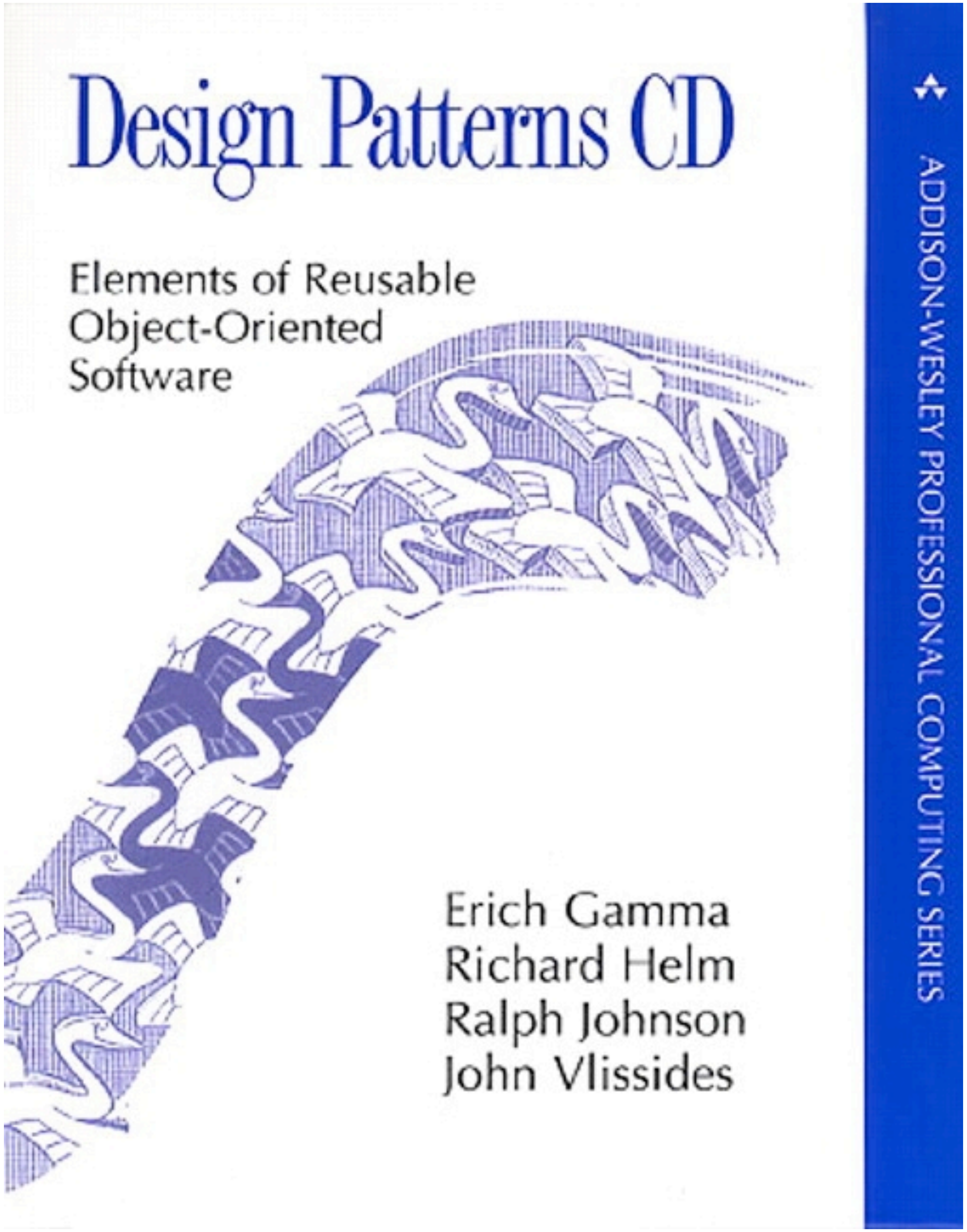
(json-str)

```
[{:kind "fixed-price",
  :team ["john" "chuck" "james" "bill"],
  :budget 100000,
  :id "2",
  :description "Development for Startup"}
 {:kind "fixed-price",
  :team ["joe" "john"],
  :budget 3000,
  :id "3",
  :description "Clojure Training"}
 {:kind "time-material",
  :team ["joe" "chuck" "james"],
  :budget 25000,
  :id "1",
  :description "Consulting for BigCo"}]
```

```
[{"kind":"fixed-price",
  "team":["john", "chuck", "james",
"bill"],
  "budget":100000,
  "id":"2",
  "description":"Development for Startup"},
 {"kind":"fixed-price",
  "team":["joe", "john"],
  "budget":3000,
  "id":"3",
  "description":"Clojure Training"},
 {"kind":"time-material",
  "team":["joe", "chuck", "james"],
  "budget":25000,
  "id":"1",
  "description":"Consulting for BigCo"}]
```

(read-json)

Design Patterns are a Design Smell



Design Patterns CD

Elements of Reusable
Object-Oriented
Software

Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides

ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

> Patterns are signs of weakness in programming languages.

Mark Dominus, “Design Patterns of 1972”,
http://blog.plover.com/2006/09/11/

Patterns & Dynamic Languages

First-class types	AbstractFactory, Flyweight, FactoryMethod, State, Proxy, Chain-of-Responsibility
First-class functions	Command, Strategy, TemplateMethod, Visitor
Macros	Interpreter, Iterator
Method Combination	Mediator, Observer
Multimethods	Builder, (Visitor)
Modules	Facade

http://norvig.com/design-patterns/

Project Example 1

Case tool model export/import

XMI -> transformation -> Java API

Use of a Java Lisp dialect (SISC)

XML libraries from Java

:-)

Yet …

… languages don’t matter

HTTP

URIs

HTML

CSS

Clients

Web Servers

Caching Proxies

CDN

Atom/RSS

Databases

1. Language Equality

Languages differ drastically

2. Ecosystems

Development Environment

Libraries

Runtime Environment

Community

Culture

Project Example 2

Large scale banking application

Java / J2EE

Numerous external DSLs, implemented in Haskell

Exchange of binary data w/ Java

Excel/sed/awk/Haskell/Java pipeline

:-(

2. Ecosystems

No language is an island

3. Multi-language Platforms

.NET

JVM

.NET	JVM
C#	Java
VB	JRuby
F#	Scala
IronPython	Groovy
IronRuby	Clojure
DLR	Java 7

Development Environment

Libraries

Runtime Environment

Community

Culture

Development Environment

Community

Culture

Libraries

Runtime Environment

Project Example 2

Environmental information system

JRuby/Rails

Use of Java graphics libraries

:-)

3. Multi-language platforms

MLVMs enable diversity

4. Polyglot Programming

Polyglot Programming

vs.

Polyglot Programmer

polyglot *(noun)*: a person who knows and is able to use several languages

Question:
What language do you use?

Question:

What languages do you use?

sh XSLT cmd

SQL XML BPEL

CSS

HTML EL Ant

JavaScript

The right tool ...

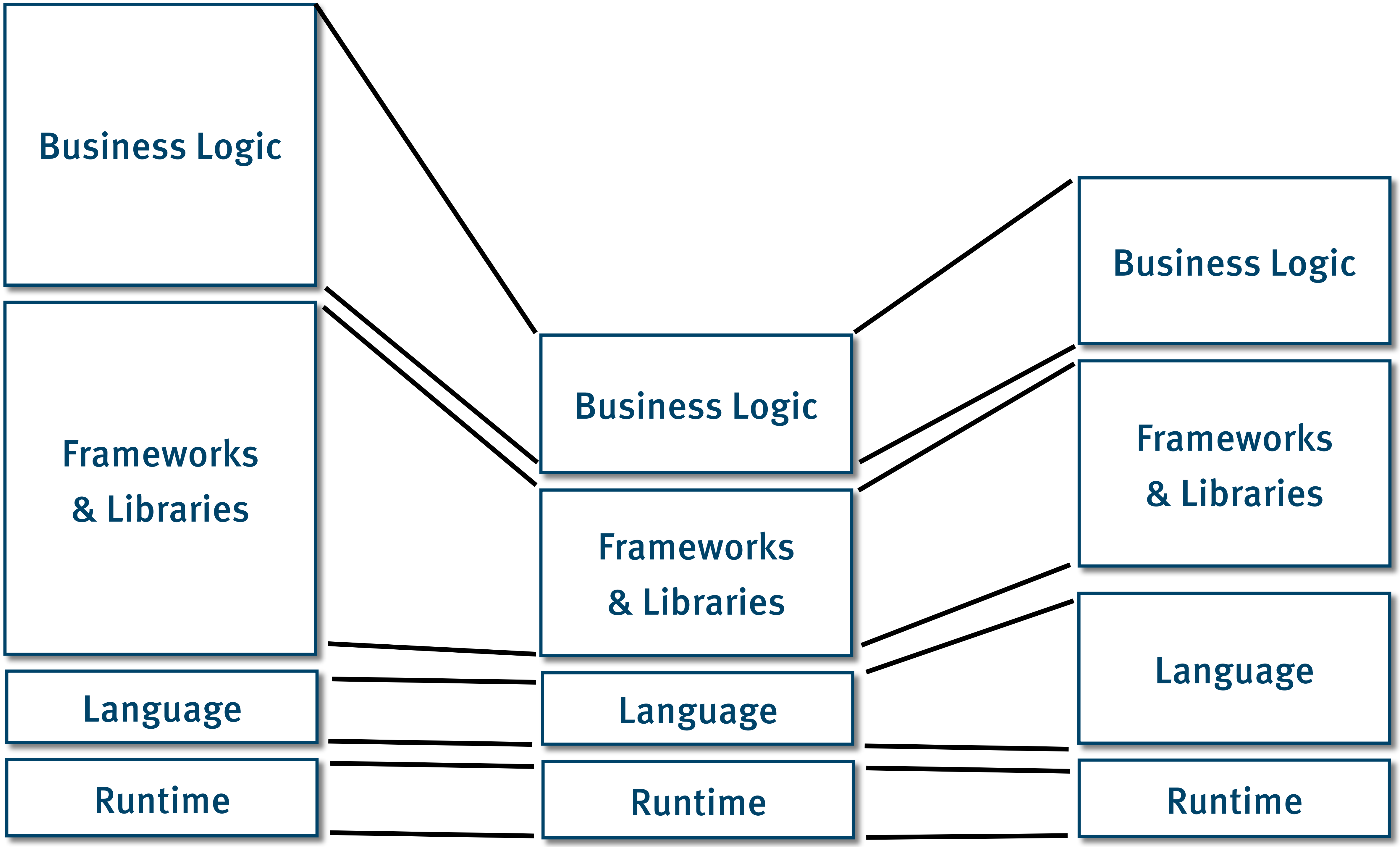
mon·o·cul·ture |ˈmänəˌkəlCHər|

n.

1. The cultivation of a single crop on a farm or in a region or country.

2. A single, homogeneous culture without diversity or dissension.

100% PURE
JAVA™


Business Logic
Frameworks & Libraries
Language
Runtime
Business Logic
Frameworks & Libraries
Language
Runtime
Business Logic
Frameworks & Libraries
Language
Runtime

Example Criteria

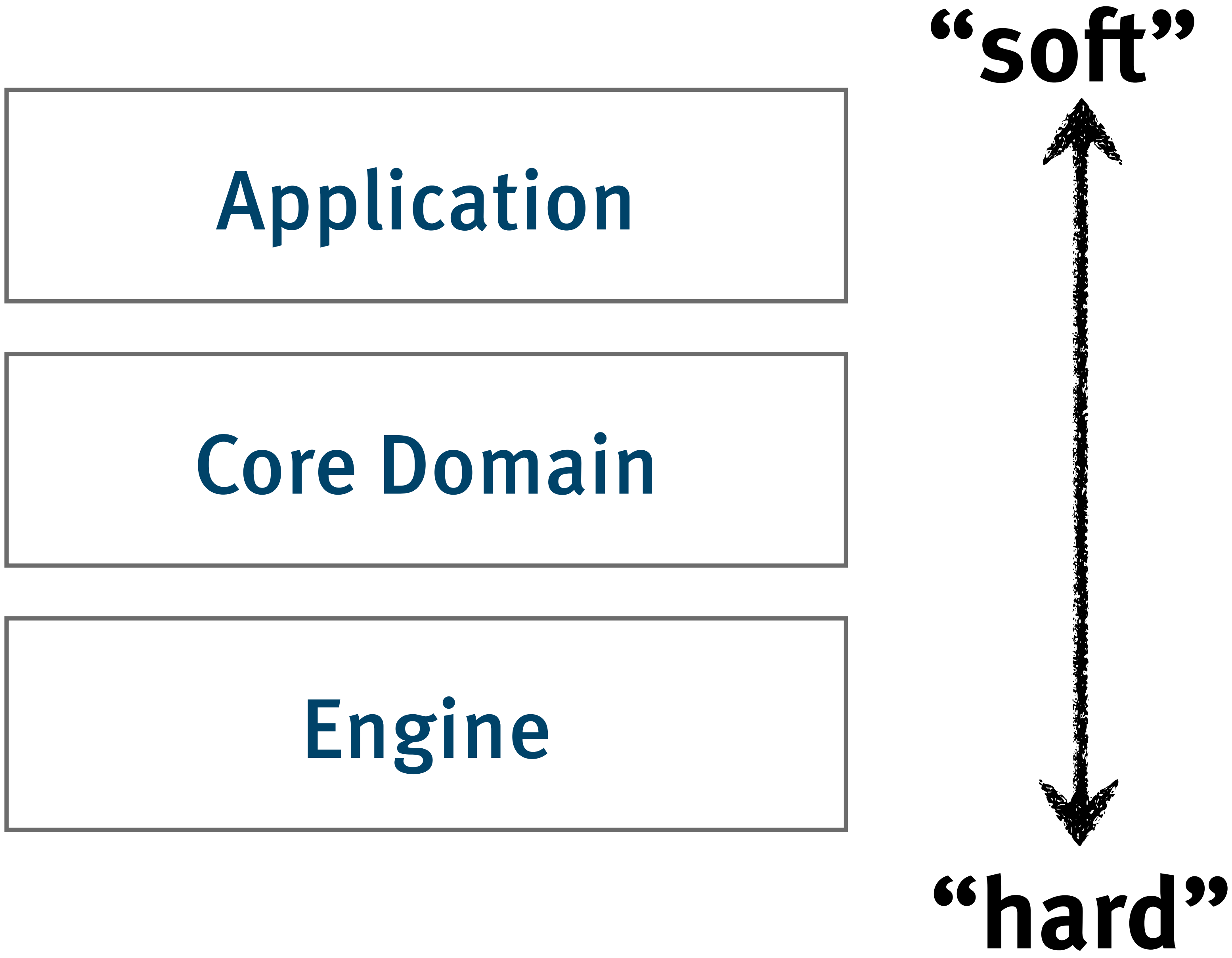
Java	Mainstream
Erlang	Distributed systems, 24x7
Clojure	Complex algorithms, concurrency
Ruby	Productivity

4. Polyglot programming

Nobody **uses a single language ...**

... nor should they.

5. Stability Layers



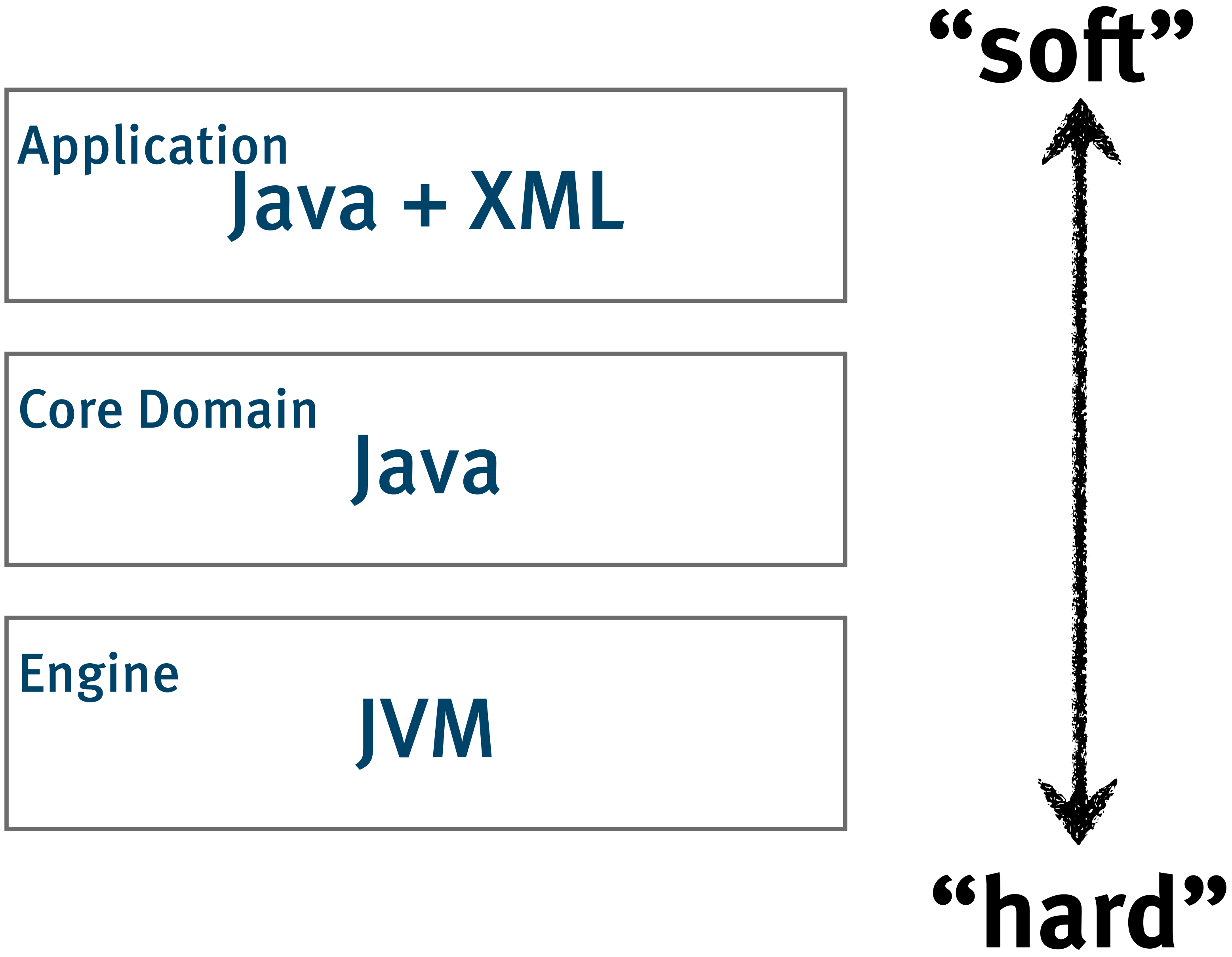
“soft”

Application

Core Domain

Engine

“hard”



“soft”

Application

Java + XML

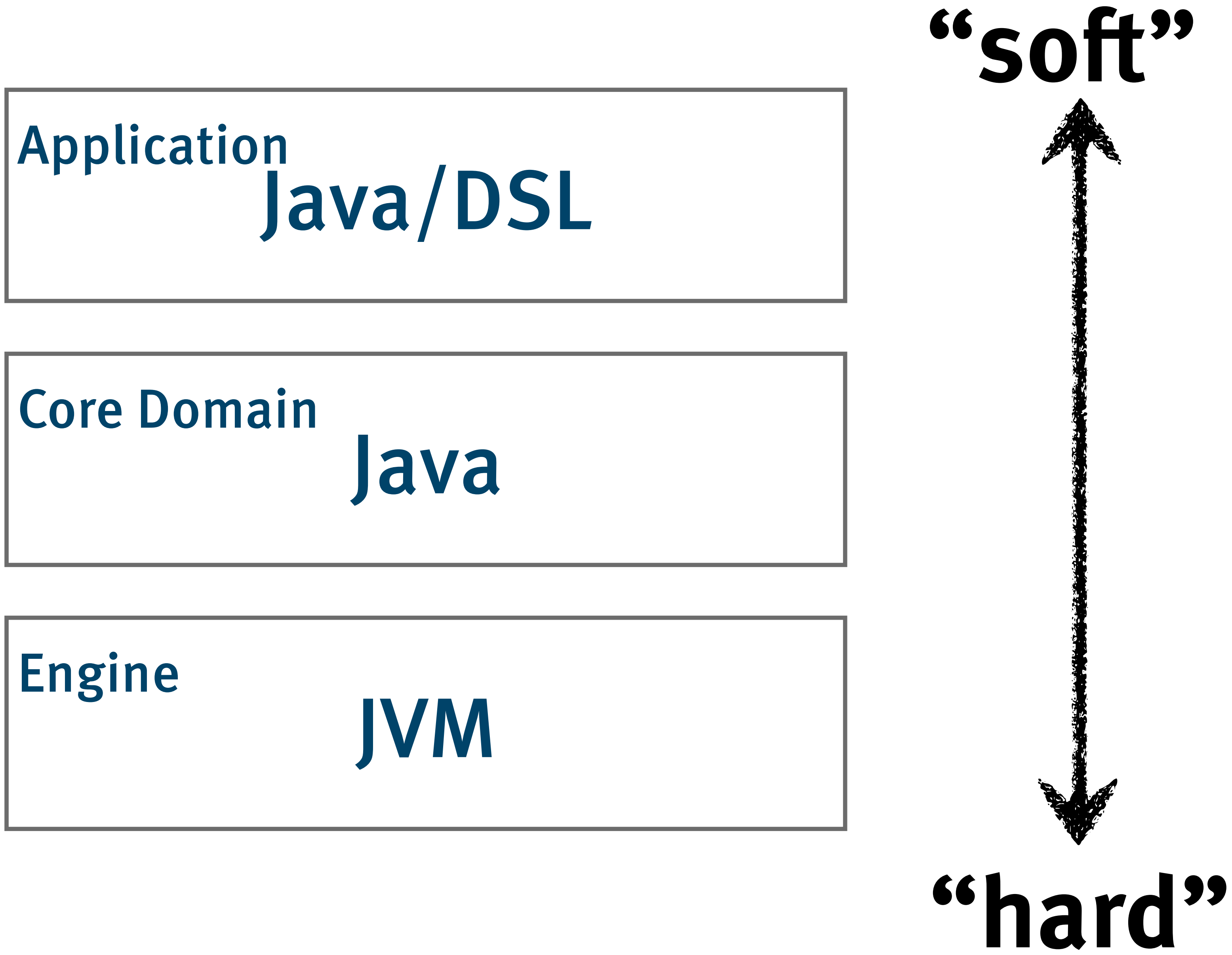
Core Domain

Java

Engine

JVM

“hard”



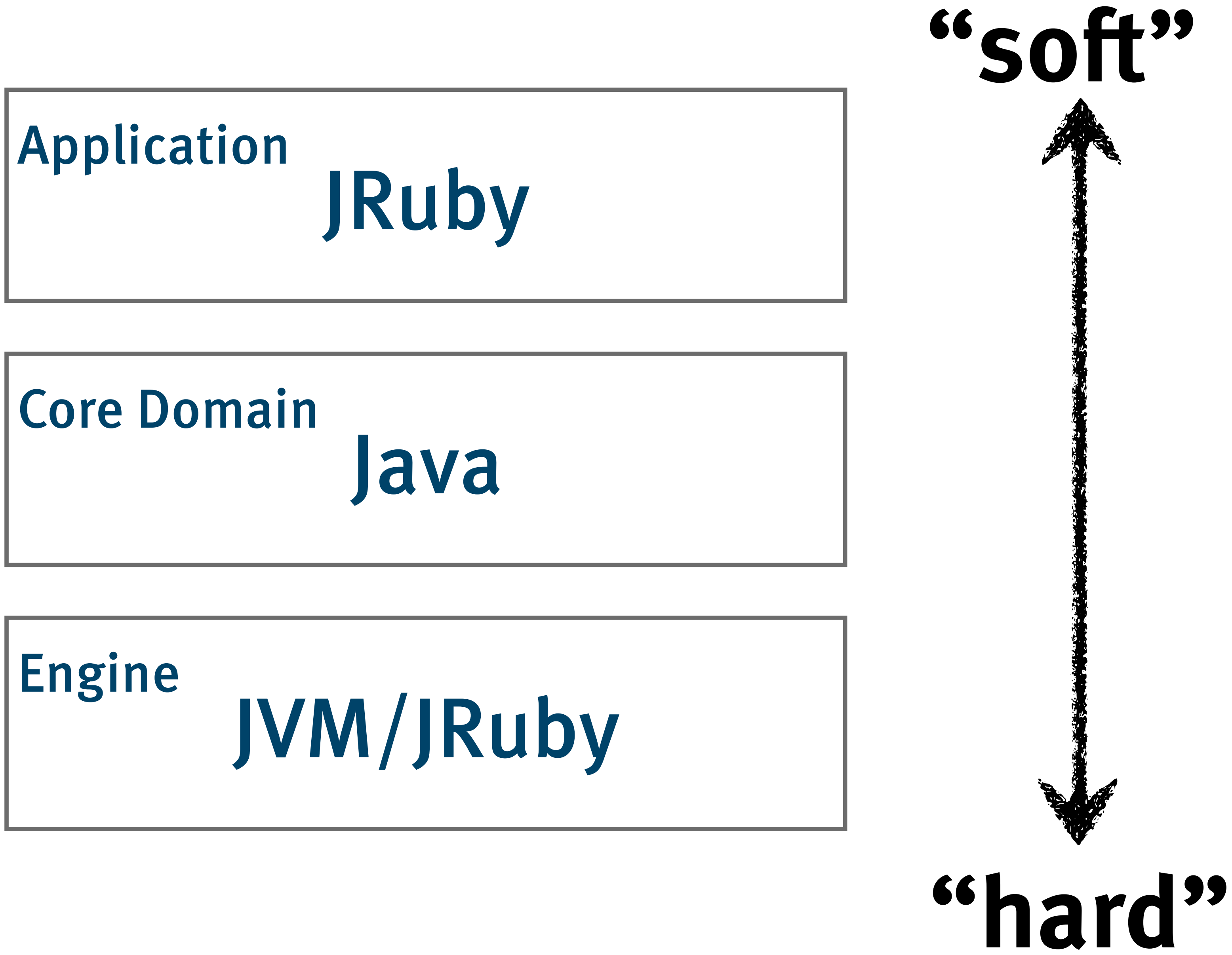
Application
Java/DSL

Core Domain
Java

Engine
JVM

“soft”

“hard”



“soft”

Application JRuby

Core Domain Java

Engine JVM/JRuby

“hard”

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

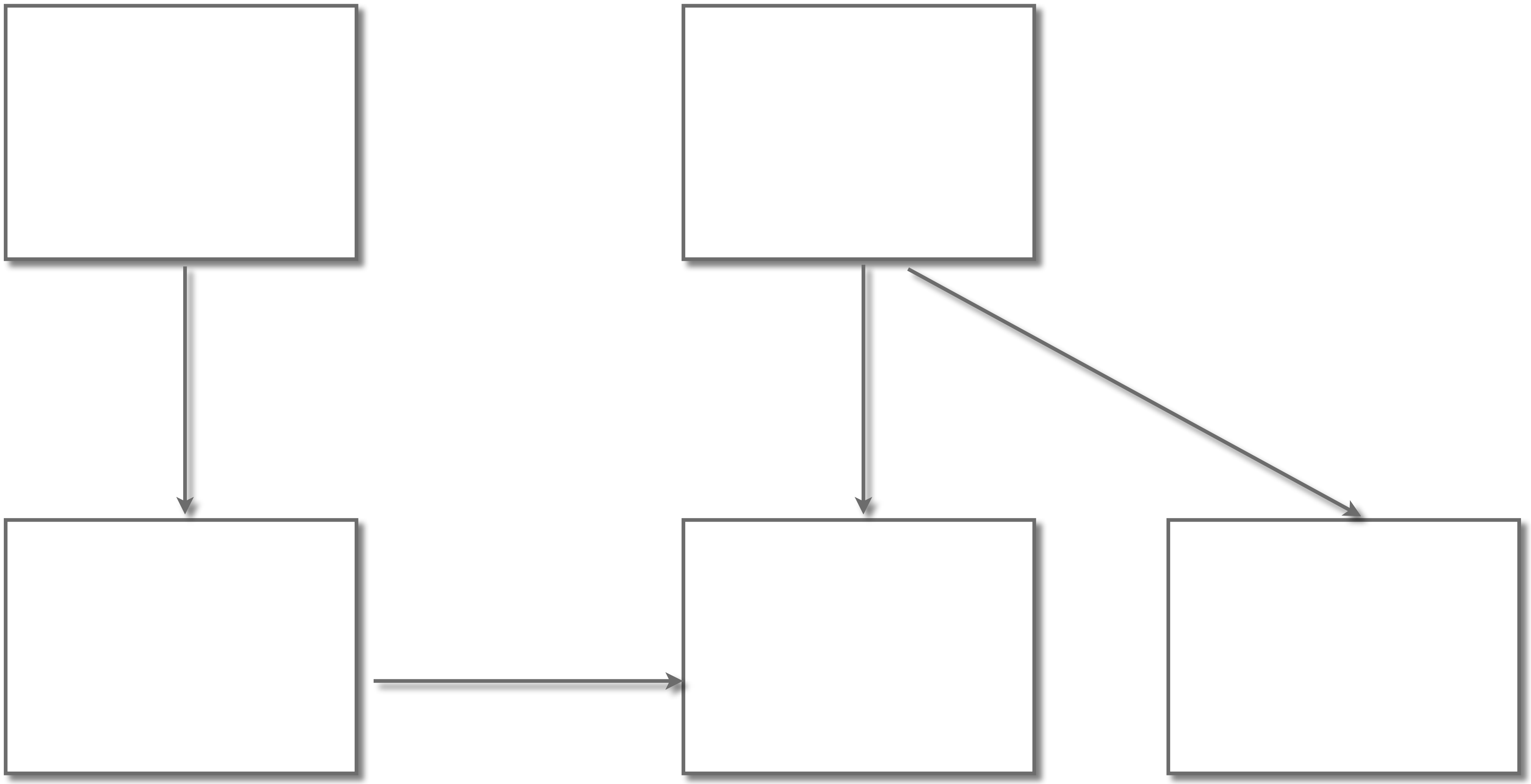
Philip Greenspun's Tenth Rule of Programming
http://philip.greenspun.com/research/

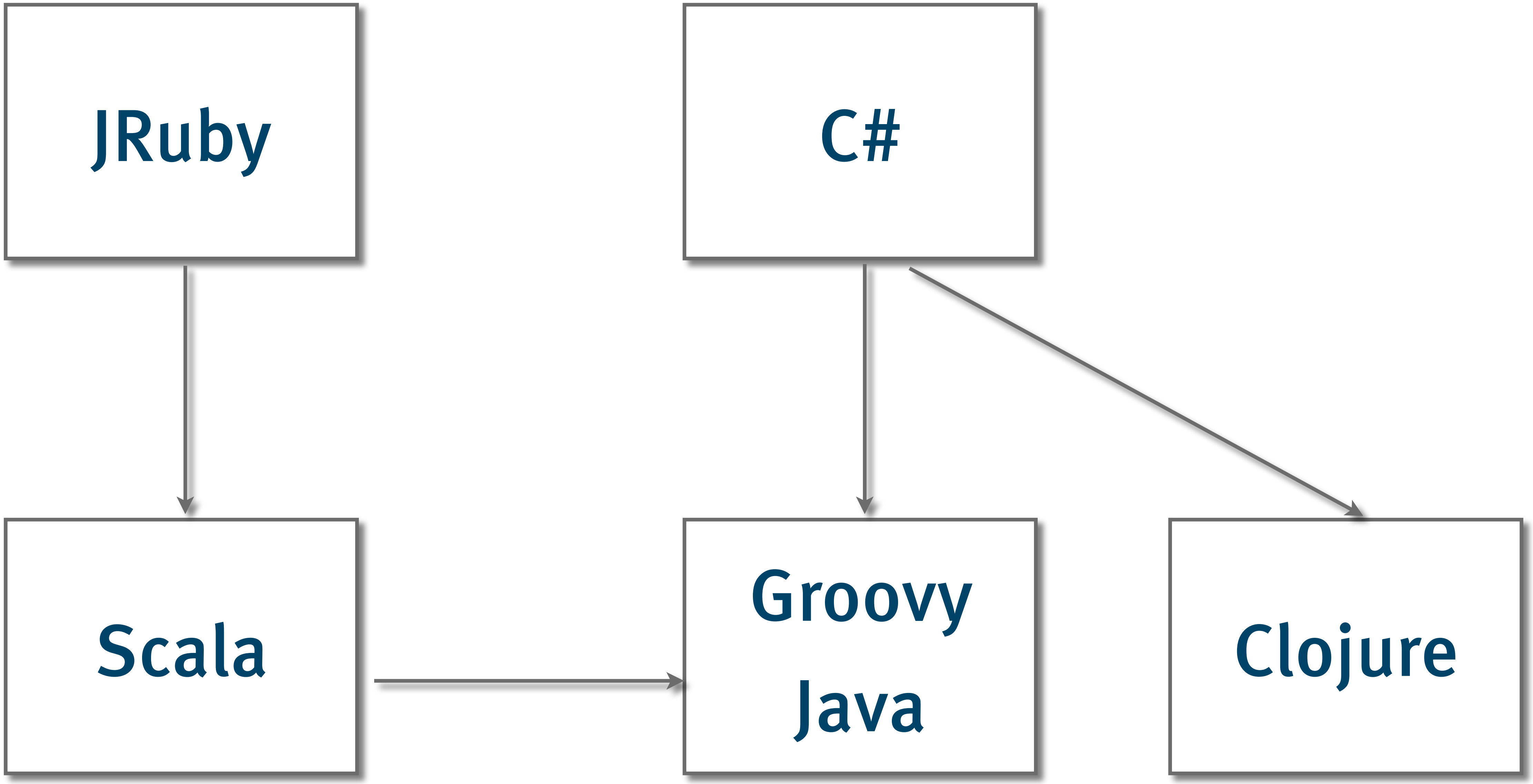
What do you know that the developers of {Ruby|Clojure|Groovy|...} don’t?

5. Stability layers

Soft and hard spots suggest different languages

6. Distributed Systems




JRuby
C#
Scala
Groovy
Java
Clojure

Modularization

Application Size	Modularization
1-50 LOC	1 file
50-500 LOC	few files, many functions
500-1000 LOC	library, class hierarchy
1000-2000 LOC	framework + application
>2000 LOC	more than one application

Necessary Rules & Guidelines

Cross-system

Responsibilities

UI integration

Communication protocols

Data formats

Redundant data

BI interfaces

Logging, Monitoring

System-internal

Programming languages

Development tools

Frameworks

Process/Workflow control

Persistence

Design patterns

Coding guidelines

(Deployment, Operations)

Project Example 4

Web-based secure email service

Java/Spring/JAX-RS RESTful HTTP services

JRuby/Rails frontend

Puppet/Ruby for automation

Numerous components in C

:-)

6. Distributed Systems

Modern distribution architecture creates freedom

7. People

Skills

Community

Prejudice

Dependencies

Frustration

Motivation

7. People

As usual, people matter most

1. **Languages differ drastically**
2. **No language is an island**
3. **MLVMs enable diversity**
4. **Nobody uses a single language**
5. **Soft and hard spots suggest different languages**
6. **Modern distribution architecture creates freedom**
7. **As usual, people matter most**

Q&A

Stefan Tilkov, @stilkov

stefan.tilkov@innoq.com

http://www.innoq.com

Phone: +49 170 471 2625

We will take care of it - personally.