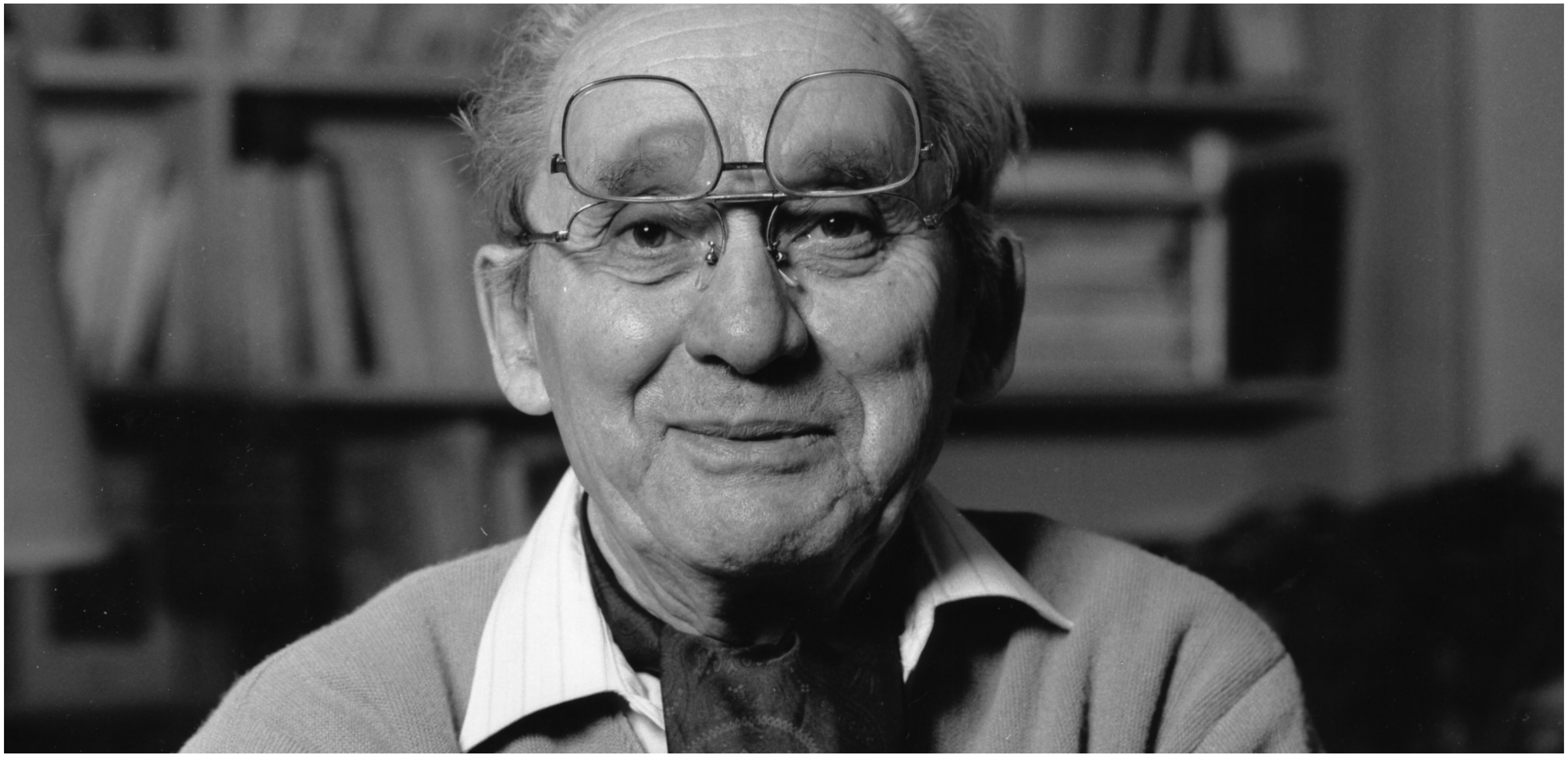
Digital Ricœur

Racket in the Humanities

or, “Come for the web server; stay for the **`#lang`**.”



Philip M^c^Grath

The University of Chicago & Digital Ricœur

philip@philipmcgrath.com

What is Digital Ricœur?

Digital Humanities

13e Arrt
PLACE
PAUL RICOEUR
1913 - 2005
PHILOSOPHE

What is Digital Ricœur?

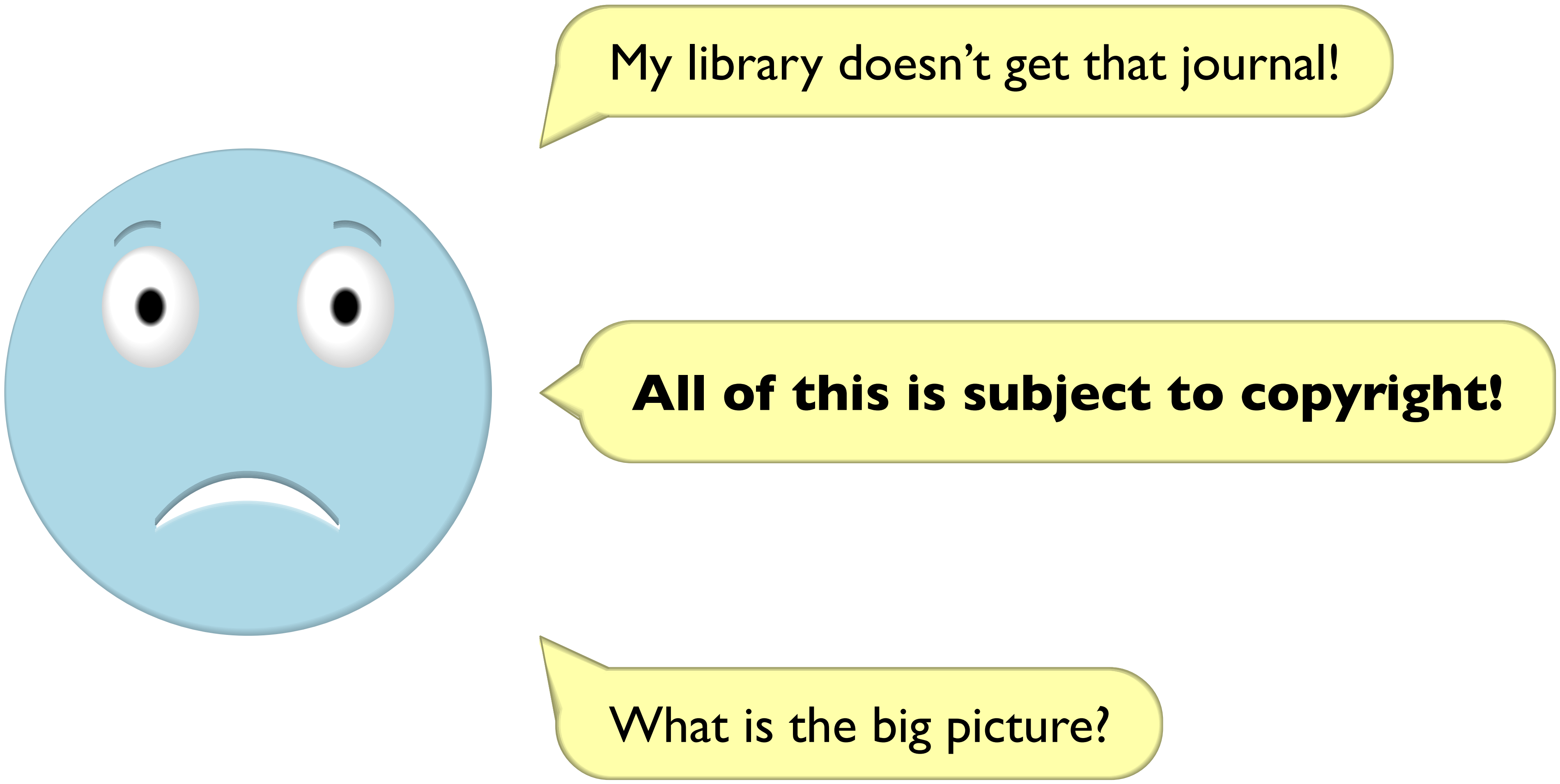
- Kyoto Prize, 2000
- Broad impact beyond philosophy:
    - History
    - Religious Studies
    - Law
    - Psychology
    - Medicine
    - Education

What is Digital Ricœur?

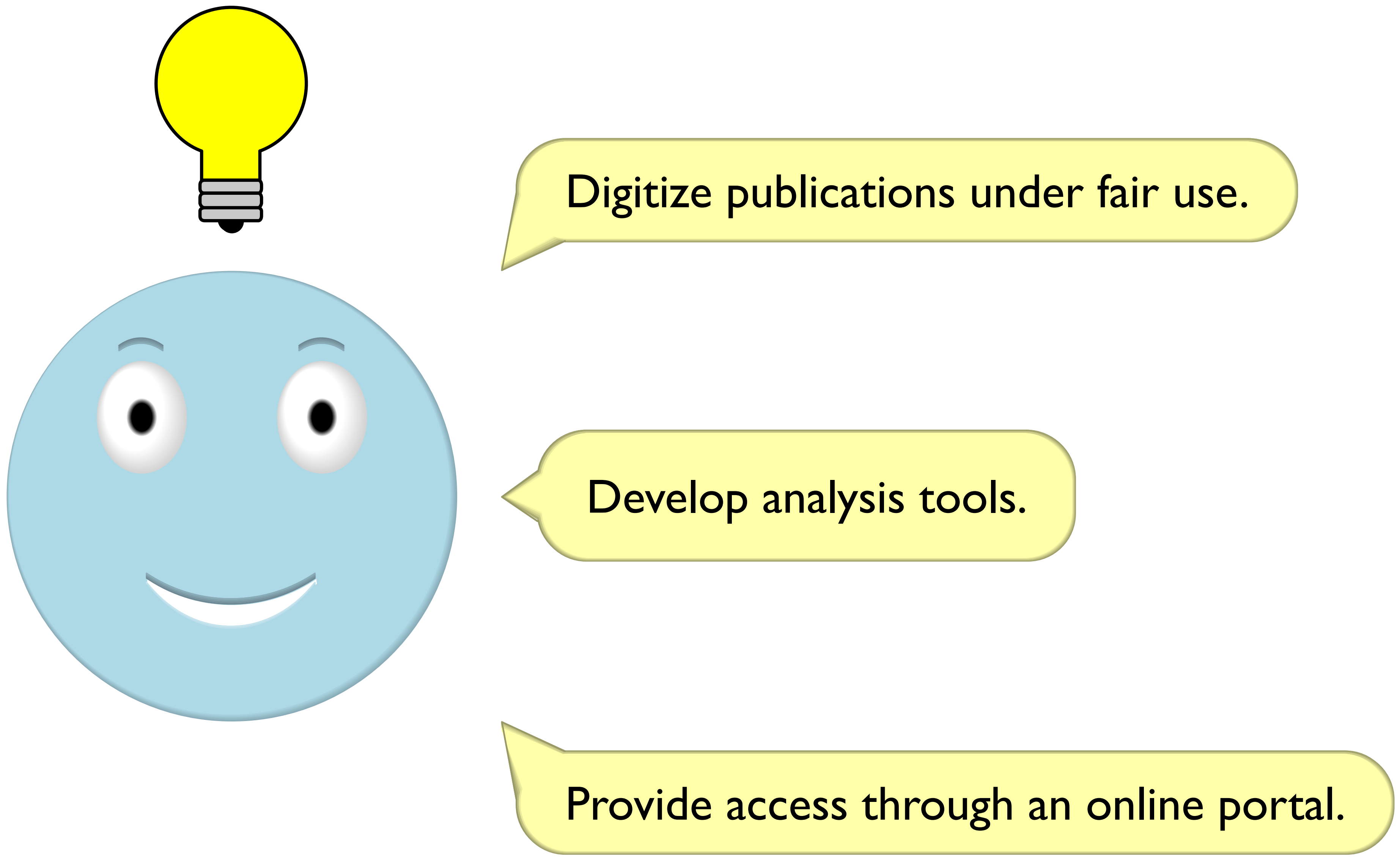
Publications (approximate):

- French:
  - 40 books
  - 800 articles
- English:
  - 40 books *(not all the same books!)*
  - 240 articles
- 30 other languages
- Secondary literature ...

What is Digital Ricœur?


My library doesn't get that journal!
All of this is subject to copyright!
What is the big picture?

What is Digital Ricœur?


Digitize publications under fair use.
Develop analysis tools.
Provide access through an online portal.

What is Digital Ricœur?

Other researchers can ask new questions —even without technical skills!

What is Digital Ricœur?

- Launched portal in October 2017
  - < 6 months of development
- Digitized all English primary sources
- French, German, and Spanish in progress
- Enthusiastic reception
  - > 200 users
  - > 20 countries
  - North & South America; Europe; Africa; Asia
  - Growing steadily

Who is Digital Ricœur?

- Small team of academics:
  - Musicology & Web Development
  - Philosophy & Software Engineering
  - Religious Studies & Electrical Engineering
  - Law
- Some with no technical background
- Distributed across 5 countries (4 continents)
- Not full-time

What is Digital Ricœur?

How did we do it?

Racket!

1. "Racket Internalizes Extra-Linguistic Mechanisms"

2. "Racket is a Programming-Language Programming Language"

The Racket Manifesto

“Racket Internalizes Extra-Linguistic Mechanisms”

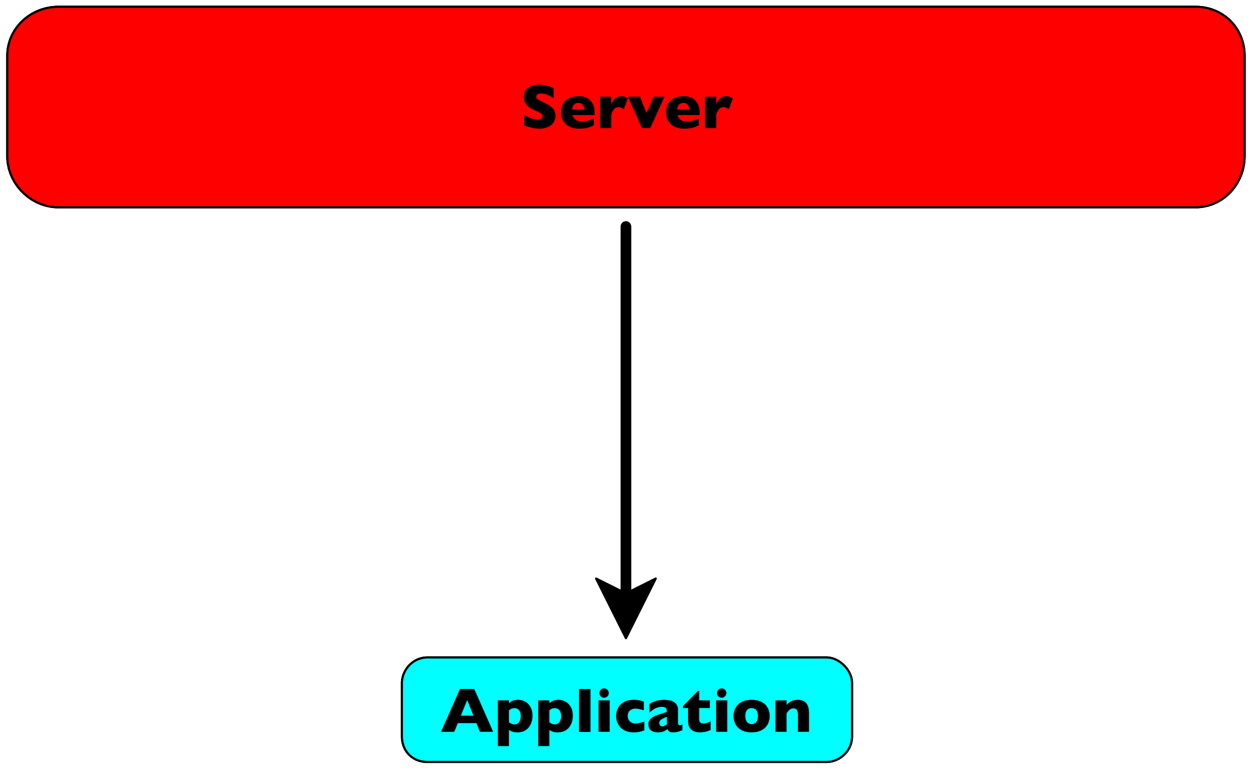
Initial Goals

- Portal website should integrate independent tools
- Uniform access control
- Use Voyant Tools[†] for visualizations
    - Runs its own webserver
    - JavaScript UI
    - Embedding supported via **`<iframe />`**

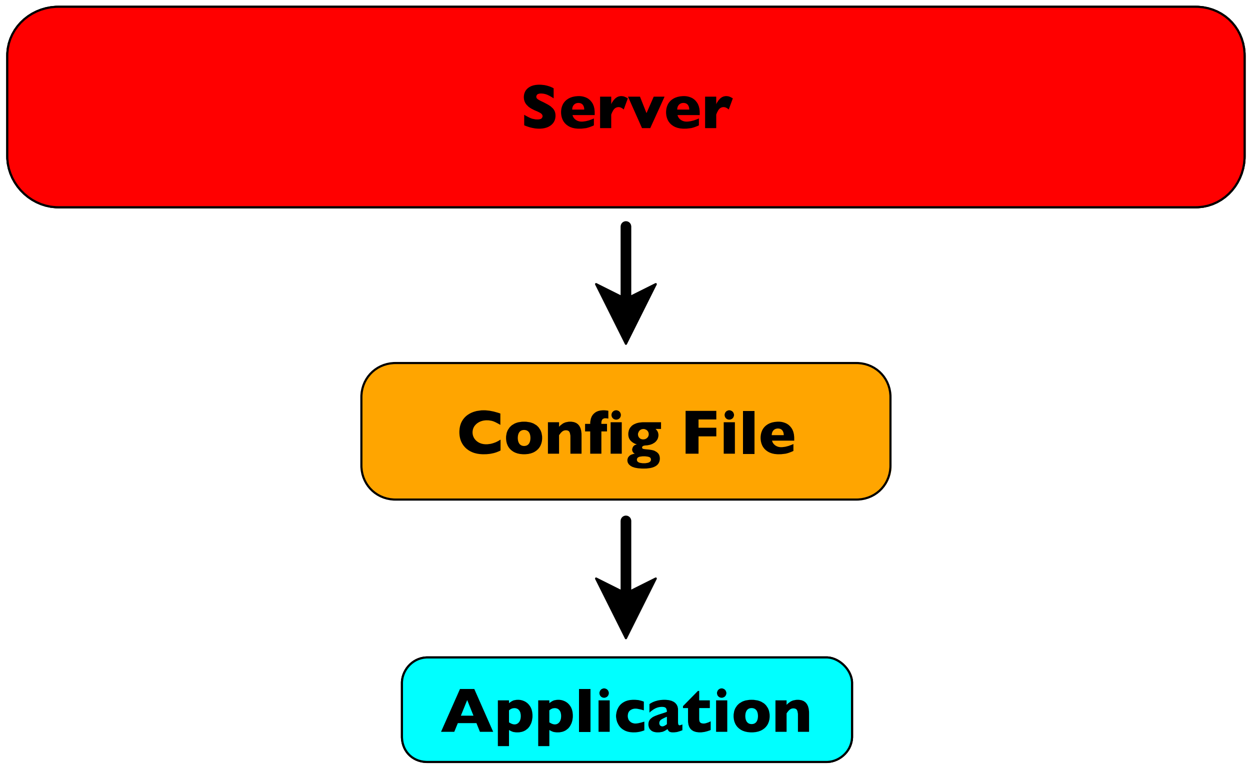
∴ Run Voyant Tools behind a proxy

[†] Stéfan Sinclair, Geoffrey Rockwell, et al. 2012: voyant-tools.org

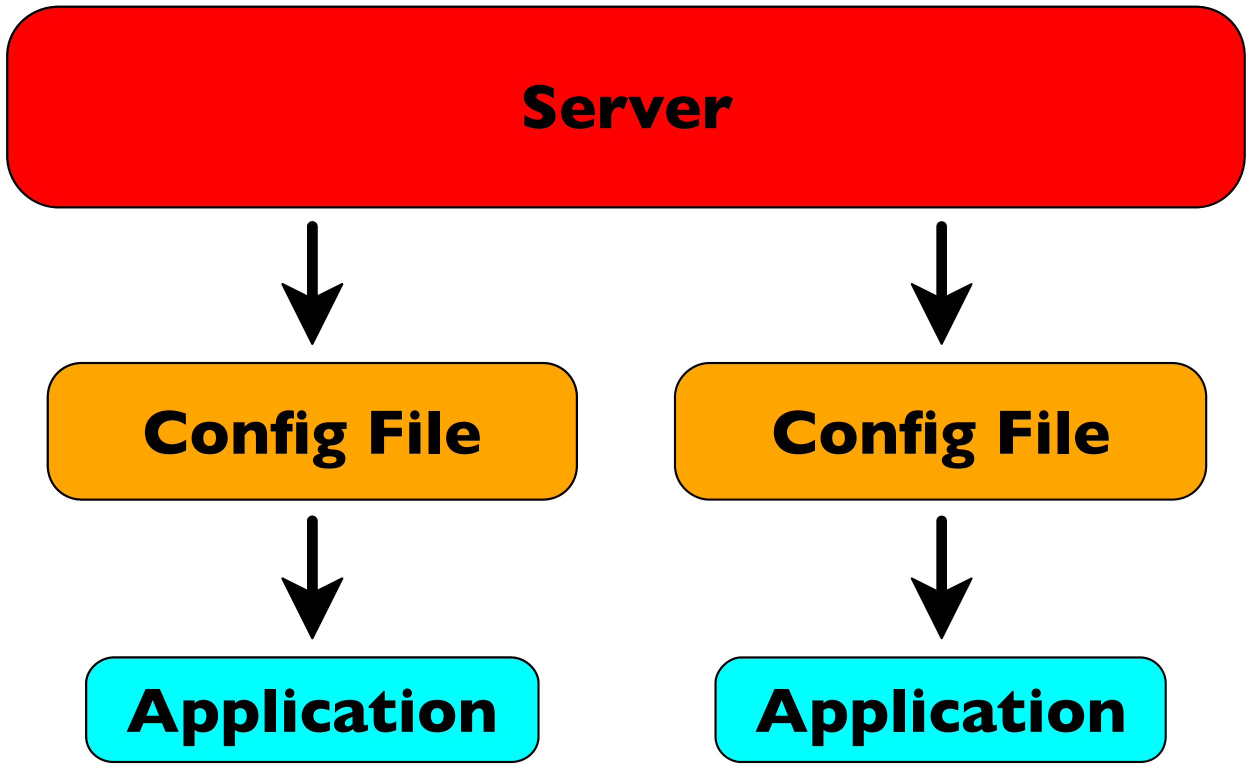
Other Languages & CGI



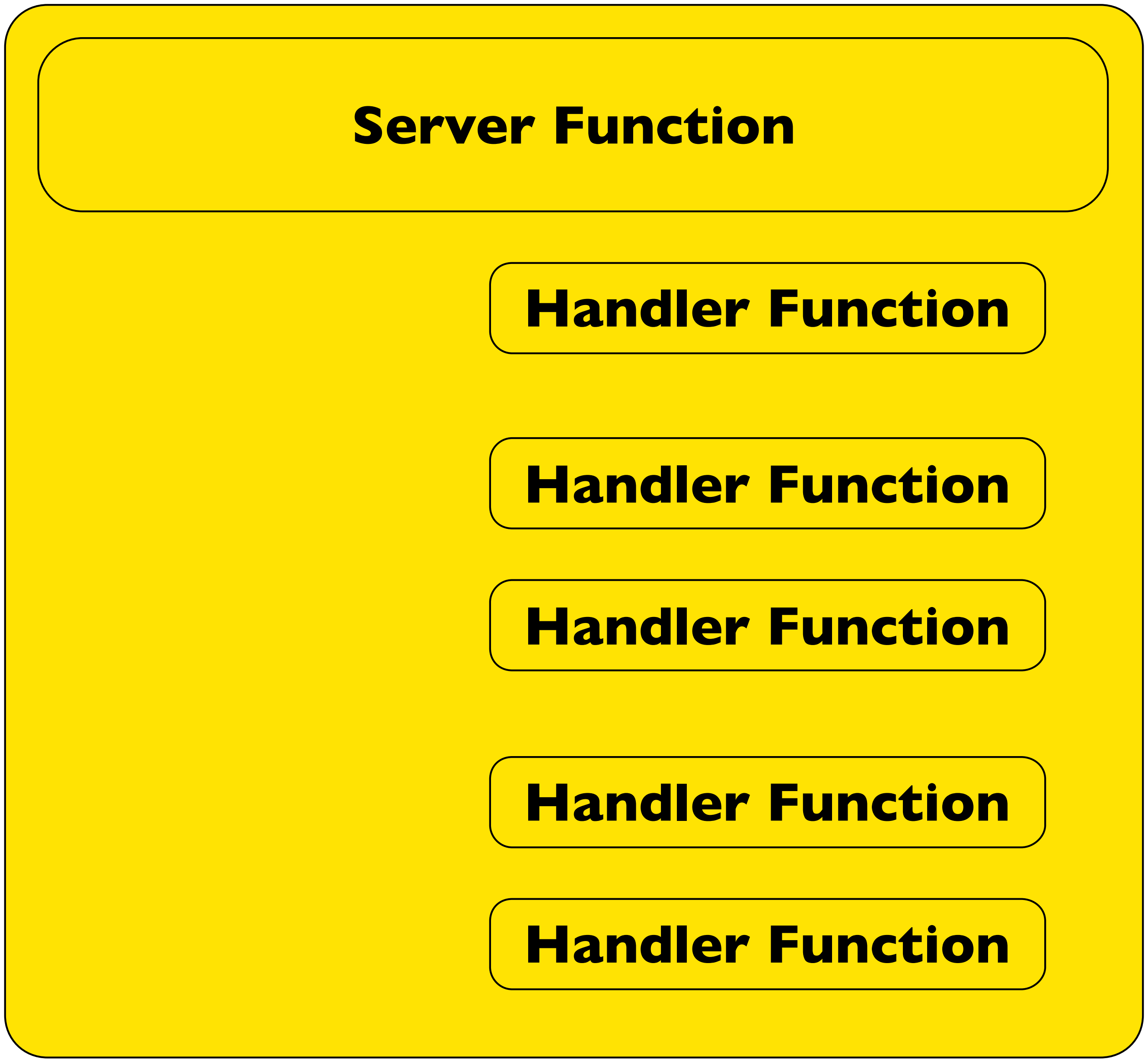
Other Languages & CGI



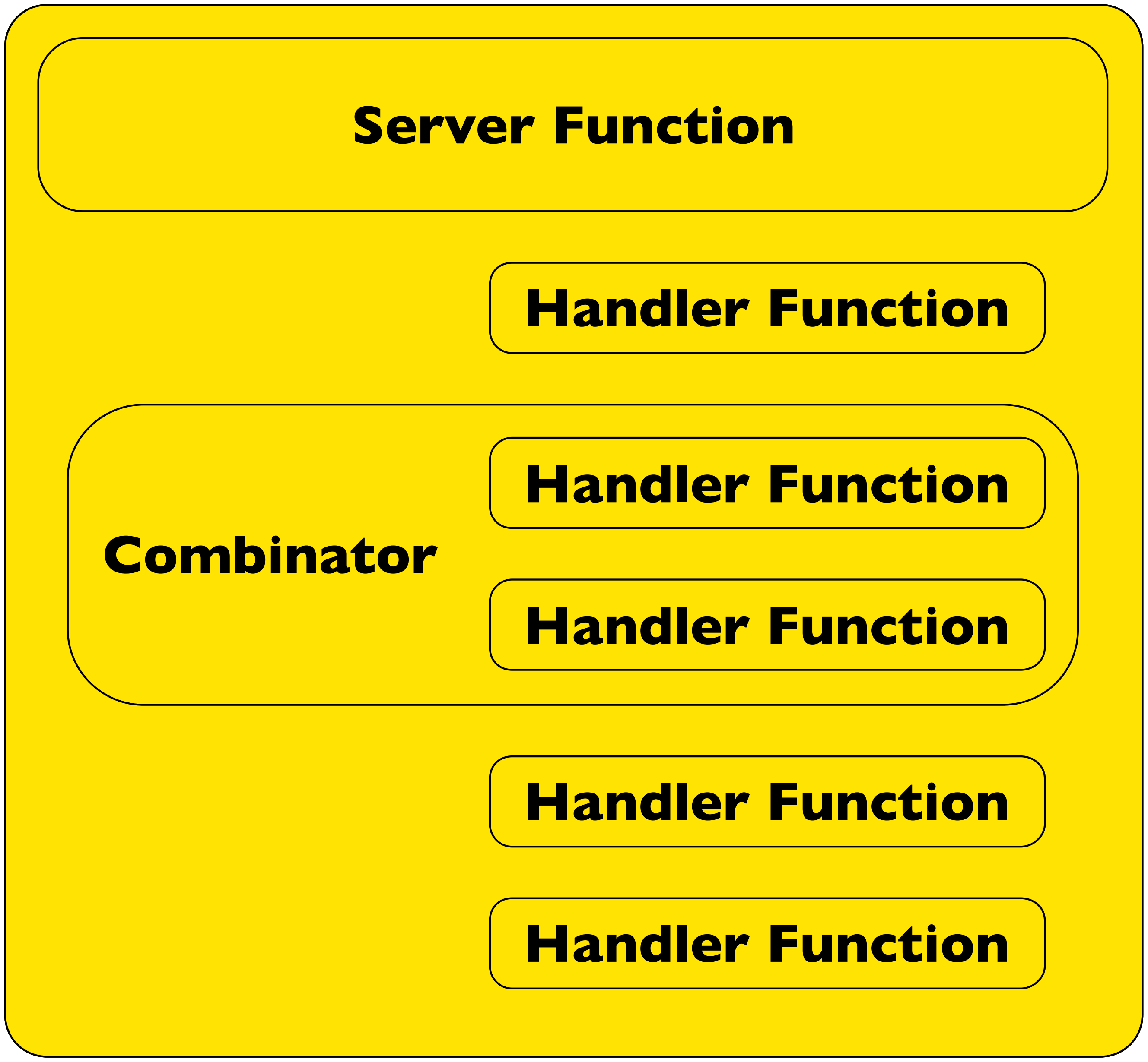
Other Languages & CGI



Racket Web Server



Racket Web Server



Practical Impact

- Prototype in an afternoon
- Launch in < 6 months
- Deployment is delightful (package system; **`raco setup`**)
- Runs the same way on laptop or server

“Racket is a Programming-Language Programming Language”

TEI

Text Encoding Initiative

XML

X-Expressions

```
<div type="chapter" n="1">
  <p>Hi, world!</p>
</div>
```

```
'(div ([type "chapter"]
       [n "1"])
      (p "Hi, world!"))
```

Add an Interface

```
(define element%
  (class object%
    (super-new)
    (init-field xexpr)
    (define/public (get-name)
      (car xexpr))
    (define/public (get-attributes)
      ... xexpr ...)
    (define/public (get-body)
      ... xexpr ...)))
```

Add an Interface

```
(define div%
  (class element%
    (super-new)
    (inherit get-attributes)
    (define/public (get-number)
      (string->number
       (car (dict-ref (get-attributes)
                      'n))))))
```

Get More Checking …

```
<?xml version="1.0" encoding="utf-8"?>
<TEI xmlns="http://www.tei-c.org/ns/1.0"
     version="5.0">
  <teiHeader>
    <fileDesc>
      <titleStmt><title/></titleStmt>
      <publicationStmt>
        <authority/>
      </publicationStmt>
      <sourceDesc><bibl/></sourceDesc>
    </fileDesc>
  </teiHeader>
  <text>
    <body>
      <ab/>
    </body>
  </text>
</TEI>
```

With Contracts!

```
(define div/c
  (make-element-contract
   'div
   #:children `([0+ p]
                [0+ pb]
                [0+ div])
   #:attr-contracts
   `([type ,(or/c "chapter"
                  "section")])
   #:required-attrs '(type)))
```

Literate Programming

```
#lang scribble/lp2

The @tag{div} element must have
the attributes @attr{type} and @attr{n}.

@chunk[<*>
       (define div/c
         (make-element-contract
          'div
          #:children `([0+ p]
                       [0+ pb]
                       [0+ div])
          #:attr-contracts
          `([type ,(or/c "chapter"
                         "section")])
          #:required-attrs '(type)))]
```

There is a Bug …

```
#lang scribble/lp2

The @tag{div} element must have
the attributes @attr{type} and @attr{n}.

@chunk[<*>
       (define div/c
         (make-element-contract
          'div
          #:children `([0+ p]
                       [0+ pb]
                       [0+ div])
          #:attr-contracts
          `([type ,(or/c "chapter"
                         "section")])
          #:required-attrs '(type)))]
```

There is a Bug …

```
(define div/c
  (make-element-contract
   'div
   #:children `([0+ p]
                [0+ pb]
                [0+ div])
   #:attr-contracts
   `([type ,(or/c "chapter"
                  "section")])
   #:required-attrs '(type)))
```

There is a Bug …

```
(define div%
  (class element%
    (super-new)
    (inherit get-attributes)
    (define/public (get-number)
      (string->number
       (car (dict-ref (get-attributes)
                      'n))))))

(define div/c
  (make-element-contract
   'div
   #:children `([0+ p]
                [0+ pb]
                [0+ div])
   #:attr-contracts
   `([type ,(or/c "chapter"
                  "section")])
   #:required-attrs '(type)))
```

Out of Sync!

```
(define div%
  (class element%
    (super-new)
    (inherit get-attributes)
    (define/public (get-number)
      (string->number
       (car (dict-ref (get-attributes)
                      'n))))))

(define div/c
  (make-element-contract
   'div
   #:children `([0+ p]
                [0+ pb]
                [0+ div])
   #:attr-contracts
   `([type ,(or/c "chapter"
                  "section")])
   #:required-attrs '(type)))
```

Out of Sync!

```
(define tei/c                  (define-signature contracts^
  (match-lambda
    ['TEI TEI/c]                 (TEI/c
    ['div div/c]                  div/c
    ['p p/c]                      p/c
    ['note note/c]
    ['pb pb/c]))                  pb/c))
```

Domain-Specific Language

```
#lang ricoeur/tei/kernel

ƒtitle{Formal Specification}

ƒbegin-for-runtime[
 (require "support.rkt")
 (provide tei-document?
          tei-document-checksum)]

ƒ(define-element TEI
   #:children ([1 teiHeader]
               [1 text])
   #:required-order (teiHeader text)
   #:attr-contracts
   ([version "5.0"]
    [xmlns "http://www.tei-c.org/ns/1.0"])
   #:required-attrs (version xmlns)
   #:predicate tei-document?
   #:constructor
   [#:body/elements-only body/elements-only
    #:this/thunk get-this
    (field text #:hide)
    (match-define (list teiHeader text)
      body/elements-only)
    (define/field pr:md5
      (delay/thread
       (xexpr->md5
        (tei-element->xexpr (get-this)))))]
   #:begin [(define (tei-document-checksum doc)
              (force (get-field pr:md5 doc)))]
   #:property prop:element->plain-text
   (λ (this)
     (element-or-xexpr->plain-text
      (get-field text this)))
   #:prose ƒ{
 The root element contains ƒtag{teiHeader}
 and ƒtag{text} elements.})
```

Domain-Specific Language

```
#lang ricoeur/tei/kernel

ƒtitle{Formal Specification}

ƒbegin-for-runtime[
 (require "support.rkt")
 (provide tei-document?
          tei-document-checksum)]

ƒ(define-element TEI
   #:children ([1 teiHeader]
               [1 text])
   #:required-order (teiHeader text)
   #:attr-contracts
   ([version "5.0"]
    [xmlns "http://www.tei-c.org/ns/1.0"])
   #:required-attrs (version xmlns)
```

Domain-Specific Language

```
ƒ(define-element TEI
   #:children ([1 teiHeader]
               [1 text])
   #:required-order (teiHeader text)
   #:attr-contracts
   ([version "5.0"]
    [xmlns "http://www.tei-c.org/ns/1.0"])
   #:required-attrs (version xmlns)
   #:predicate tei-document?
   #:constructor
   [#:body/elements-only body/elements-only
    #:this/thunk get-this
    (field text #:hide)
    (match-define (list teiHeader text)
      body/elements-only)
    (define/field pr:md5
      (delay/thread
```

Domain-Specific Language

```
    #:this/thunk get-this
    (field text #:hide)
    (match-define (list teiHeader text)
      body/elements-only)
    (define/field pr:md5
      (delay/thread
       (xexpr->md5
        (tei-element->xexpr (get-this)))))]
   #:begin [(define (tei-document-checksum doc)
                (force (get-field pr:md5 doc)))]
   #:property prop:element->plain-text
   (λ (this)
     (element-or-xexpr->plain-text
      (get-field text this)))
   #:prose ƒ{
 The root element contains ƒtag{teiHeader}
 and ƒtag{text} elements.})
```

`TEI` TEI element

Attributes:

  `version : "5.0"`

  `xmlns : "http://www.tei-c.org/ns/1.0"`

Required attributes:

  `version` and `xmlns`

Children:

  `1 teiHeader`

  `1 text`

Required order:

  `teiHeader`, `text`

The document should begin with a prelude, which must be exactly as follows:

```
<?xml version="1.0" encoding="utf-8"?>
```

The root element is a `TEI` element, which contains exactly (in order) `teiHeader` and `text` elements. It must have the attributes `version="5.0"` and `xmlns="http://www.tei-c.org/ns/1.0"`.

Thank You!

- Empower non-technical users
  - Obvious technologies can still be transformative
- Use linguistic constructs, not external state
  - Prototype quickly & deploy delightfully
- Grow languages to express problems naturally

Digital Ricœur

- Website: **`digitalricoeur.org`**
- Code: **`bitbucket.org/digitalricoeur/tei-utils`**
- *More to follow!*