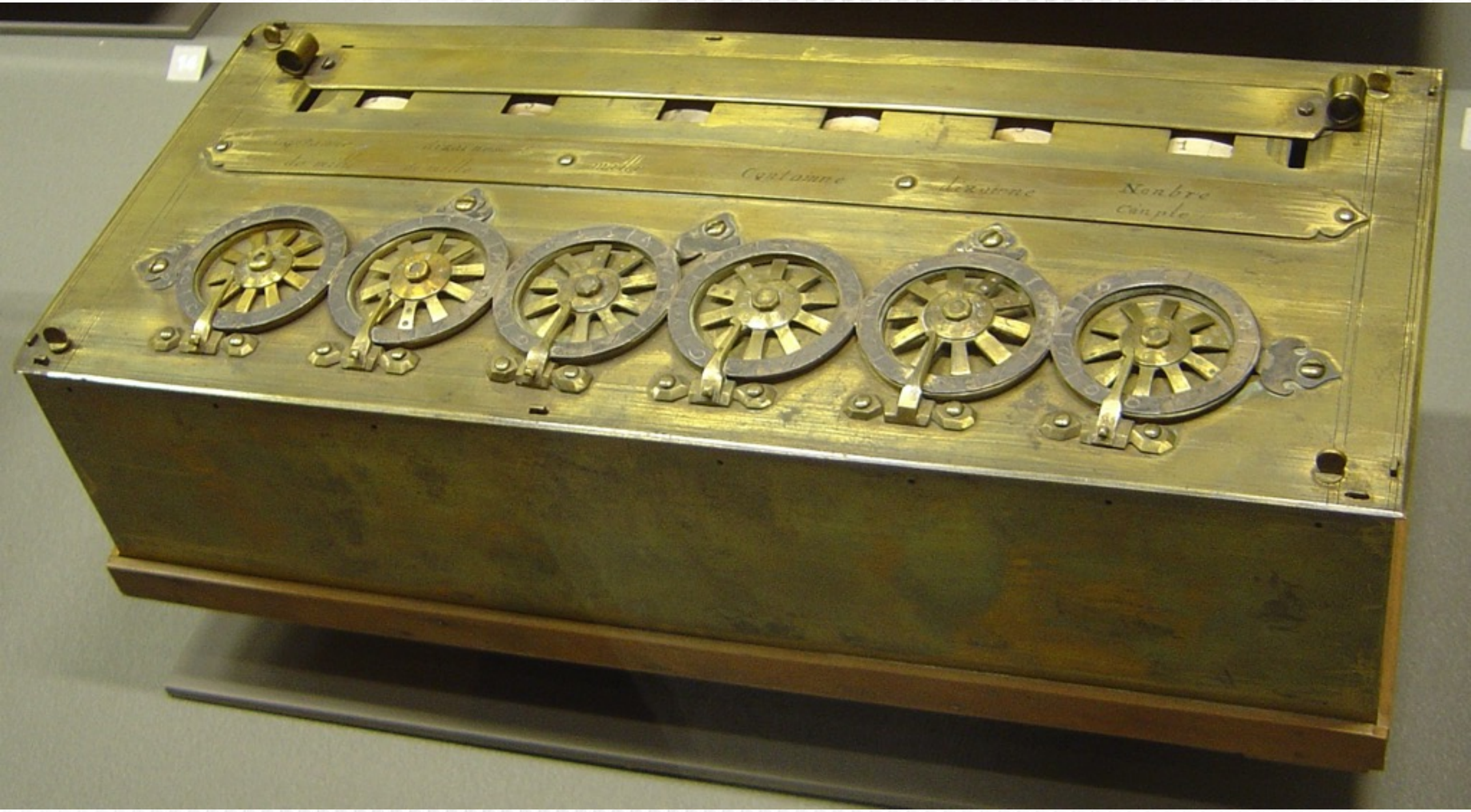
Linguistic History of Software Engineering

Vadim Zaytsev aka @grammarware,
Universiteit van Amsterdam, 2014

Vadim Zaytsev

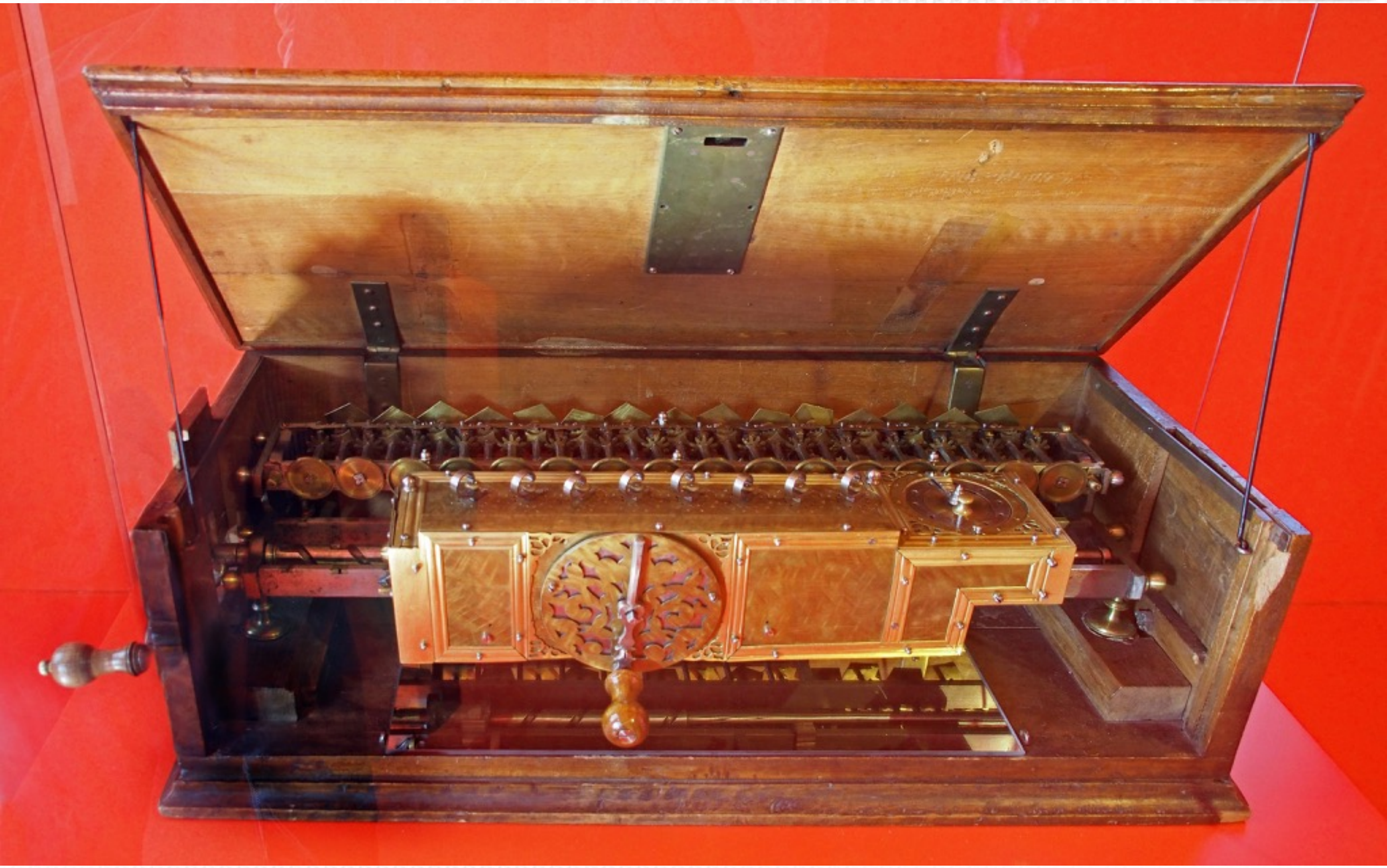
- Universiteit van Amsterdam (2013–2016?)
- Centrum Wiskunde & Informatica (2010–2013)
- Universität Koblenz-Landau (2008–2010)
- Vrije Universiteit Amsterdam (2004–2008)
- Universiteit Twente (2002–2004)
- Rostov State Transport University (1999–2008)
- Rostov State University (1998–2003)

1642



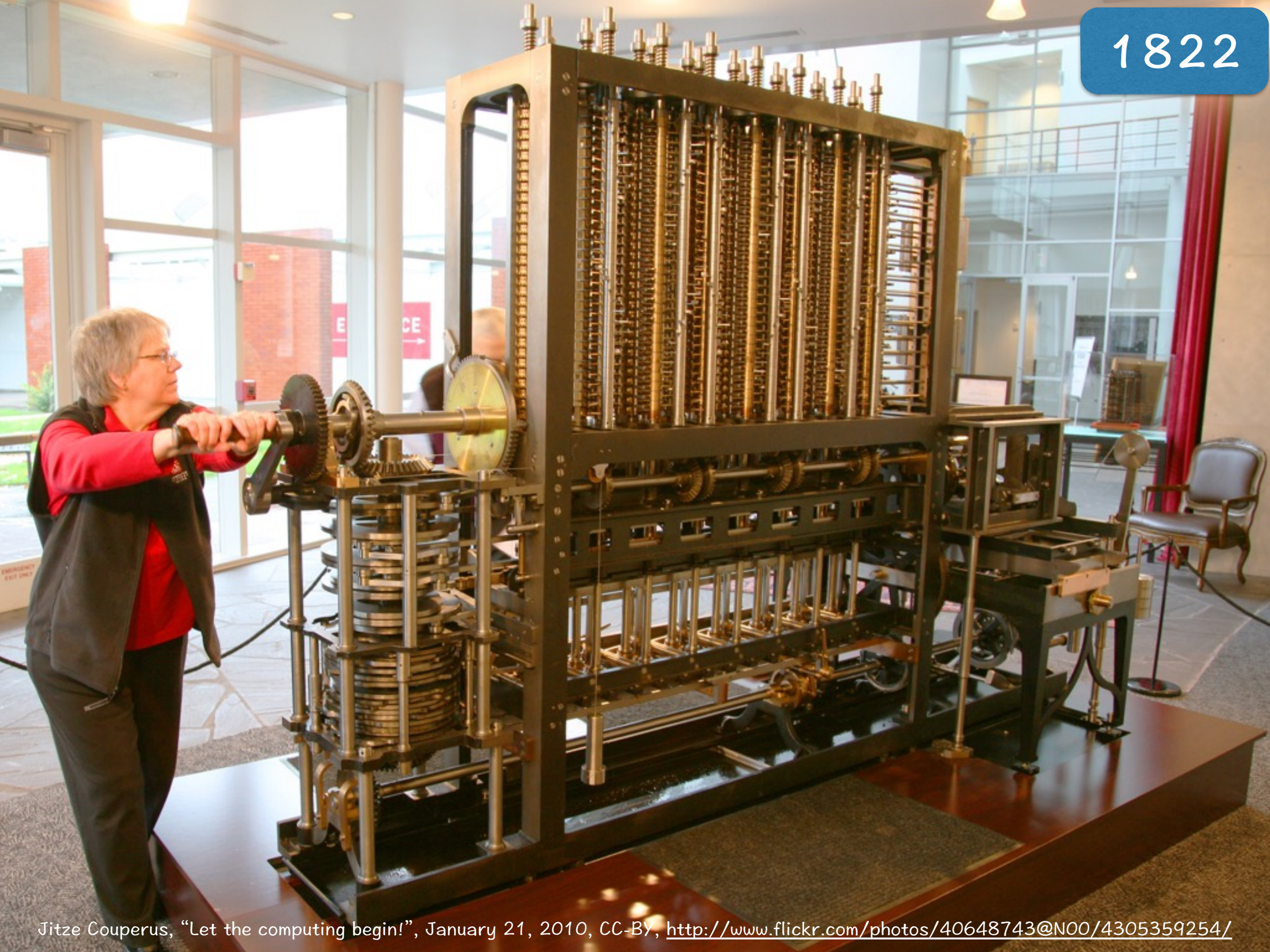
David Monniaux, “Machine à calculer de Blaise Pascal sans sous ni deniers”, https://commons.wikimedia.org/wiki/File:Arts_et_Metiers_Pascaline_dsc03869.jpg, CC-BY-SA

1673



Hajotthu, “Leibniz' Vier-Spezies-Rechenmaschine”,
https://commons.wikimedia.org/wiki/File:Leibniz_Rechenmaschine_(1690).jpg, CC-BY

1822



Jitze Couperus, “Let the computing begin!”, January 21, 2010, CC-BY, http://www.flickr.com/photos/40648743@N00/4305359254/

1805

These cards, however, have nothing to do with the regulation of the particular *numerical* data. They merely determine the *operations** to be effected, which operations may of course be performed on an infinite variety of particular numerical values, and do not bring out any definite numerical results unless the numerical data of the problem have been impressed on the requisite portions of the train of mechanism. In the above example, the first essential step towards an arithmetical result, would be the substitution of specific numbers for n, and for the other primitive quantities which enter into the function.

Again, let us suppose that for F we put two complete equations of the fourth degree between x and y. We must then express on the cards the law of elimination for such equations. The engine would follow out those laws, and would ultimately give the equation of one variable which results from such elimination. Various *modes* of elimination might be selected; and of course the cards must be made out accordingly. The following is one mode that might be adopted. The engine is able to multiply together any two functions of the form

$$a + bx + cx^2 + \ldots . px^n.$$

This granted, the two equations may be arranged according to the powers of y, and the coefficients of the powers of y may be arranged according to powers of x. The elimination of y will result from the successive multiplications and subtractions of several such functions. In this, and in all other instances, as was explained above, the particular *numerical* data and the *numerical* results are determined by means and by portions of the mechanism which act quite independently of those that regulate the *operations*.

In studying the action of the Analytical Engine, we find that the peculiar and independent nature of the considerations which in all mathematical analysis belong to *operations*, as distinguished from *the objects operated upon* and from the *results* of the operations performed upon those objects, is very strikingly defined and separated.

It is well to draw attention to this point, not only because its full appreciation is essential to the attainment of any very just and adequate general comprehension of the powers and mode of action of the Analytical Engine, but also because it is one which is perhaps too little kept in view in the study of mathematical science in general. It is, however, impossible to confound it with other considerations, either when we trace the manner in which that engine attains its results, or when we prepare the data for its attainment of those results. It were much to be desired, that when mathematical processes pass through the human brain instead of through the medium of inanimate mechanism, it were equally a necessity of things that the reasonings connected with *operations* should hold the same just place as a clear and well-defined branch of the subject of analysis, a fundamental but yet independent

* We do not mean to imply that the *only* use made of the Jacquard cards is that of regulating the algebraical *operations*. But we mean to explain that *those* cards and portions of mechanism which regulate these *operations*, are wholly independent of those which are used for other purposes. M. Menabrea explains that there are *three* classes of cards used in the engine for three distinct sets of objects, viz. *Cards of the Operations*, *Cards of the Variables*, and certain *Cards of Numbers*. (See pages 678 and 687.)

1843

Ada Lovelace (1815–1852)

"Sketch of the Analytical Engine Invented by Charles Babbage, Esq.," Scientific Memoirs:3, "Notes by the Translator", https://en.wikisource.org/wiki/Page%3AScientific_Memoirs%2C_Vol._3_(1843).djvu/702

1954

8

1954

9

1952

1952

Friden²

REP

NON ENT.

CTR CON

NEG MULT SQ

ACCUM MULT SQ

MULT CLEAR

MULT SQ

0

LOCK

DIV STOP

ADD

ENTER DIVD

+

÷

R R CLEAR

CARR CLEAR

photo credit: http://academictree.org/physics/peopleinfo.php?pid=25212

195x

Adriaan J. van Wijngaarden (1916–1987)

photo credit: http://www.kennislink.nl/publicaties/rekenmeisjes-en-rekentuig

1944

1976

Konrad Zuse
Plankalkül

```
1    A2 = (A9, AΔ1)
2    P1      | R(V)  ⇒  R
3            V| 0      0
4            A| Δ1     Δ1
5            | √|V| + 5 x V³  ⇒  R
6            V| 0             0     0
7            A| Δ1            Δ1    Δ1
8    P2      | R(V)  ⇒  R
9            V|   0      0
10           A| 11 x Δ1  11 x 2
11           | W2(11)[ R1(V)  ⇒  Z
12           V|      [ 0  0      0
13           K|      [    i
14           A|      [    Δ1     Δ1
15           |       [ Z > 400  →  (i,+∞)  ⇒  R[(10-i)
16           V|      [ 0                      0
17           K|      [
18           A|      [ Δ1              9      2        9
19           |       [ Z > 400  →  (i, Z)  ⇒  R[(10-i)
20           V|      [ 0                0     0
21           K|      [
22           A|      [ Δ1              9 Δ1   2        9
```

13. 'Boilenveld'

12

Adele Katz Goldstine (1920–1964)

1946

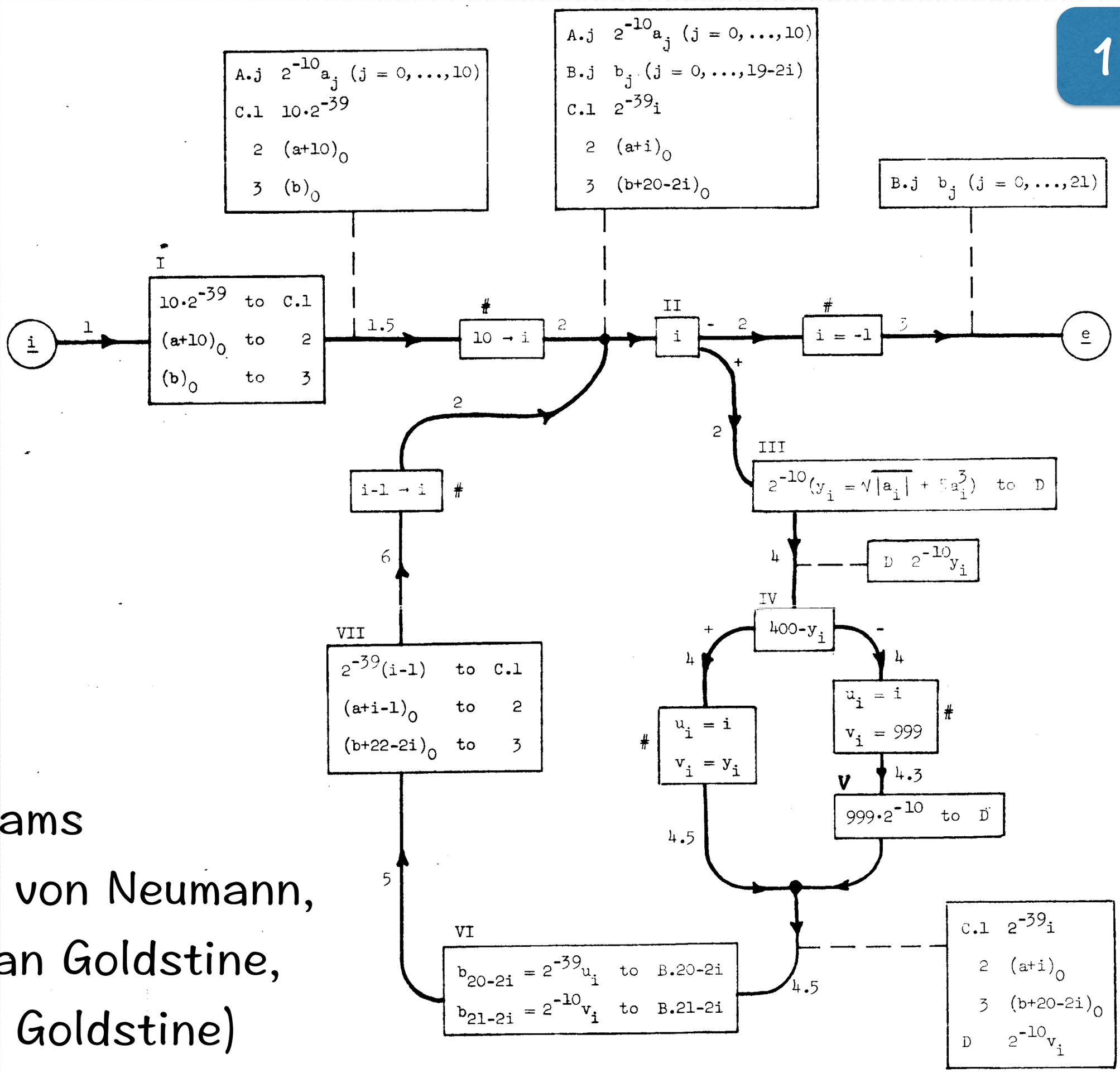
photo credit: http://nurdangolbek.blogspot.com/2012/08/teknoloji-tarihindeki-15-onemli-kadn.html

1945

John von Neumann (1903–1957)

1947

Flow Diagrams (John von Neumann, Herman Goldstine, Adele Goldstine)



1950

"the technique of program composition can be mechanised"

Haskell Brooks Curry (1900–1982)

John Mauchly, William Schmitt

1950

		Equations	Coded representation
00		i = 10	00 00 00 W0 03 Z2
01	0:	y = (√ abs t) + 5 cube t	T0 02 07 Z5 11 T0
02			00 Y0 03 09 20 06
03		y 400 if≤to 1	00 00 00 Y0 Z3 41
04		i print, 'TOO LARGE' print-and-return	00 00 Z4 59 W0 58
05		0 0 if=to 2	00 00 00 Z0 Z0 72
06	1:	i print, y print-and-return	00 00 Y0 59 W0 58
07	2:	T0 U0 shift	00 00 00 T0 U0 99
08		i = i-1	00 W0 03 W0 01 Z1
09		0 i if≤to 0	00 00 00 Z0 W0 40
10		stop	00 00 00 00 ZZ 08

Short Code

1952

```
1    Für i = 10(-1)0
2    a_i ⇒ t
3    (Sqrt Abs t) + (5 x t x t x t) ⇒ y
4    Max(Sgn(y-400), 0) ⇒ h
5    Z 0_i ⇒ b_20-2i
6    (h x 999) + ((1-h) x y) ⇒ b_21-2i
7    Ende Index i
8    Schluss
```

Heinz Rutishauser (1918–1970)

photo credit: Висарик, CC-BY-SA, https://commons.wikimedia.org/wiki/File:Aleksandr_Brudno.JPG

1954

Alexander Brudno (1918–2009)

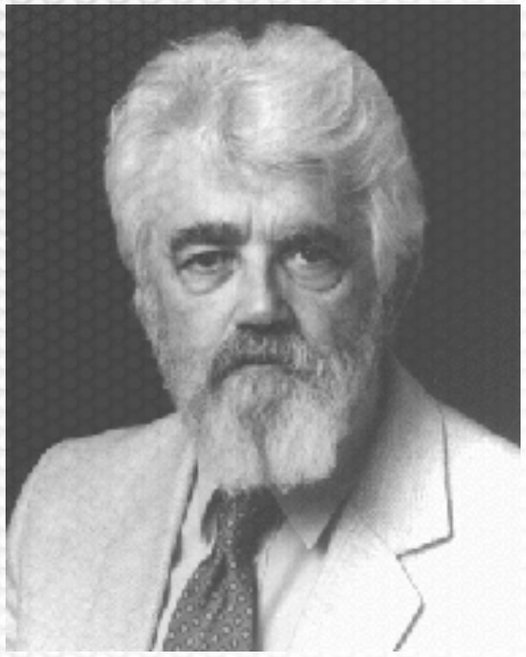
Перед каждой командой в кодированной форме Брудно писал еще ее адрес. Получилось так:

					A	
	$0+0=S$	1003	001	0	0	2570
	$1 \to n$	1004	056	0101	2571	1006
	$n+1=n$	1005	001	2571	0101	2571
	$n \times n = R1$	1006	005	2571	2571	0011
	$1 : R1 = R2$	1007	004	0101	0011	0012
	$S+R2=S$	1010	001	2570	0012	2570
	$n < 10$	1011	036	2571	0112	1005
	Стоп	1012	017	0	0	0
	1	0101	101	4000	0	0
	10	0112	104	5000	0	0

Теперь, чтобы запомнить, какую ячейку отвели данной букве, и чтобы ячейки не налезали друг на друга, понадобилась шпаргалка (роспись памяти).

Она выглядела так:

1260	1261	1252	1263	1254
1300 *a*	1301 *b*	1302 *c*	1303 17,5	1304
1320 Δ	1321 *R1*	1322 *R2*	1323	1324
1340	1341 *Блок подготовки*	1342	1343	1344
1360	1361	1362	1363	1364
1400	1401	1402	1403	1404
1420	1421	1422	1423	1424



(from A. S. Kronrod, "Discussions about Programming", 1963)

AIKEN - IBM AUTOMATIC SEQUENCE CONTROLLED CALCU

photo credit: http://techie.com/amazing-grace/ / personal.psu.edu

Grace Murray Hopper (1906–1992)

1957

```
(1)    READ-ITEM A(11) .
(2)    VARY I 10(-1)0 SENTENCE 3 THRU 10 .
(3)    J = I+1 .
(4)    Y = SQR |A(J)| + 5*A(J)^3 .
(5)    IF Y > 400, JUMP TO SENTENCE 8 .
(6)    PRINT-OUT I, Y .
(7)    JUMP TO SENTENCE 10 .
(8)    Z = 999 .
(9)    PRINT-OUT I, Z .
(10)   IGNORE .
(11)   STOP .
```

MATH-MATIC

1958

```
(1)   COMPARE PART-NUMBER (A) TO PART-NUMBER (B) ; IF GREATER GO TO
      OPERATION 13 ; IF EQUAL GO TO OPERATION 4 ; OTHERWISE GO TO
      OPERATION 2 .
(2)   READ-ITEM B ; IF END OF DATA GO TO OPERATION 10 .
```

FLOW-MATIC

code by: Rick Malek, Data Access with ADO.NET,
http://www.csharphelp.com/2006/12/data-access-with-ado-net/

1959

```
 MOVE B"1" TO MY-BOOLEAN. *>Set to True to begin with
 MOVE ZEROS TO MY-COUNT.
*> Primer read to see if we have data to read back
 INVOKE DATAREADEROBJ "Read" RETURNING MY-BOOLEAN. *> True if data, False if none
 PERFORM WITH TEST BEFORE UNTIL MY-BOOLEAN NOT EQUAL B"1"
   INVOKE DATAREADEROBJ "GetString" USING 0 RETURNING FIRSTNAME
   INVOKE DATAREADEROBJ "GetString" USING 1 RETURNING LASTNAME
   INVOKE DATAREADEROBJ "GetString" USING 2 RETURNING HOMEPHONE
   ADD 1 TO MY-COUNT
   DISPLAY "Record: ", MY-COUNT, " NAME: ", FIRSTNAME, " ", LASTNAME, " ", HOMEPHONE
   INVOKE DATAREADEROBJ "Read" RETURNING MY-BOOLEAN *> True if data, False if none
END-PERFORM.
```

COBOL

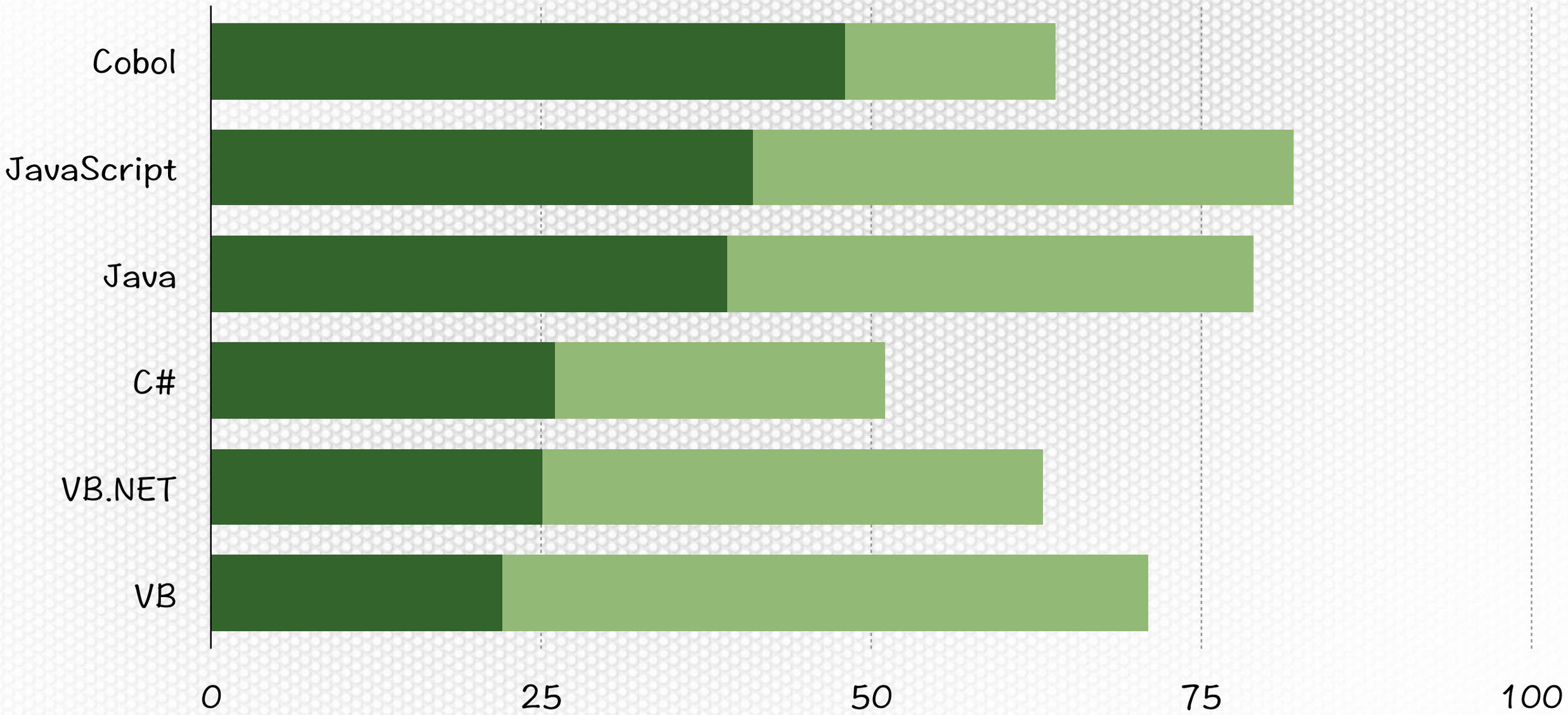
1996

"…it's only a matter of time before all the existing COBOL programmers die of old age…"

photo credit: Ed Yourdon, CC-BY-SA,
https://commons.wikimedia.org/wiki/File:Ed_Yourdon_(1970s).jpg
https://commons.wikimedia.org/wiki/File:Yourdon,_Edward_(2008).jpg

2012

To what extent do you use these languages?



Cobol brain drain: Survey results, http://www.computerworld.com/article/2502430/data-center/cobol-brain-drain--survey-results.html

1957

```
FTN 5.1+642        87/02/02. 10.44.20 PAGE    18
SUBROUTINE RCKELI    74/175  OPT=0,ROUND= A/ S/ M/-D,-DS
       DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER
       /-FIXED,DB= TB/ SB/ SL/ ER/-ID/ PMD/-ST,-AL,PL=5000
       FTN5,I=CHRLGS,L,LO,DB,PN,PW=90,PS=80.

      1       SUBROUTINE RCKELI(N,A,Y,X)
      2       PARAMETER (IA=50)
      3       REAL A(IA,IA), Y(IA), X(IA)
      4       INTEGER N
      5 *                                                                    *
      6 **********************************************************************
      7 *                                                                    *
      8 * SUBROUTINE ZUR RUECKWAERTSELIMINATION, D.H. ZUR LOESUNG VON        *
      9 *                R*X=Y                                               *
     10 *                                                                    *
     11 * ERKLAERUNG DER FORMALEN PARAMETER:                                 *
     12 *                                                                    *
     13 * NAME    TYP      DIMENSION  ERKLAERUNG                             *
     14 * ------------------------------------------------------------------ *
     15 * N       INTEGER      0      DIMENSION DER MATRIX A (E)             *
     16 * A       REAL         2      DREICKSZERLEGUNG DER MATRIX A          *
     17 *                              (UEBERSPEICHERT) (E)                  *
     18 * Y       REAL         1      VEKTOR, DER DIE LOESUNG VON            *
     19 *                              L*Y=B ENTHAELT (E)                    *
     20 * X       REAL         1      VEKTOR ZUR AUFNAHME DER LOESUNG VON    *
     21 *                              R*X=Y (A)                             *
     22 *                                                                    *
     23 *                                                                    *
     24 *                                                                    *
     25 *                                                                    *
     26 *   E: EINGABEPARAMETER                                              *
     27 *   A: AUSGABEPARAMETER                                              *
     28 *                                                                    *
     29 * ================================================================== *
     30 *                                                                    *
     31 * KONSTANTEN ('PARAMETER' IN FORTRAN):                               *
     32 *                                                                    *
     33 *   IA: MAXIMALE DIMENSION DER MATRIX A                              *
     34 *                                                                    *
     35 * ================================================================== *
     36       X(N)=Y(N)/A(N,N)
     37       DO 10 K=N-1,1,-1
     38          S=0.
     39          DO 20 J=K+1,N
     40    20       S=S+A(K,J)*X(J)
     41    10    X(K)=(Y(K)-S)/A(K,K)
     42       RETURN
     43       END


--VARIABLE MAP--(LO=A/R)
 -NAME---ADDRESS --BLOCK-----PROPERTIES-------TYPE---------SIZE---REFERENCES-

  A             2   DUMMY-ARG                    REAL          2500       1      3     36
                                                                          40     41
 J           2018                                INTEGER                39/C   40     40
  K           176B                                INTEGER                37/C   39/C   40
                                                                          41     41     41
                                                                          41
  N             1   DUMMY-ARG                    INTEGER                 1      4     36
                                                                          36     36     36
                                                                          37/C   39/C
  S           200B                                REAL                   38/S   40     40/S
                                                                          41
  X             4   DUMMY-ARG                    REAL            50      1      3     36/S
                                                                          40     41/S
  Y             3   DUMMY-ARG                    REAL            50      1      3     36
                                                                          41


--SYMBOLIC CONSTANTS--(LO=A/R)
 -NAME----TYPE-----------------------VALUE---REFERENCES-

  IA       INTEGER                      50       2/S    3      3      3      3
```

John Backus (1924–2007)

Phrontis, FORTRAN-77 program with compiler output, CC-BY-SA, https://commons.wikimedia.org/wiki/File:Ftn-elim-1240x1709.jpg

1957

PROGRAMMER'S REFERENCE MANUAL

Fortran

AUTOMATIC CODING SYSTEM

FOR THE IBM 704

John Backus (1924–2007)

Phrontis, FORTRAN-77 program with compiler output, CC-BY-SA, https://commons.wikimedia.org/wiki/File:Ftn-elim-1240x1709.jpg

1958

```
proc abs max = ([,]real a, ref real y, ref int i, k)real:
comment The absolute greatest element of the matrix a, of size ⌜a by 2⌜a
is transferred to y, and the subscripts of this element to i and k; comment
begin
    real y := 0; i := ⌞a; k := 2⌞a;
    for p from ⌞a to ⌜a do
      for q from 2⌞a to 2⌜a do
        if abs a[p, q] > y then
            y := abs a[p, q];
            i := p; k := q
        fi
      od
    od;
    y
end # abs max #
```

1958

"a language so far ahead of its time that it was not only an improvement on its predecessors but also on nearly all its successors"

```
proc abs max = ([,]real a, ref real y, ref int i, k)real:
comment The absolute greatest element of the matrix a, of size ⌜a by 2⌜a
is transferred to y, and the subscripts of this element to i and k; comment
begin
    real y := 0; i := ⌞a; k := 2⌞a;
    for p from ⌞a to ⌜a do
      for q from 2⌞a to 2⌜a do
        if abs a[p, q] > y then
            y := abs a[p, q];
            i := p; k := q
        fi
      od
    od;
    y
end # abs max #
```

Tony Hoare (b. 1934)

photo credits: http://www.cs.cornell.edu/gries/banquets/iticse2002/iticse2002.html

1958

BNF

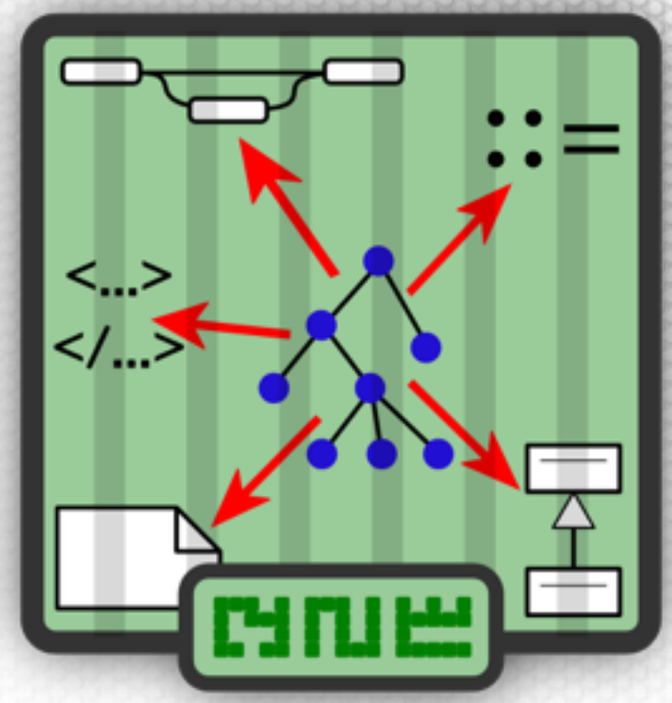
```
compilation ::=
        compilation_unit*
```

```
compilation_unit ::=
        visibility_restriction? "separate"? unit_body
```

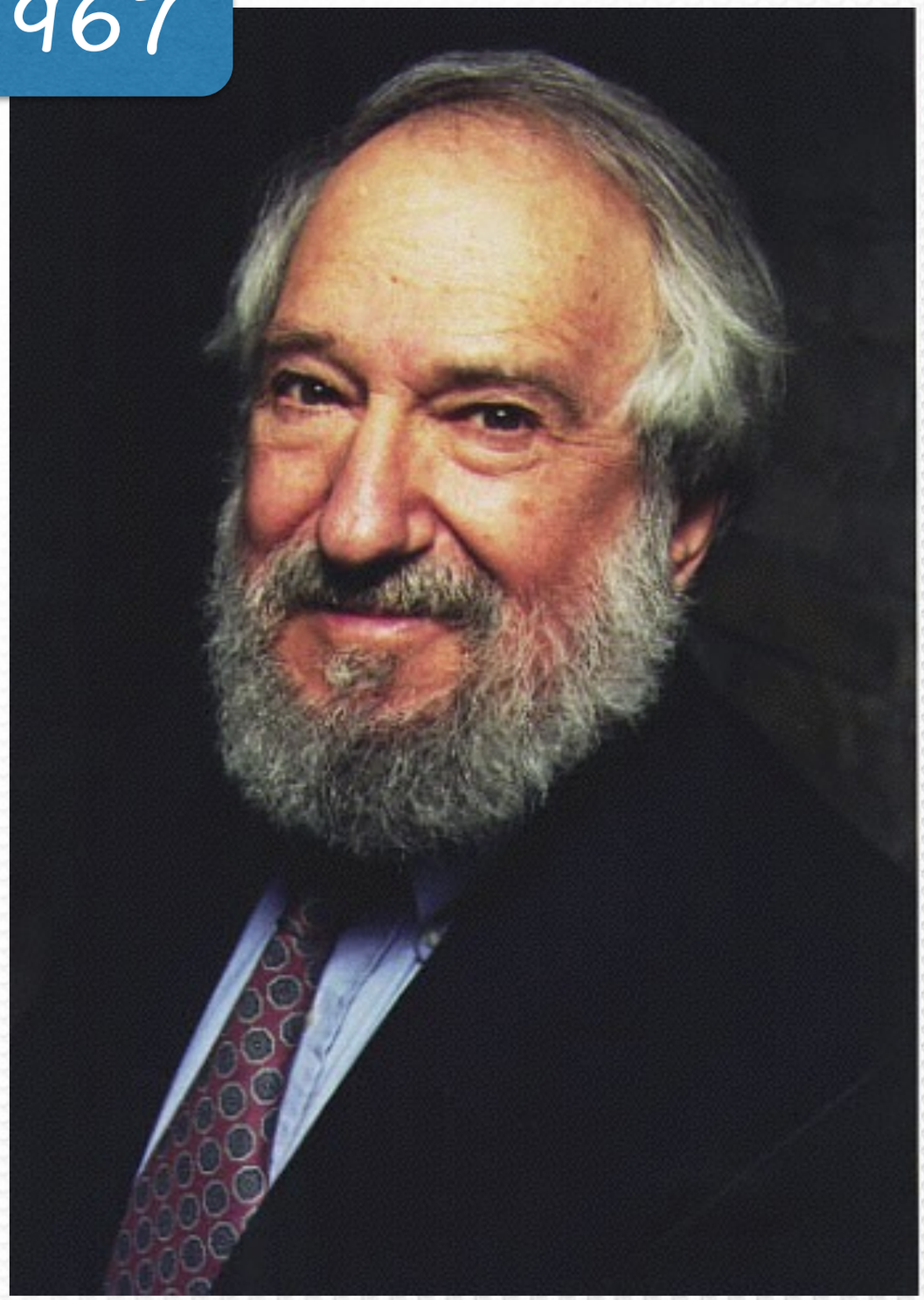
```
visibility_restriction ::=
        "restricted" visibility_list?
```

```
visibility_list ::=
        "(" ⟨unit_name⟩:name ("," ⟨unit_name⟩:name)* ")"
```

```
unit_body ::=
        subprogram_body
        module_specification
        module_body
```



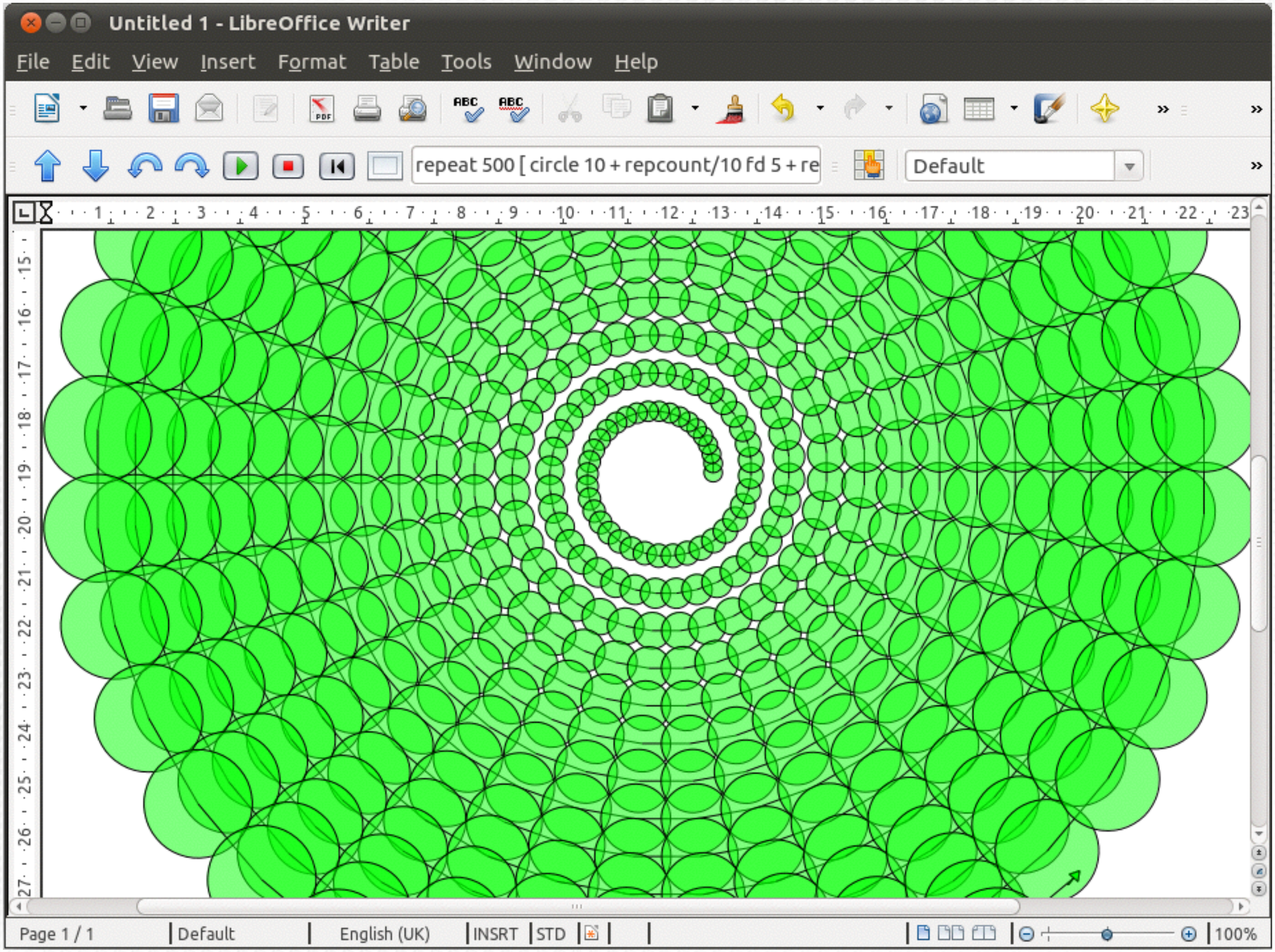
1967



Seymour Papert
(b. 1928)

1974

Radia Perlman
(b. 1951)



https://www.libreoffice.org/download/4-0-new-features-and-fixes, GPL

2006

Scratch, http://scratch.mit.edu

1964

Rauner Special Collection Library, GOTO LINE 1, http://raunerlibrary.blogspot.nl/2011_10_30_archive.html

IDEs

- 40% time spent editing code
- 50% efficiency boost with better tools
- We want:
  - editing support, code navigation
  - debugging, profiling
  - build integration, versioning
  - quality assurance, acceptance testing

claims from: Steve McConnell, Code Complete, 2ed, 2004. www.cc2e.com

2003

Everything is a…

- …machine?
- …bit string?
- …word?
- …object?
- …model?

Jean Bézivin

Eelco Visser, “Jean Bezivin”, https://www.flickr.com/photos/eelcovisser/4772370824/

Metamodel

conforms to

represented by

System

Model

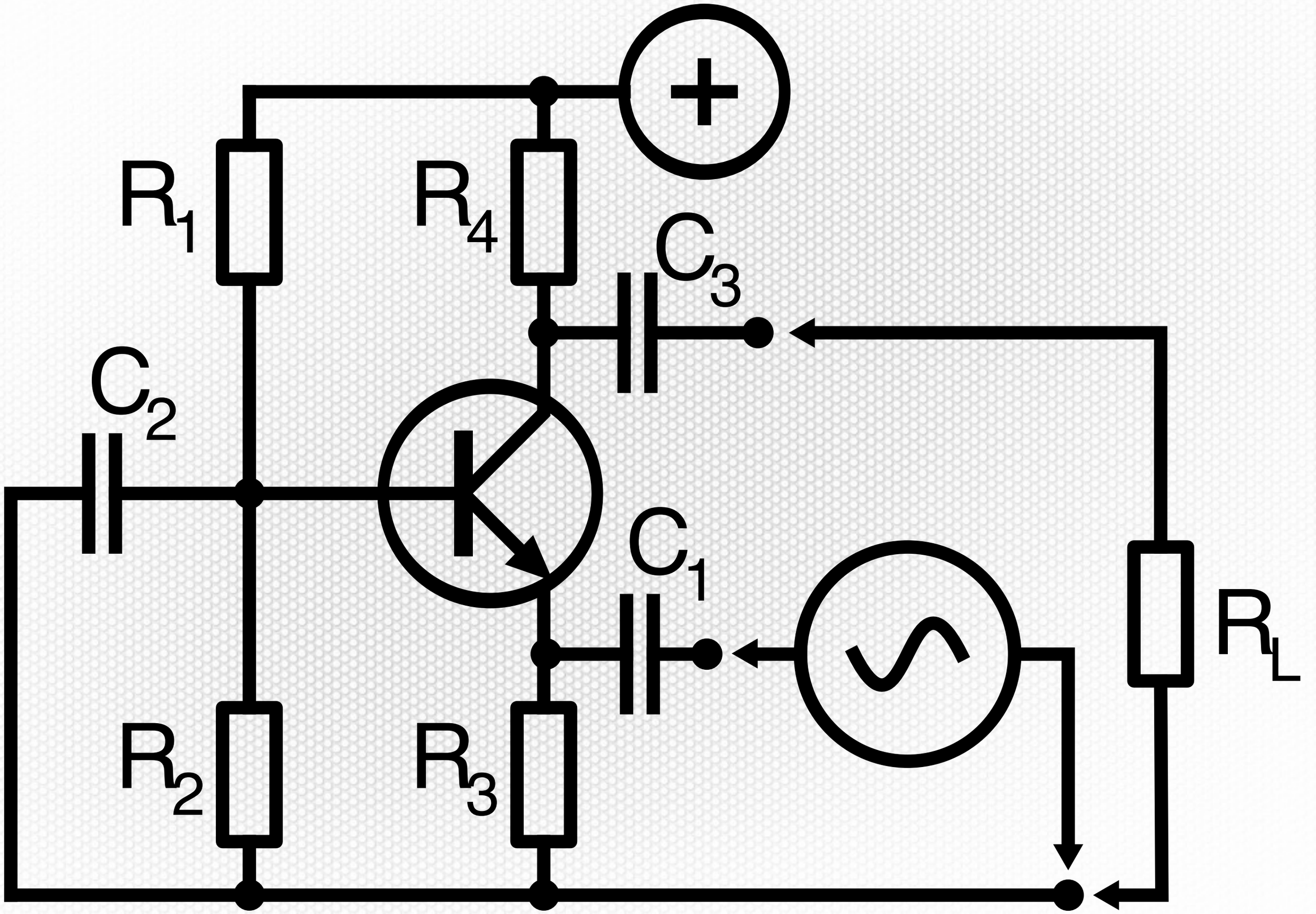
conforms to

represented by

Leipnizkeks, https://commons.wikimedia.org/wiki/File:2_Pfeifen.jpg, CC-BY-SA

René Magritte, The Treachery of Images, 1929


VIA 9GAG.COM
Ceci n'est pas un chat


+
R1
R4
C3
C2
C1
R2
R3
RL

2014

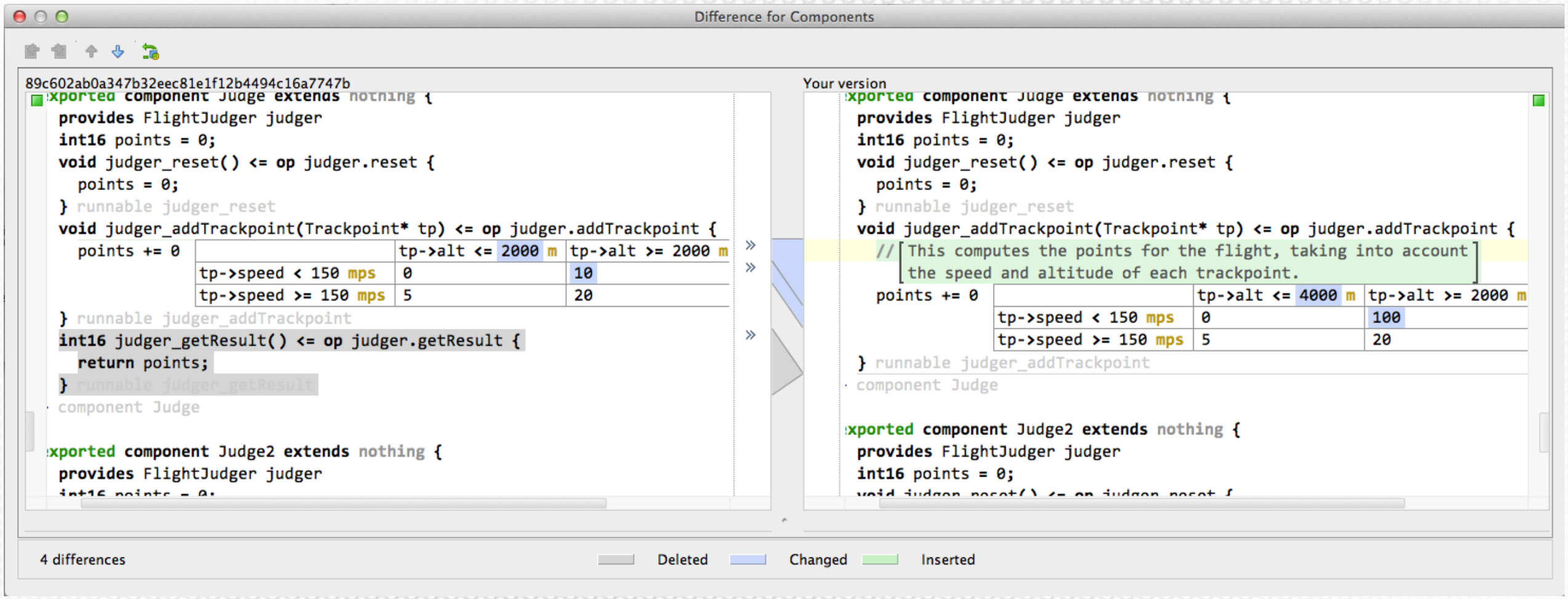
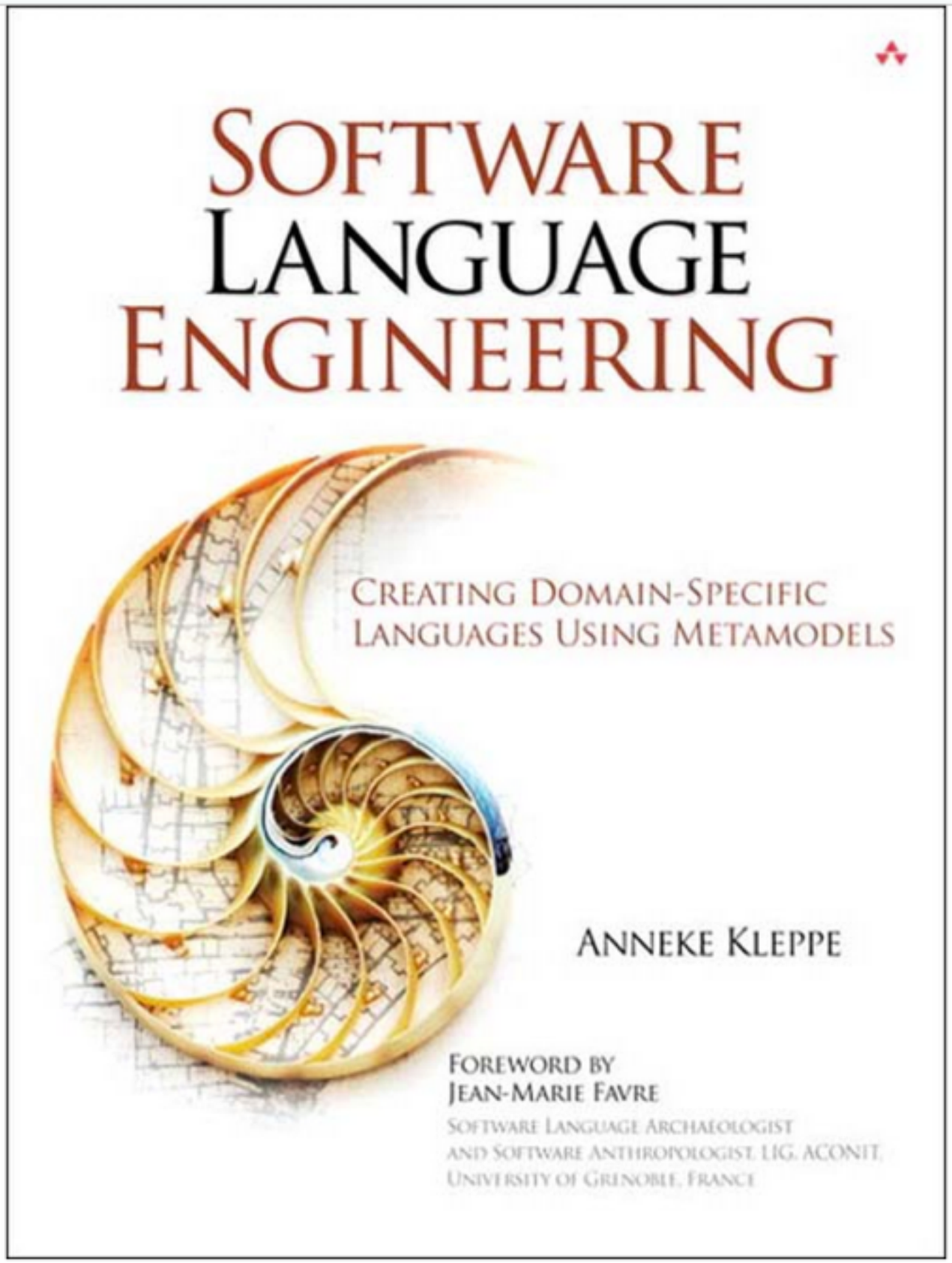


Figure 4.20 This dialog shows the diff between the local version and the latest version from git before a commit. The local version has changed two of the values in the decision table, has added a comment and has removed one runnable.

photo credit: Eelco Visser, https://www.flickr.com/photos/eelcovisser/2006308962/

2008



2009

The Pragmatic Programmers

Language Implementation Patterns

Create Your Own Domain-Specific and General Programming Languages

Terence Parr

2013

DSL Engineering

Designing, Implementing and Using Domain-Specific Languages

Markus Voelter

with Sebastian Benz, Christian Dietrich, Birgit Engelmann
Mats Helander, Lennart Kats, Eelco Visser, Guido Wachsmuth

dslbook.org

2014

Software languages

- Not (just) textual
- Exist in many formats
- (Now) easy to develop
- Easy to map to one another
- Language-parametric methods

Grammars in a broad sense

- Exist in various forms and formats
  - metamodels, schemata, domain models, class dictionaries
- Grammar recovery, convergence, transformation, …
- Grammars as contracts/commitments
  - for people and tools

Language workbenches

- Extensive IDE support beyond compilation
- Syntax highlighting & visualisation
- Interactive recommendations
- Beyond textual
- TDD and debugging

```
test bool testCode0() = unambiguous("");
test bool testCode1() = unambiguous("int x");
test bool testCode2() = unambiguous(":");
test bool testCode3() = unambiguous(":=");
test bool testCode3a() = unambiguous(":= ");
test bool testCode3b() = parses(":= ");
test bool testCode4() = unambiguous(": =");
test bool testCode5() = unambiguous("int FACES = 6;");
test bool testCode6() = unambiguous("1");
test bool testCode6a() = unambiguous("1 ");
test bool testCode7() = unambiguous("1 //");
test bool testCode7a() = parses("1 //");
test bool testCode8() = unambiguous("//");
test bool testCode9() = unambiguous("1 // comment");
test bool testCode9a() = parses("1 // comment");
```

Milestone summary

- Universal hardware + programs
- Automated code generation
- Programming with words
- Language documentation
- Domain-specific languages
- Merging/bridging disciplines: MDE, PLT, CC, TT, CT, NLP…

Questions?