The Roots of Go

GOPHERCON INDIA 2015

図式建築

Or, how

our past can lead us to the future

वैशम्पायन घोष (BG)

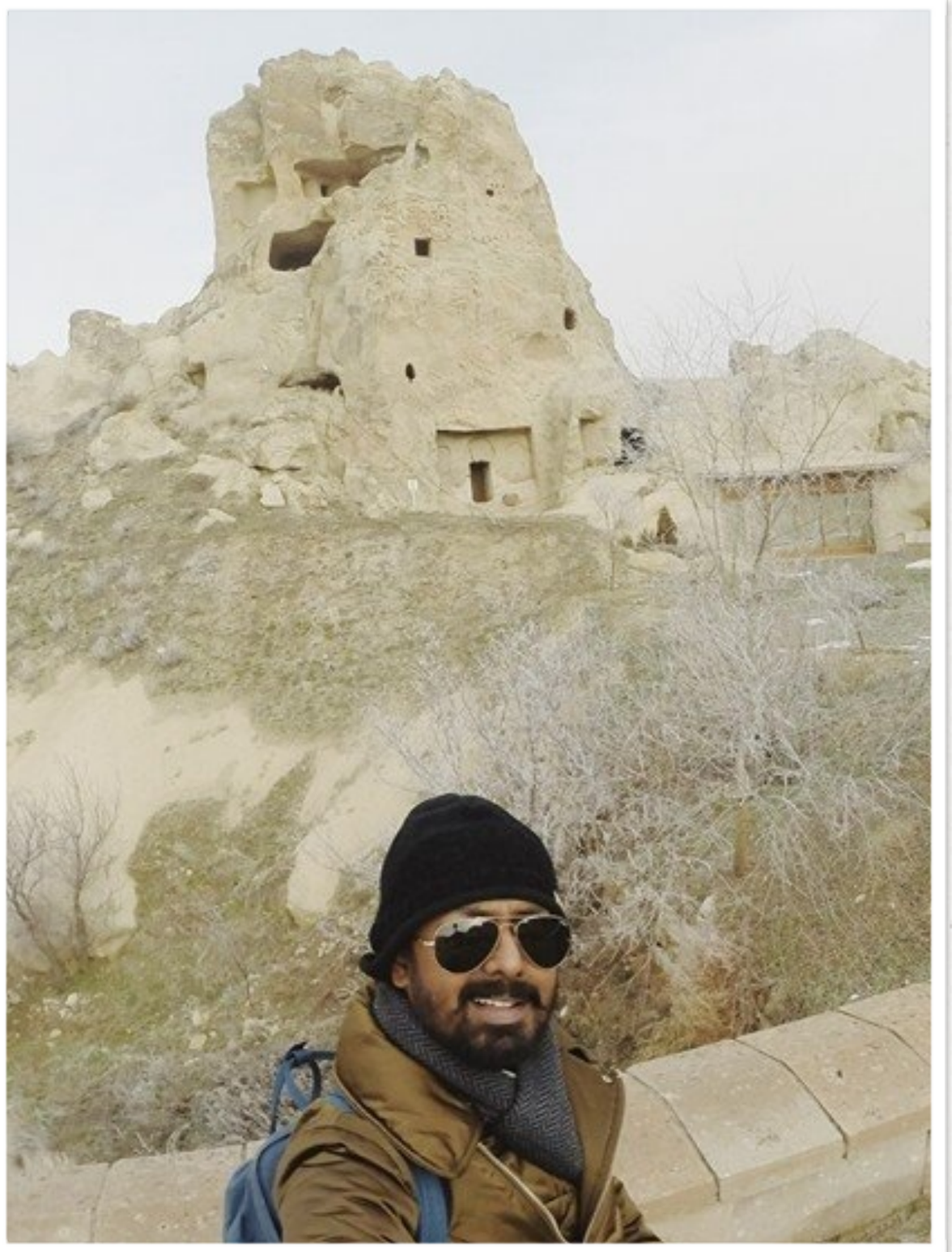
CTO/Co-founder

Helpshift, Inc.

@ghoseb

Language Archæologist

#helpshift

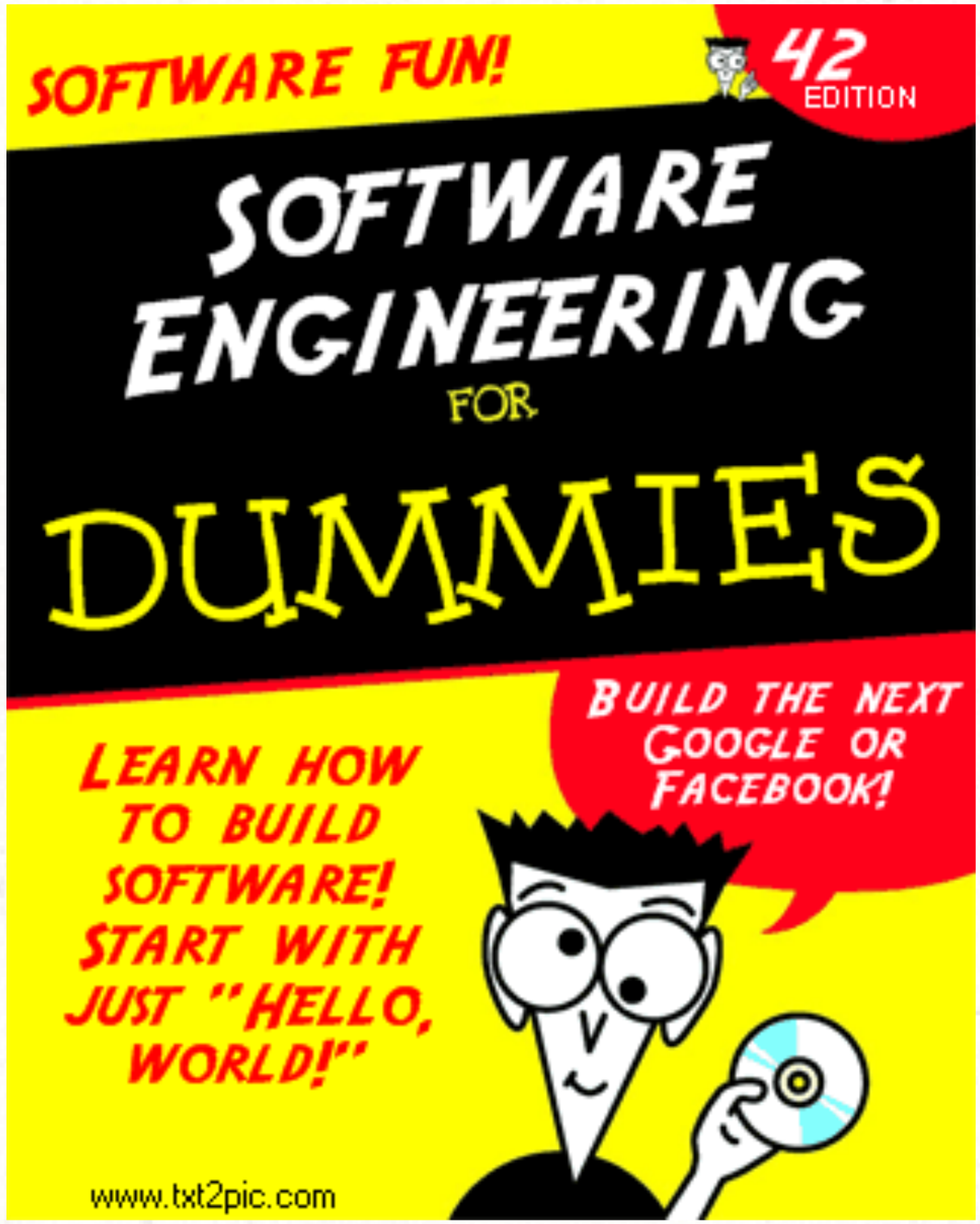


図式建築

Motivation

Why should I care?

And now, for something completely different…



Design Pattern!

建築図式

{h=3, w=3, b=3}

x 100

Let’s scale this, baby!

{h=3, w=3, b=3}

x 100

Let’s scale this, baby!

図式建築

{h=3, w=3, b=3}

x 100

Let’s scale this, baby!

図式建築

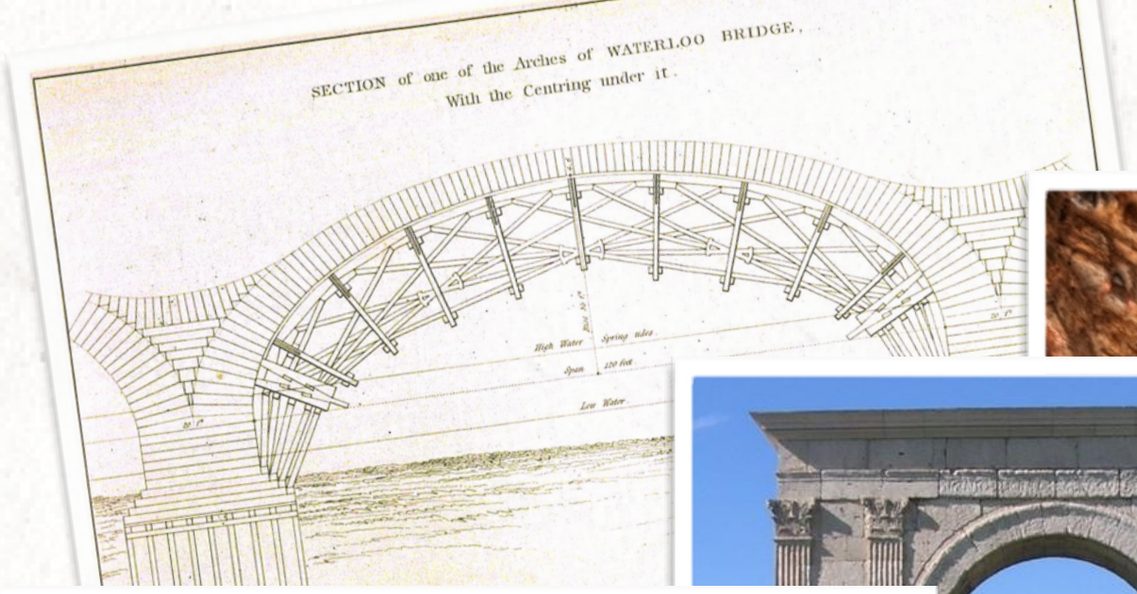
{h=3, w=3, b=3}

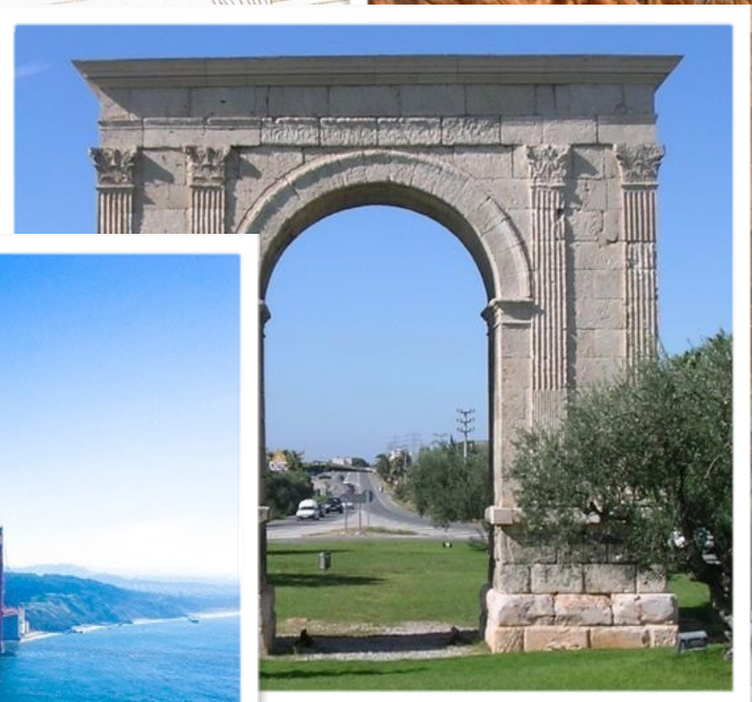
x 100

It’s a feature!

Let’s scale this, baby!

図式建築

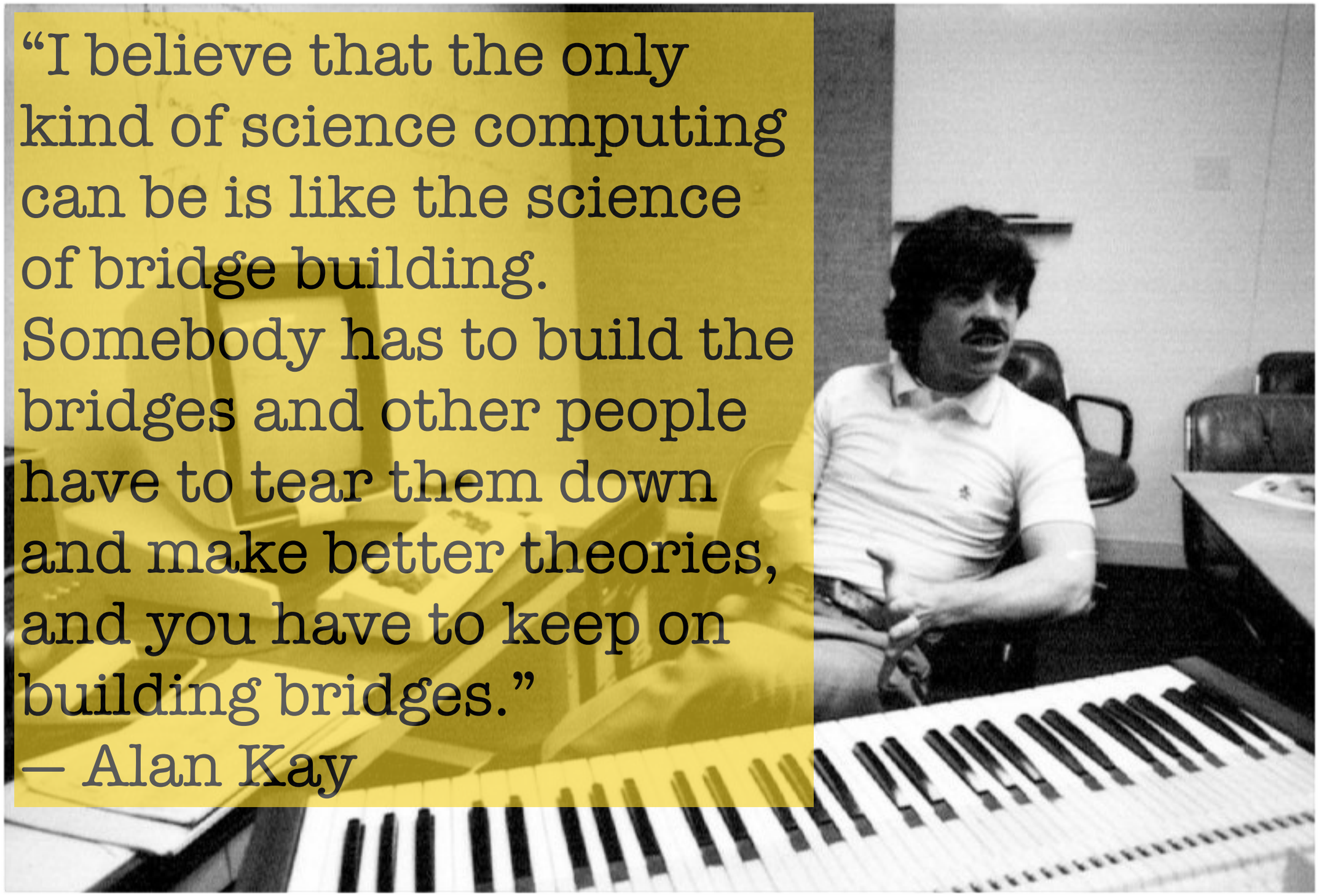




Architecture

n. the art or practice of designing structures

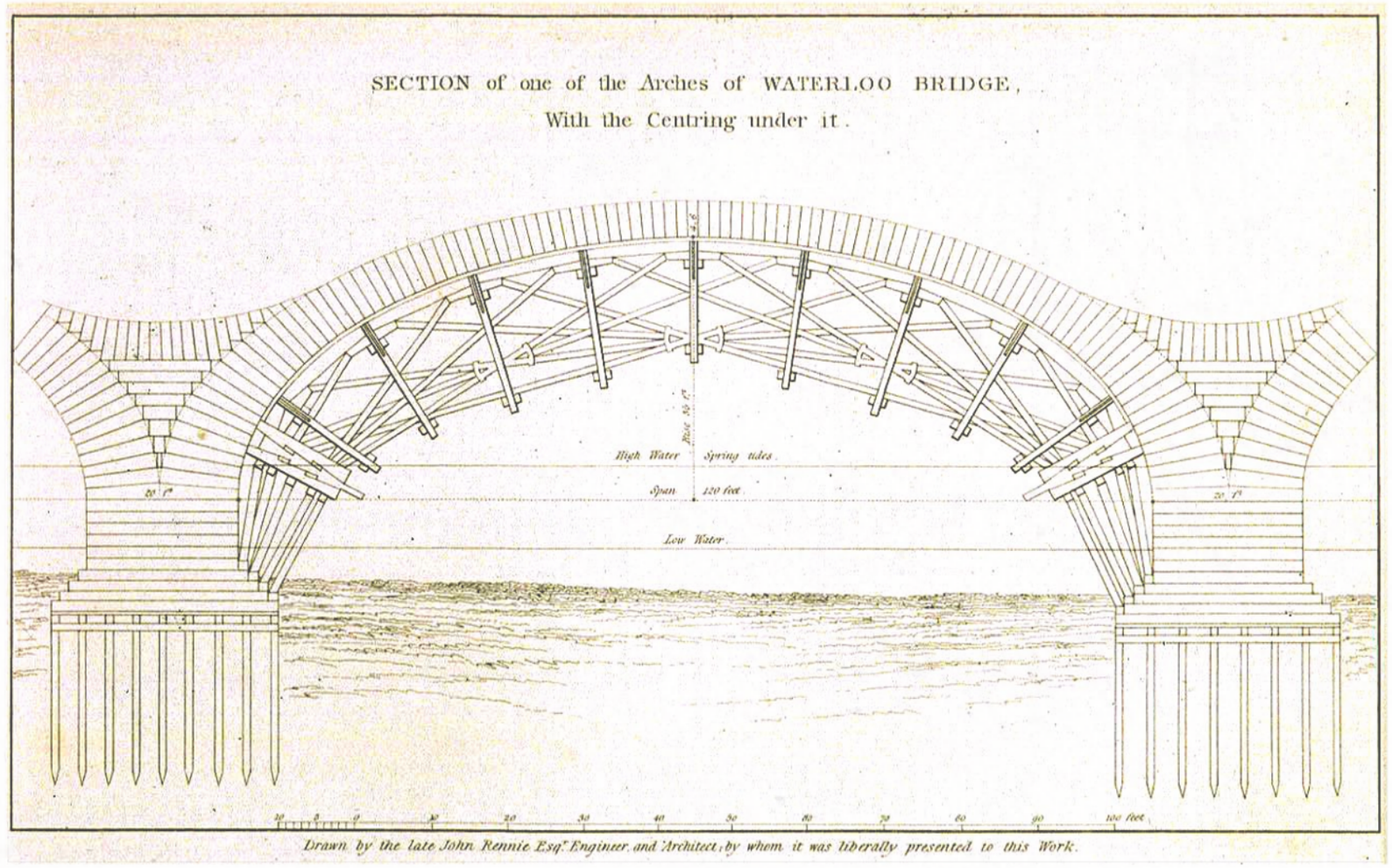
式築
図建


“I believe that the only kind of science computing can be is like the science of bridge building. Somebody has to build the bridges and other people have to tear them down and make better theories, and you have to keep on building bridges.”
— Alan Kay

The Roots of Go

図式建築

The Arches of Go



MASSACHUSETTS INSTITUTE OF TECHNOLOGY

PROJECT MAC

Memorandum-M-352
July 21, 1967

To: Project MAC Participants

From: Martin Richards

Subject: The BCPL Reference Manual

ABSTRACT

BCPL is a simple recursive programming language designed for compiler writing and system programming: it was derived from true CPL (Combined Programming Language) by removing those features of the full language which make compilation difficult namely, the type and mode matching rules and the variety of definition structures with their associated scope rules.

BCPL

Martin Richards [1967]

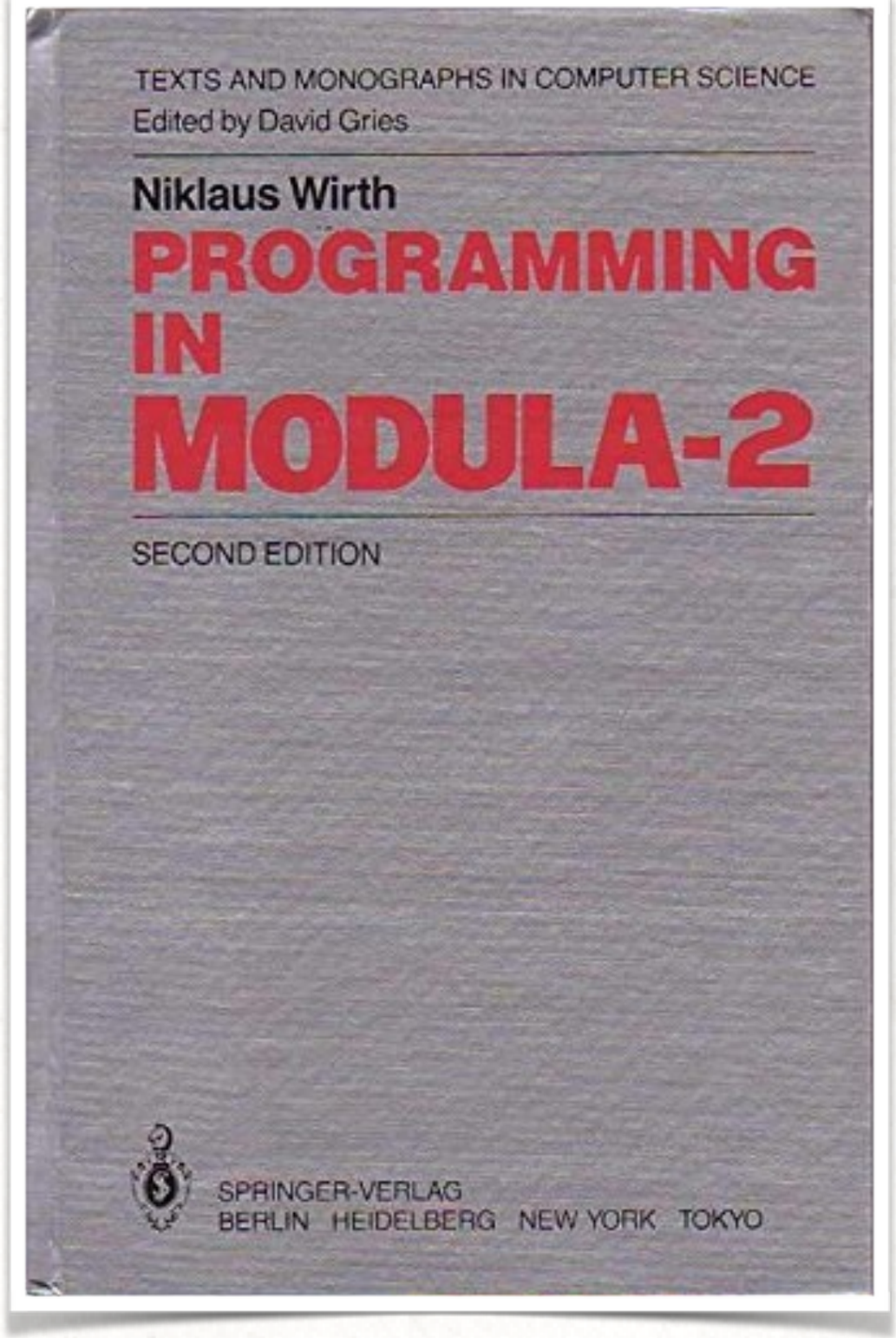
```
GET "libhdr"

LET start() = VALOF
$( writes("Hello, World!*n")
   RESULTIS 0
$)
```

BCPL

Martin Richards [1967]

Modula-2

Niklaus Wirth [1977]



図式建築

Modula-2

Niklaus Wirth [1977]

```
MODULE ggTkgV;

FROM    InOut           IMPORT  ReadCard, WriteC
WriteBf;

VAR   x, y, u, v        : CARDINAL;

BEGIN
  WriteString ("x = ");          WriteBf;
  WriteString ("y = ");          WriteBf;
  u := x;
  v := y;
  WHILE  x # y  DO
    (*  ggT (x, y) = ggT (x0, y0), x * v + y * u
    IF  x > y  THEN
      x := x - y;
      u := u + v
    ELSE
      y := y - x;
      v := v + u
    END
  END;
  WriteString ("ggT =");          WriteCard (x, 6)
  WriteString ("kgV =");          WriteCard ((u+v)
  WriteString ("u =");            WriteCard (u, 6)
  WriteString ("v =");            WriteCard (v , 6
END ggTkgV.
```

Programming Techniques
S. L. Graham, R. L. Rivest Editors

Communicating Sequential Processes

C.A.R. Hoare
The Queen's University
Belfast, Northern Ireland

This paper suggests that input and output are basic primitives of programming and that parallel composition of communicating sequential processes is a fundamental program structuring method. When combined with a development of Dijkstra's guarded command, these concepts are surprisingly versatile. Their use is illustrated by sample solutions of a variety of familiar programming exercises.

Key Words and Phrases: programming, programming languages, programming primitives, program structures, parallel programming, concurrency, input, output, guarded commands, nondeterminacy, coroutines, procedures, multiple entries, multiple exits, classes, data representations, recursion, conditional critical regions, monitors, iterative arrays
CR Categories: 4.20, 4.22, 4.32

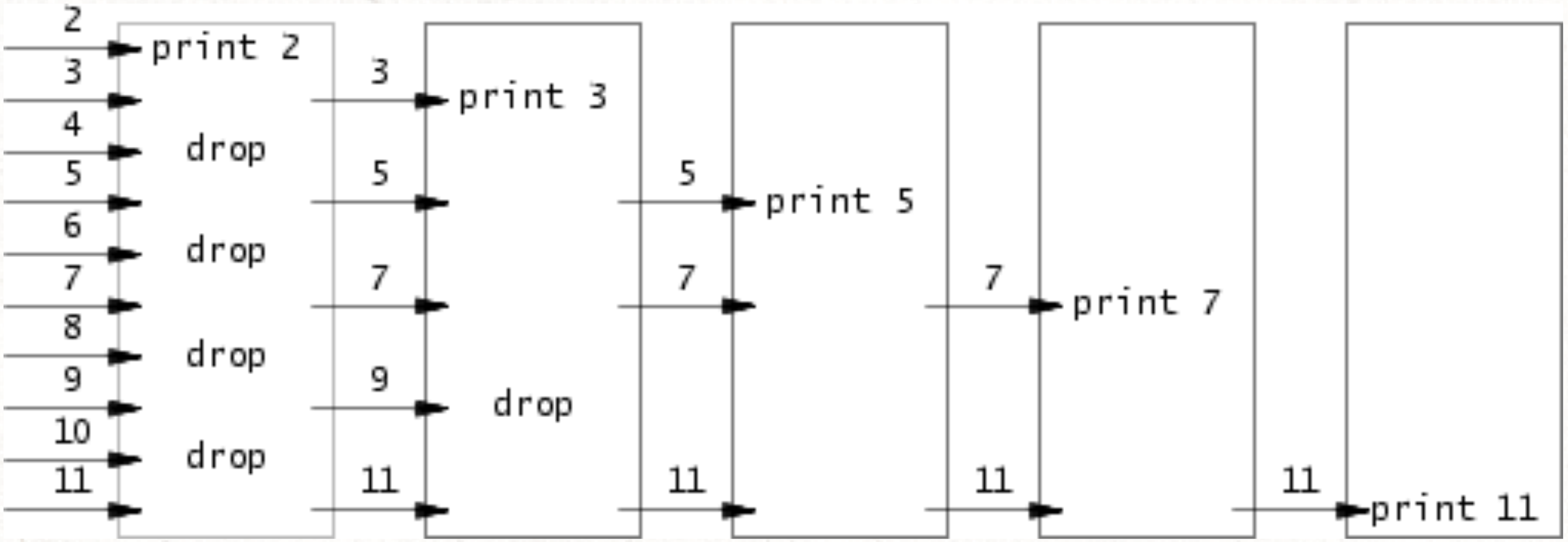
grams, three basic constructs have received widespread recognition and use: A repetitive construct (e.g. the **while** loop), an alternative construct (e.g. the conditional **if..then..else**), and normal sequential program composition (often denoted by a semicolon). Less agreement has been reached about the design of other important program structures, and many suggestions have been made: Subroutines (Fortran), procedures (Algol 60 [15]), entries (PL/I), coroutines (UNIX [17]), classes (SIMULA 67 [5]), processes and monitors (Concurrent Pascal [2]), clusters (CLU [13]), forms (ALPHARD [19]), actors (Hewitt [1]).

The traditional stored program digital computer has been designed primarily for deterministic execution of a single sequential program. Where the desire for greater speed has led to the introduction of parallelism, every attempt has been made to disguise this fact from the programmer, either by hardware itself (as in the multiple function units of the CDC 6600) or by the software (as in an I/O control package, or a multiprogrammed operating system). However, developments of processor technology suggest that a multiprocessor machine, constructed from a number of similar self-contained processors (each with its own store), may become more powerful, capacious, reliable, and economical than a machine which is disguised as a monoprocessor.

In order to use such a machine effectively on a single task, the component processors must be able to communicate and to synchronize with each other. Many methods of achieving this have been proposed. A widely adopted method of communication is by inspection and updating of a common store (as in Algol 68 [18], PL/I, and many machine codes). However, this can create severe problems in the construction of correct programs

CSP

Tony Hoare [1978]



Aside

The prime sieve

図式建築

```
[SIEVE(i:1..100)::
    p,mp:integer;
    SIEVE(i - 1)?p;
    print!p;
    mp := p; comment mp is a multiple of p;
*[m:integer; SIEVE(i - 1)?m →
    *[m > mp → mp := mp + p];
    [m = mp → skip
    ||m < mp → SIEVE(i + 1)!m
    ]   ]
||SIEVE(0)::print!2; n:integer; n := 3;
    *[n < 10000 → SIEVE(1)!n; n := n + 2]
||SIEVE(101)::*[n:integer;SIEVE(100)?n → print!n]
||print::*[(i:0..101) n:integer; SIEVE(i)?n → ...]
]
```

CSP

Tony Hoare [1978]

Newsqueak: A Language for Communicating with Mice

Rob Pike

This is an informal reference manual for the concurrent language Newsqueak. Newsqueak's roots are in Squeak, a language designed a few years ago by Luca Cardelli and Rob Pike to illustrate concurrent solutions to problems in user interface design. Newsqueak addresses the same problems but in a broader context: Squeak was for designing devices such as menus and scroll bars; Newsqueak is for writing entire applications, and in particular a window system. Besides a wholesale redesign of Squeak's syntax, Newsqueak therefore has several major components absent in Squeak: a type system, dynamic process creation, and dynamic channel creation. Also, Squeak deferred most mundane programming details to the language it compiled into, C. Newsqueak is instead a self-contained language. An interpreter for it, called squint, has been implemented.

Newsqueak draws heavily from CSP and C, and the discussion that follows assumes modest familiarity with both these languages. Roughly, the syntax and basic semantics come from C, while the message-passing primitives come from CSP. The way these are put together is, however, unique to Newsqueak.

1. Text

Input is free-form. An identifier consists of alphanumeric characters, including underscore, and does not begin with a number. A sharp character # begins a comment, which continues until a new-line character. Files may be included; an occurrence of

```
include "file name"
```

is replaced by the contents of the file. (In this manual, italic text distinguishes syntactic constructions from literal text.)

Newsqueak

Rob Pike [1988]

```
counter:=prog(end: int, c: chan of int)
{
   i:int;
   for(i=2; i<end; i++)
      c<-=i;
};

filter:=prog(prime: int, listen: chan of
int, send: chan of int)
{
   i:int;
   for(;;)
      if((i=<-listen)%prime)
         send<-=i;
};
```

Newsqueak

Rob Pike [1988]

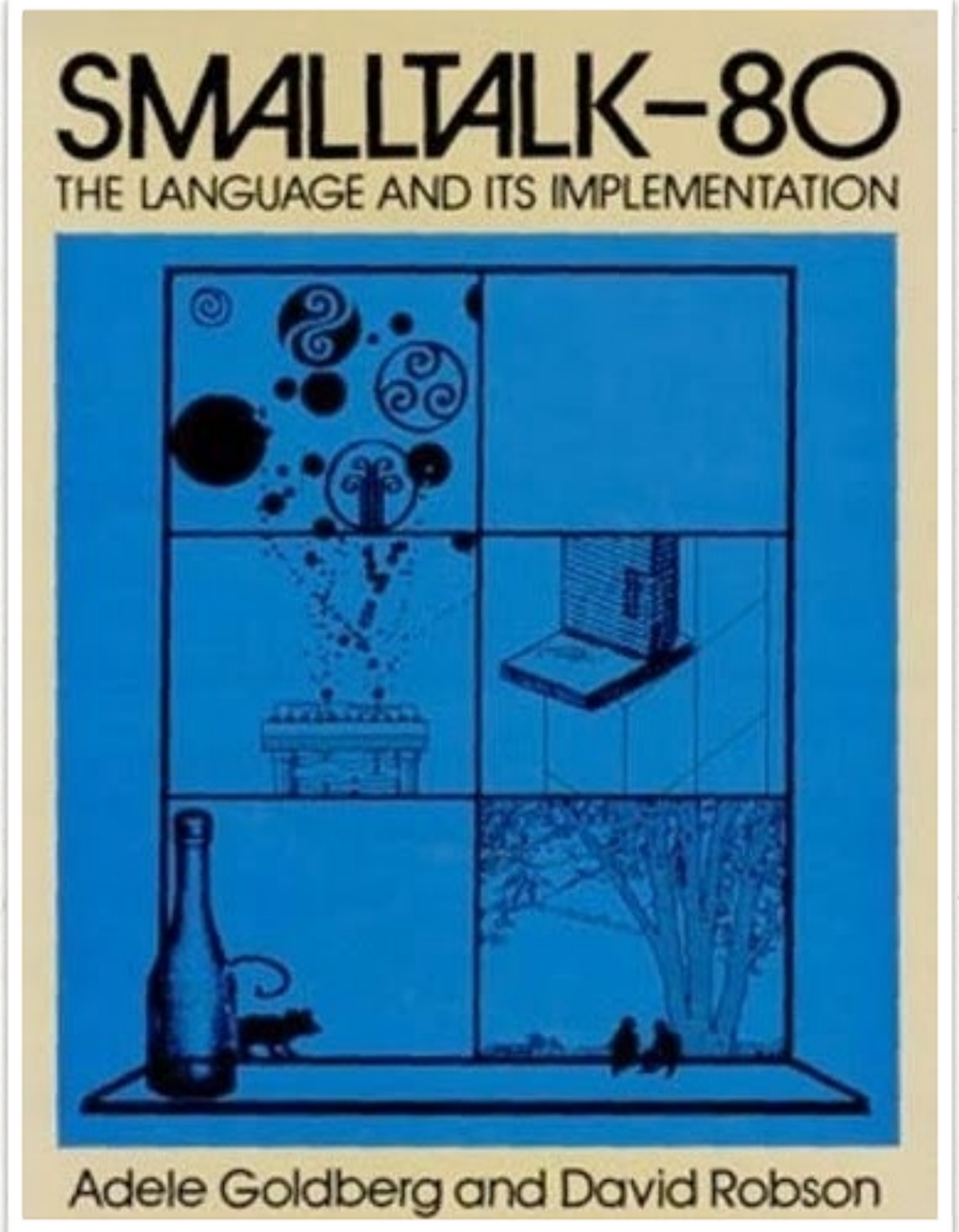
```
sieve:=prog(c: chan of int)
{
   for(;;){
      prime:=<-c;
      print(prime, " ");
      newc:=mk(chan of int);
      begin filter(prime, c, newc);
      c=newc;
   }
};

count:=mk(chan of int);

begin counter(10000, count);
begin sieve(count);
"";
```

Newsqueak

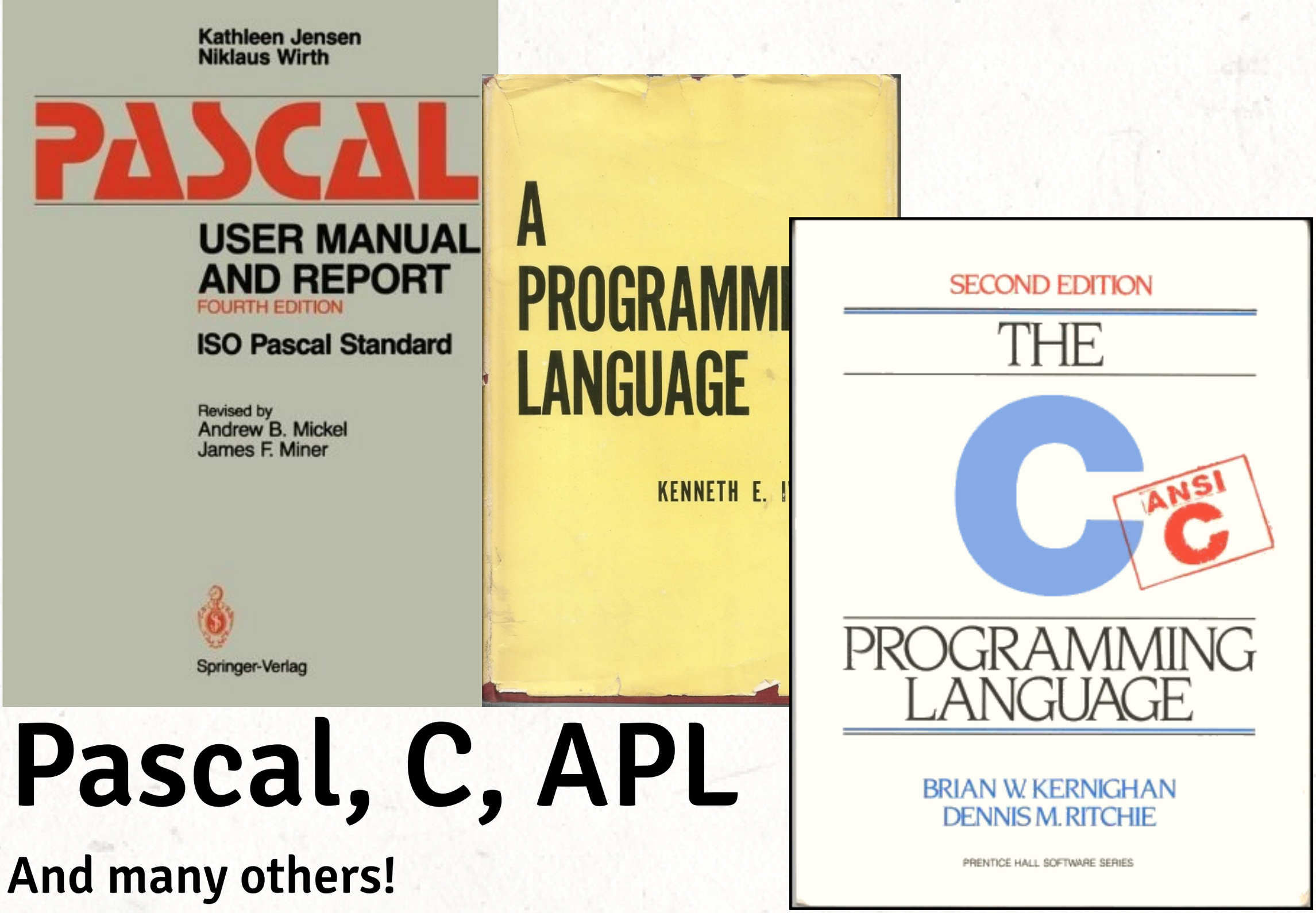
Rob Pike [1988]

Smalltalk

Alan Kay, et al [1976]

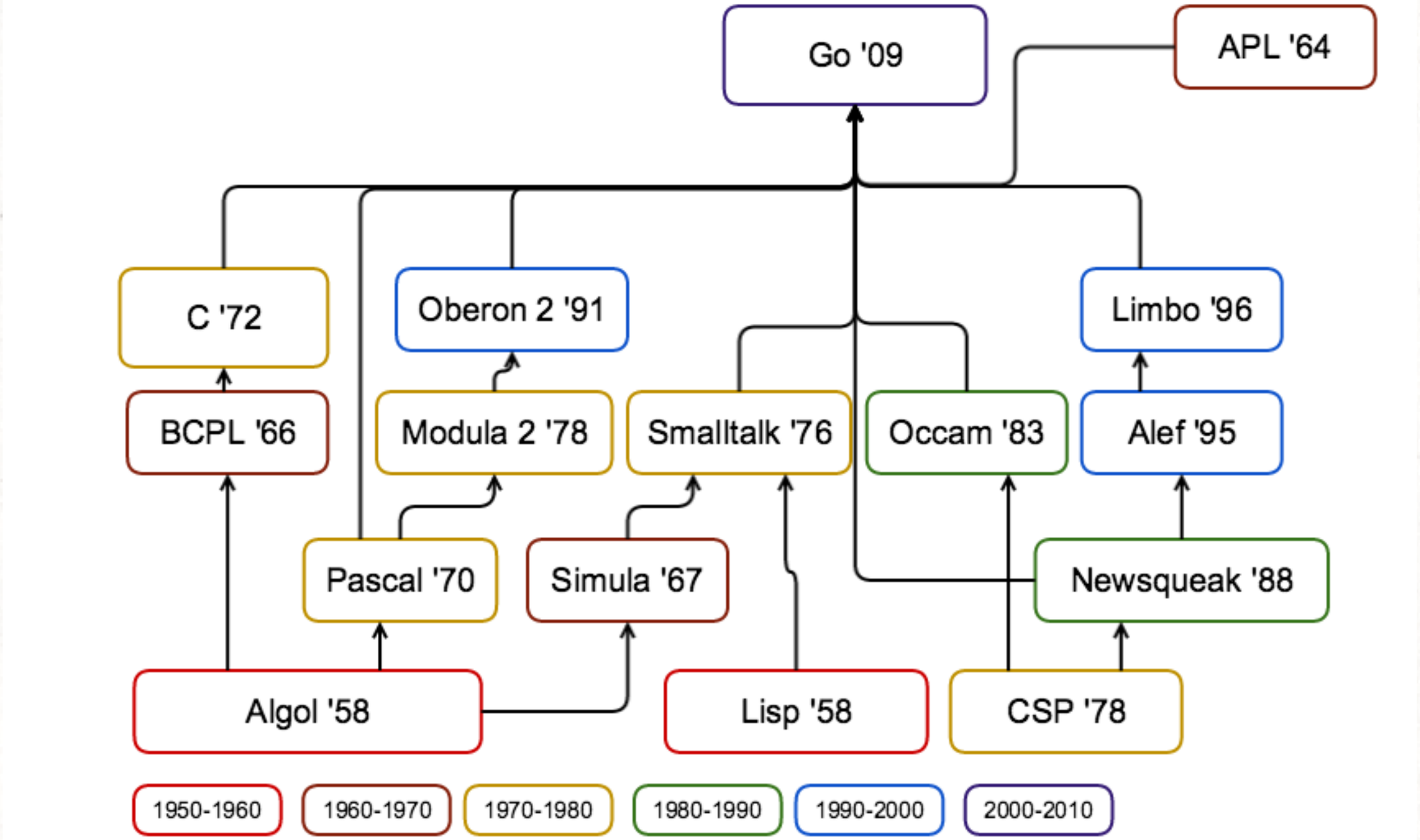


図式建築



Pascal, C, APL

And many others!



Go Genæology

図式建築

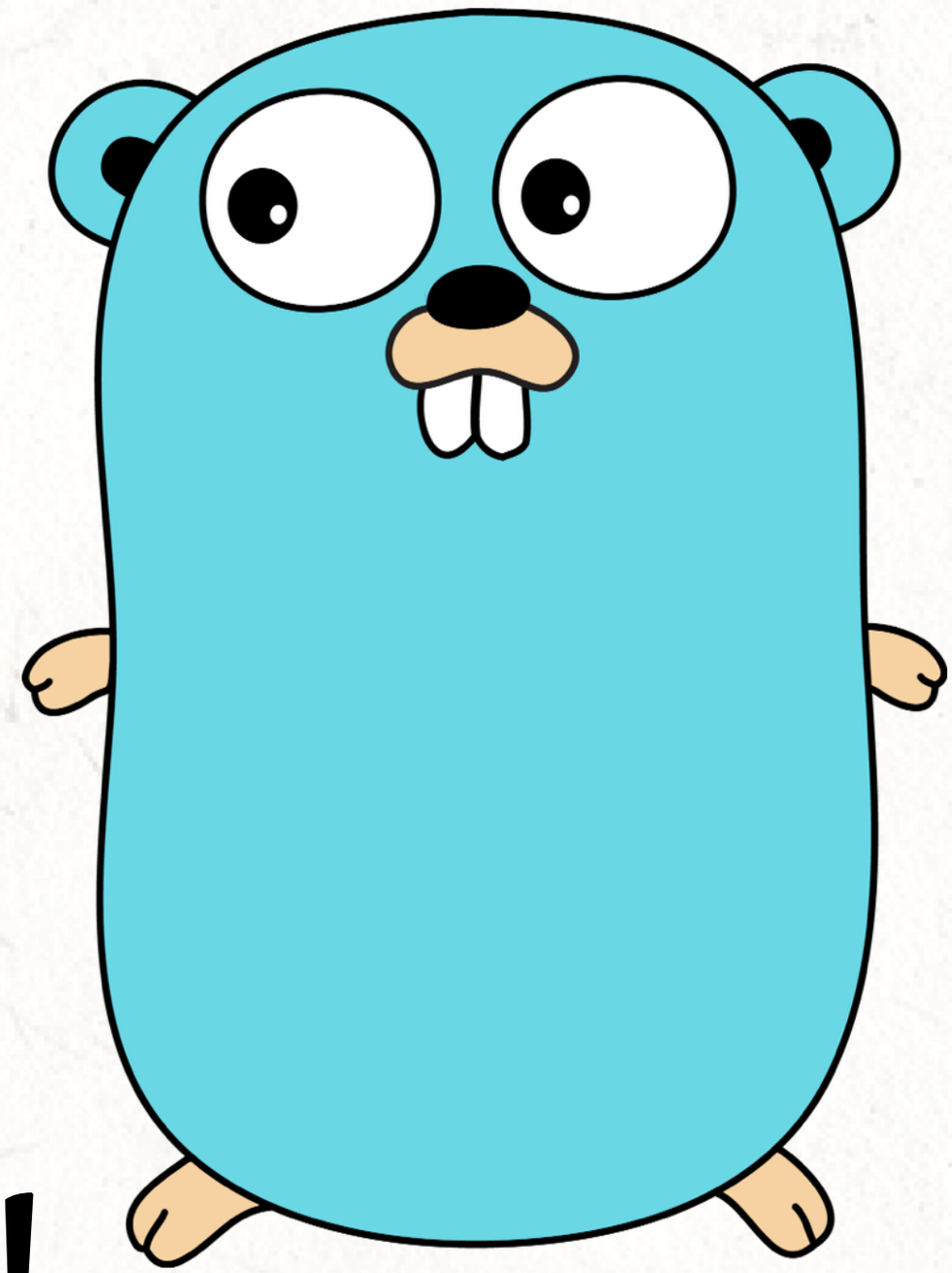
LESS Reinvention
Repetition

MORE

Progress

図式建築

Go forth, and build good software!



図式建築

Thank you!

@ghoseb