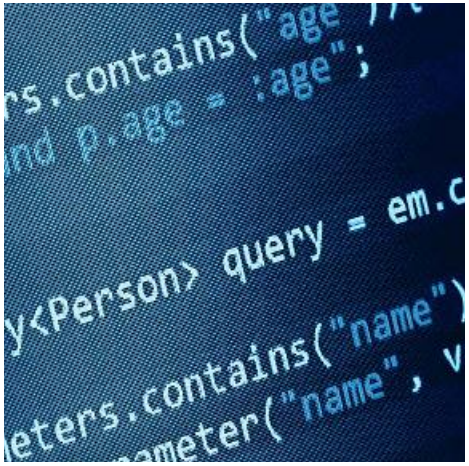
CS3500 Software Engineering



2017/2018

Dept. Computer Science
Dr. Klaas-Jan Stol

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Welcome to CS3500

The Early Days of Computing

Contents

After studying this material and associated papers, you should be able to:

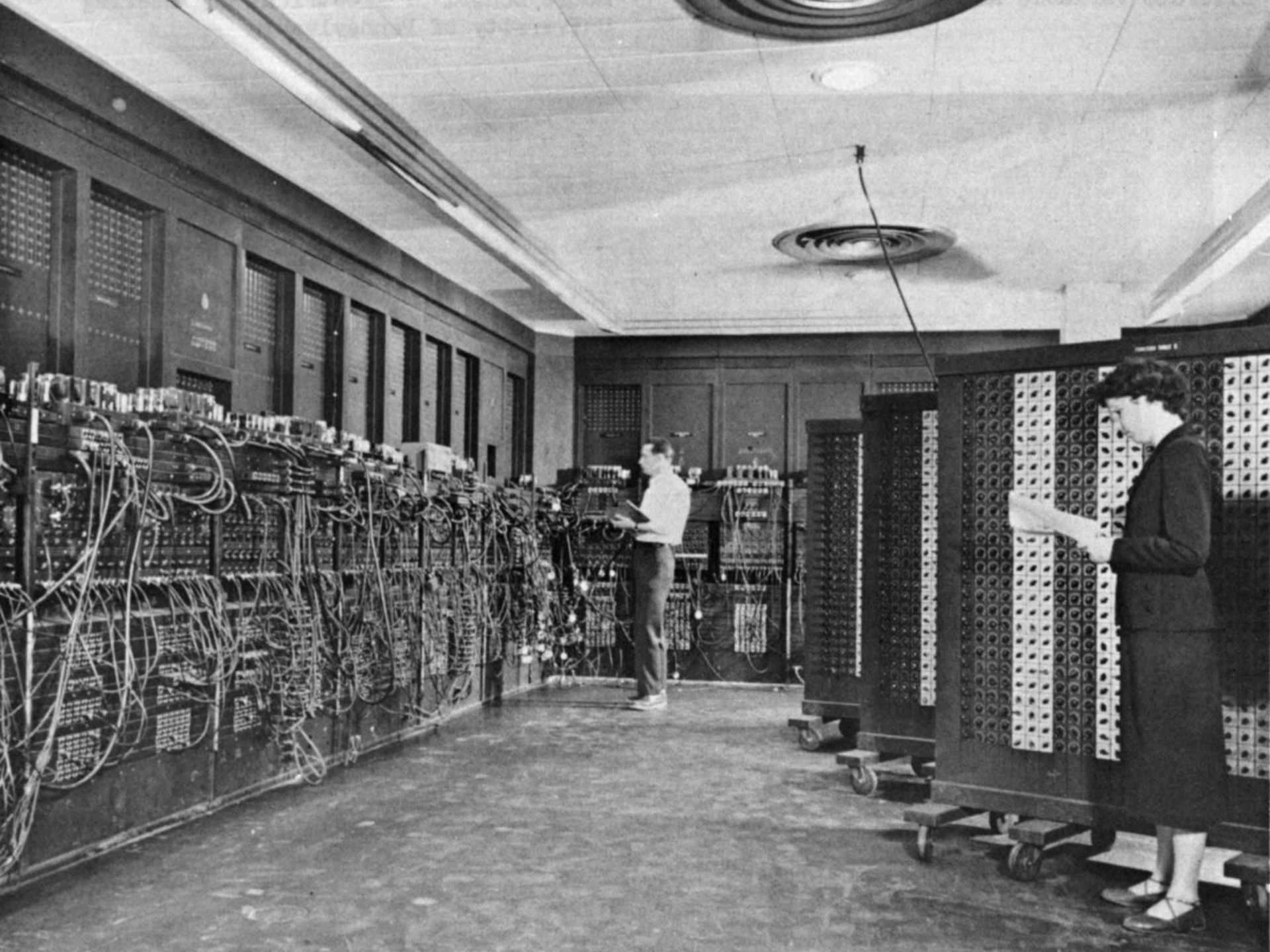
- Appreciate the early problems of programming.
- Describe the origin of the term “software engineering.”
- Describe the three key problems associated with the “Software Crisis.”

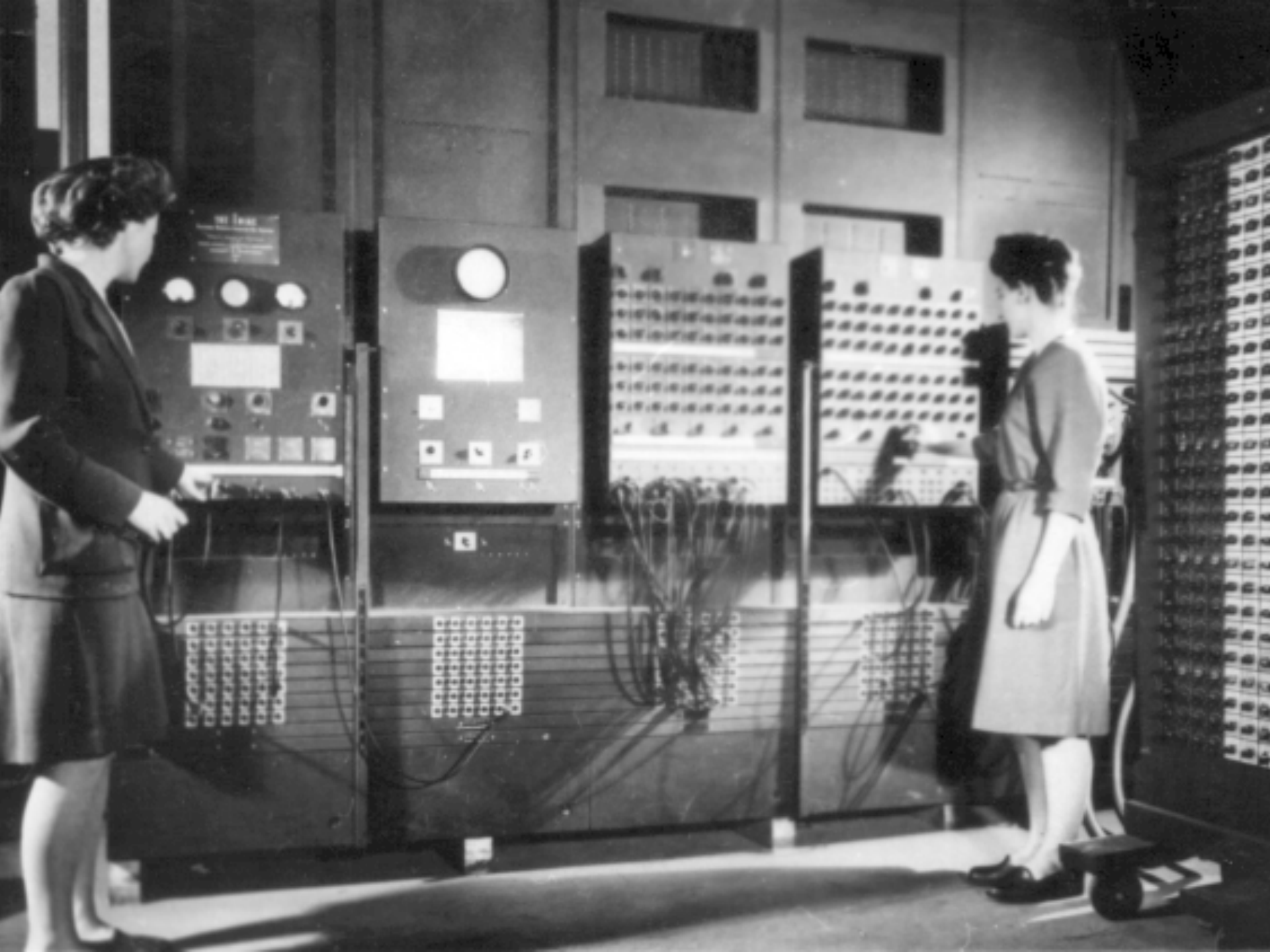
The Pioneering Years

1. ENIAC

2. Nature of programming

3. Nature of software





ENIAC in numbers

- ENIAC: Electronic Numerical Integrator And Computer
- Operation started mid-1940s
- Cost: $487k ~ $6.74m in 2016
- 5,000 additions / 333 multiplications per s.
- Weighed >30 tons
- 5m hand-soldered joints

ENIAC in numbers

- No transistors, but: 18,000 vacuum tubes
  - Vacuum tube has a lifespan of 3K hours
  - 6 fail every hour
  - Or: 1 every 10 min!
- In 1954, longest continuous period of operation at 116 hours

“Relay #70 (moth) in relay”
“First actual case of bug being found”

9/9

0800 Arctan started
1000 " stopped - arctan ✓ 1.2700 9.037 847 025
9.037 846 ?95 correct
13"UC (032) MP - MC ~~1.9821647000~~ ~~2.13047647?5 (-3)~~ 4.615925059(-2)
(033) PRO 2 2.13047647?5
correct 2.13067647?5
Relays 6-2 in 033 failed special speed test
in relay " 11.00 test.
Relays changed
1100 Started Cosine Tape (Sine check)
1525 Started Mult + Adder Test.
1545

Relay #70 Panel F
(moth) in relay.

First actual case of bug being found.

1700 1630 Arctan test started.

Relay 2145
Relay 3370

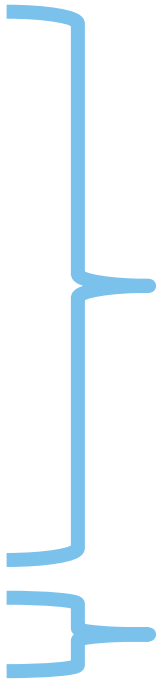
Programming? No problem!

> “It had not occurred to me that there was going to be any difficulty about getting programs working
>
> Maurice Wilkes

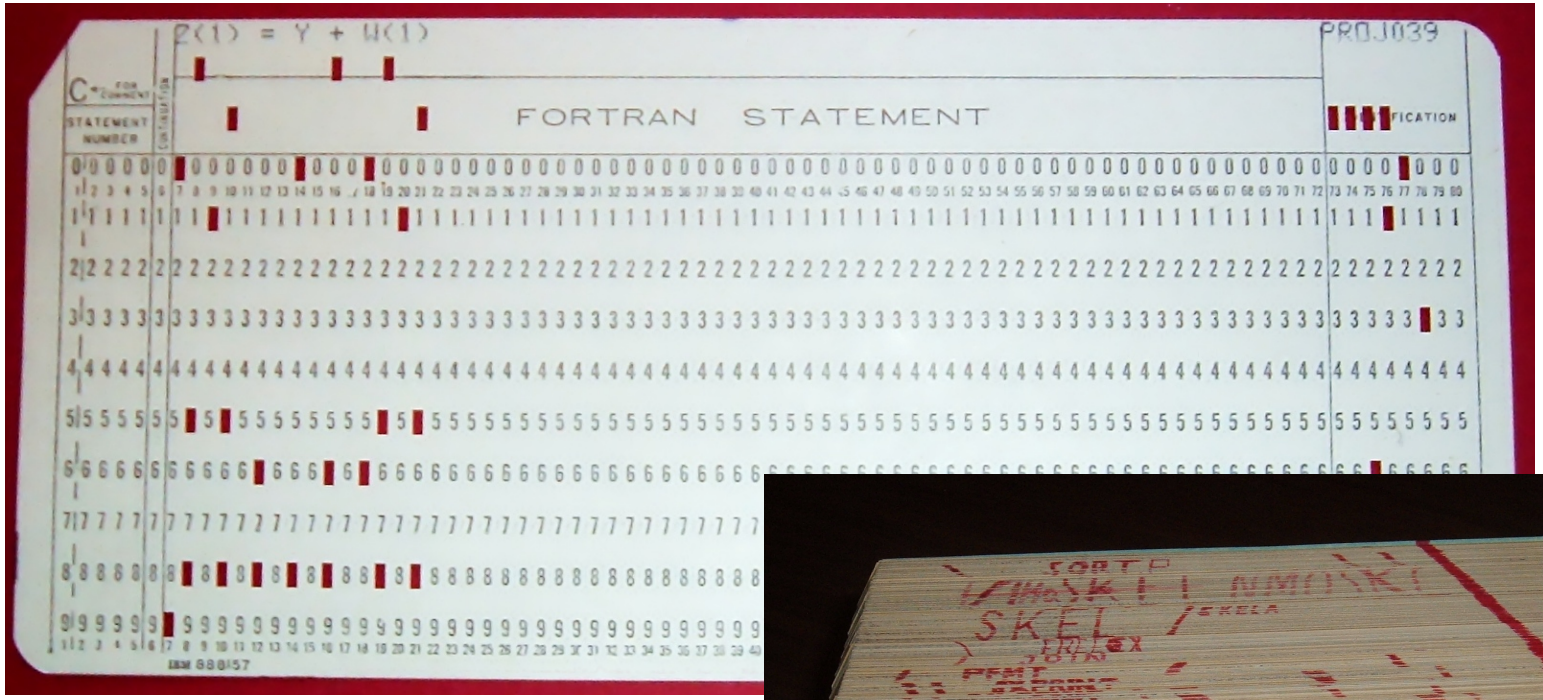
Coding as clerical work

6 steps of Programming (ca. late 1940s)

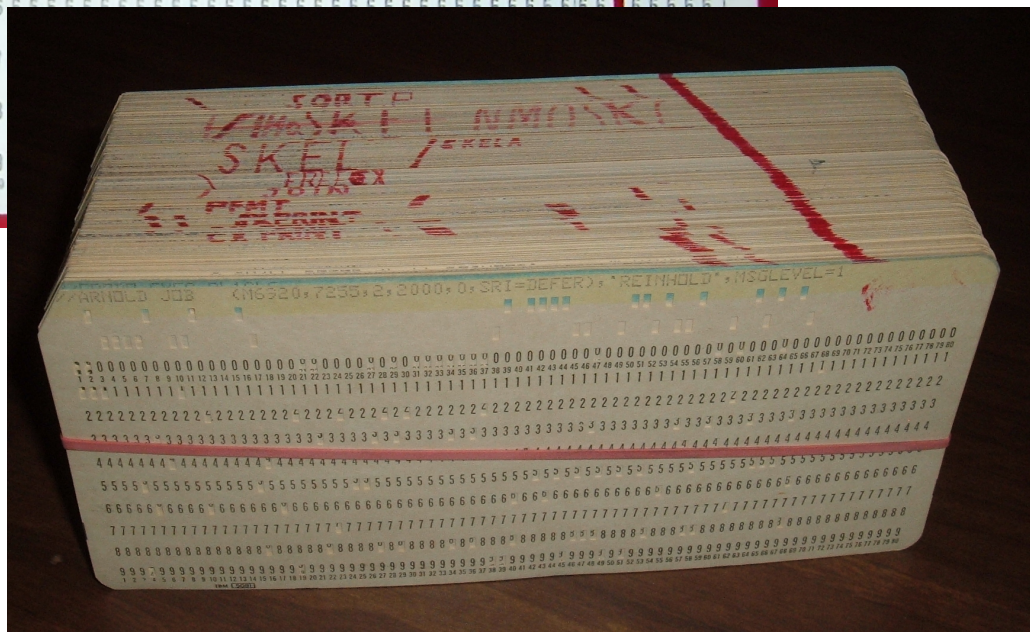
1. Conceptualize problem (math)
2. Select numerical algorithm
3. Numerical analysis to determine precision requirements
4. Determine scale factors
5. Dynamic analysis
6. Coding



Input with punch cards



A punch card: 1 line of code



A program as a deck of cards. Don't drop them!

Computing economics

1950s:

- Computer costs $600/h
- Programmer costs $2/h
- “Act accordingly”
  - Desk checking
  - Peer review
  - “Manual” execution

Focus on hardware

- In 1950s, when “computerizing” operations, focus was on hardware.
- The “Application” comprised the hardware and software together.
- Software not separately sold—no ‘software market’

What is “software” anyway?

- The term software not known until 1950s.
- Attributed to a 1958 paper by Tukey, but some claim traces back to 1953.

Use of the term ‘software’

> “Today the "software" comprising the carefully planned interpretive routines, compilers, and other aspects of automative programming are at least as important to the modern electronic calculator as its "hardware" of tubes, transistors, wires, tapes and the like.

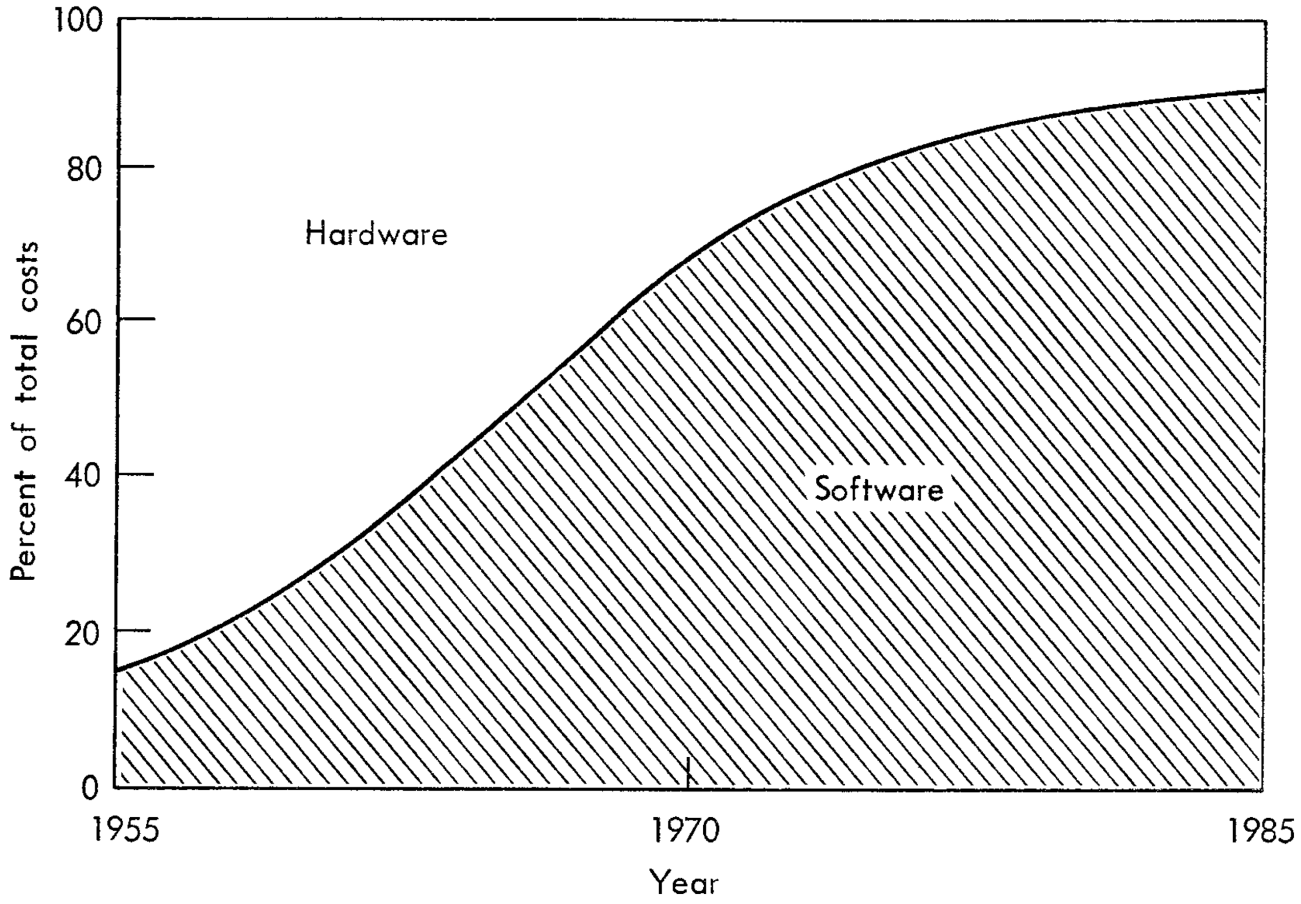
Tukey 1958

“The Teaching of Concrete Mathematics”

What is “software” anyway?

> “To each user of a computer, the total computing facility provided for his use, other than the hardware, is the software
>
> Bernard Galler
> In: Communications of the ACM 1961

Cost of “applications”



Cost of hardware vs. software

Source: Boehm ‘72

The Roots of Software Engineering

1. Computing science
2. Some large projects
3. Software crisis

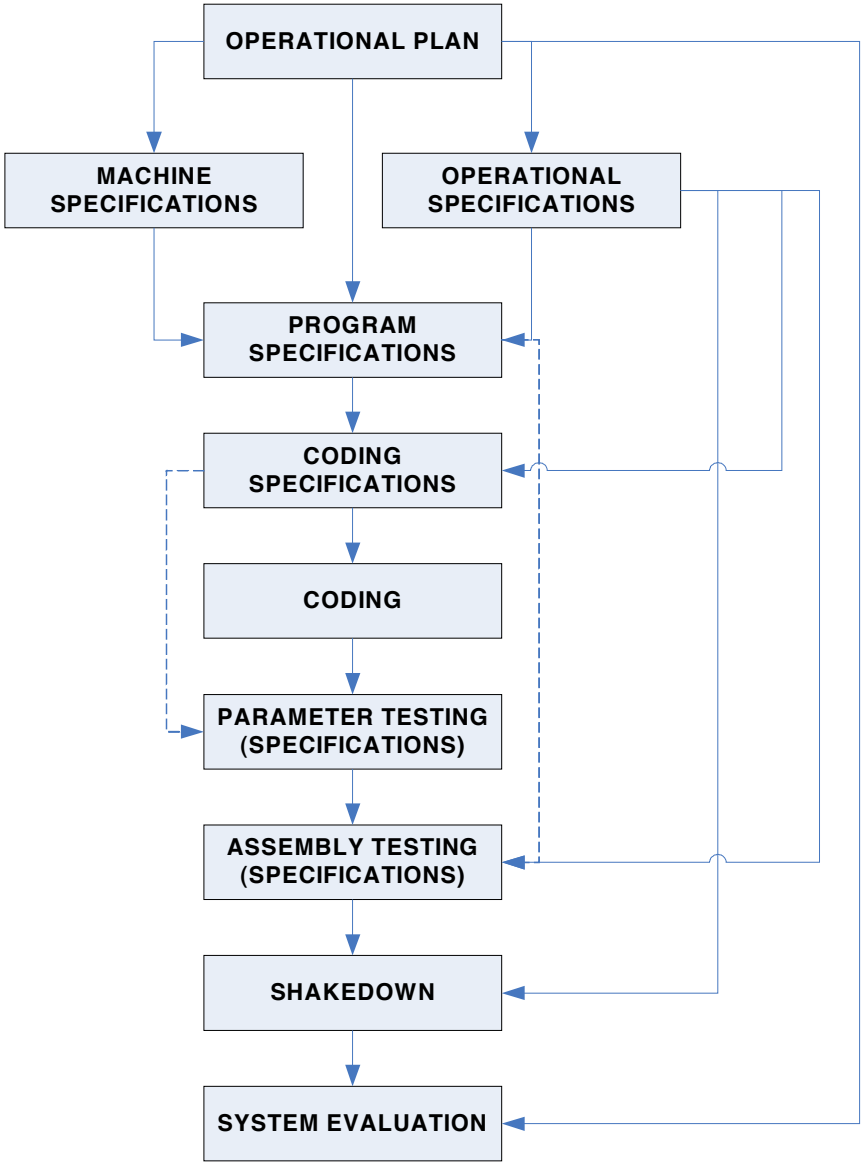
Computing science

- Early days of software development mirrored hardware development.
- Computer as powerful number cruncher —lots of math.
- Computer as a scientific instrument. Programming as incidental activity.
- Typically military and scientific applications.

Computing science

- Programmer not a recognized job category until late 1950s
- 1969 "Software unbundling" decision —start of software market
- By 1968, ca. 100,000 programmers, 175,000 2 years later

The SAGE system development plan



SAGE: Semi-Automated Ground Environment

Defense system for US and Canada

Developed by:
Software Development Corporation (SDC) (spin-off from Rand Corp.)

Cost: $8-12 billion (sources vary)
Equiv. $100bn today

Size: 1 million LOC
(largest systems to date 50K LOC)

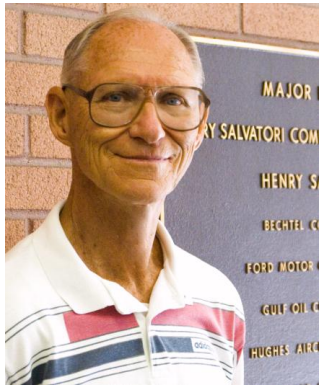
Operational from 1958-1985!

		G		*K*		
start →	0	*B*		θ	Call in *R*37	Read constants from end of program tape
	1	*F*	198	*F*		
*R*37 →	2	*T*	16	*D*	Plant $2^m h$ in $16D$	
	3	*B*	3	θ	Call in *R*37	
	4	*F*	198	*F*		
*R*37 →	5	*T*	24	*D*	Plant x_0 in $24D$	
	6	*B*	6	θ	Call in *R*37 to read *A*	
*R*37 →	7	*F*	198	*F*		
	8	*R*	1	*F*	Plant $\frac{1}{4}A$ in $22D$	
	9	*Y*		*F*		
	10	*T*	22	*D*		
15 →	11	*Z*		*F*	Wait while data tape is inserted	
39 →	12	*T*	195	*F*	Reset layout counter (i.e., clear 60th location of *P*31)	
	13	*B*	13	θ	Call in *R*37 to read y_0	Prepare for integration
	14	*F*	198	*F*		
*R*37 →	15	*G*	$11\pi\theta$		Jump to stop order if y_0 is negative	
	16	*T*	10	*D*	Place y_0 in $10D$	
	17	*A*	24	*D*	x_0 to $12D$	
	18	*T*	12	*D*		
	19	*T*	18	*D*	Clear c_0 and c_1 for integration	

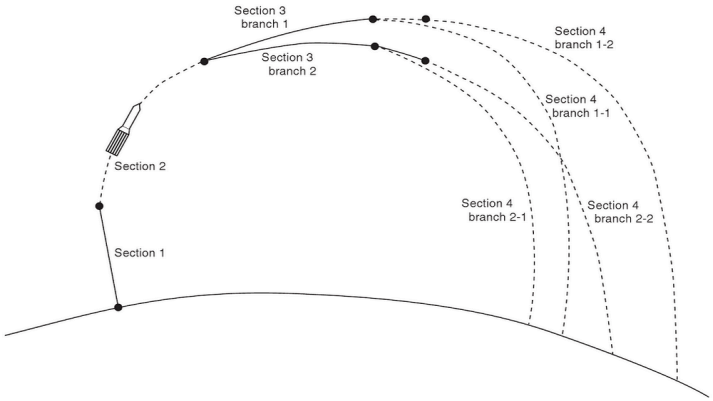
EXAMPLE

Case study: ROCKET

Rand's Omnibus Calculator of the Kinetics of Earth Trajectories



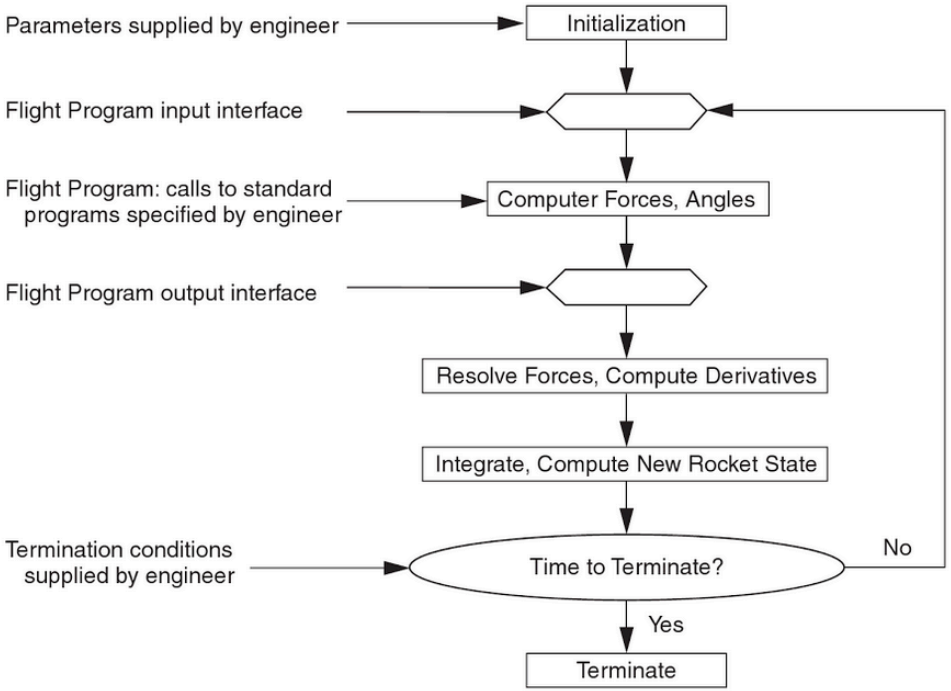
Barry Boehm



Work on rocket trajectory, 1961, Rand Corp.

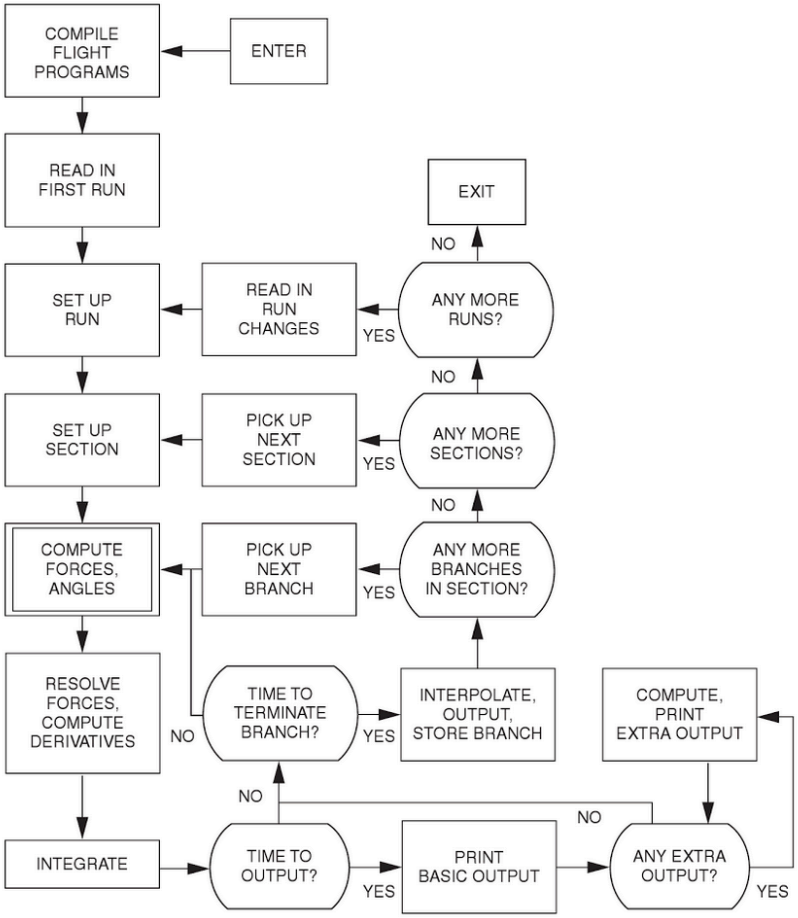
Calculate trajectories based on:

- Time
- Altitude
- Velocity
- Weight
- *etc.*



EXAMPLE

ROCKET flowchart



This flowchart represents a program with nested loops:

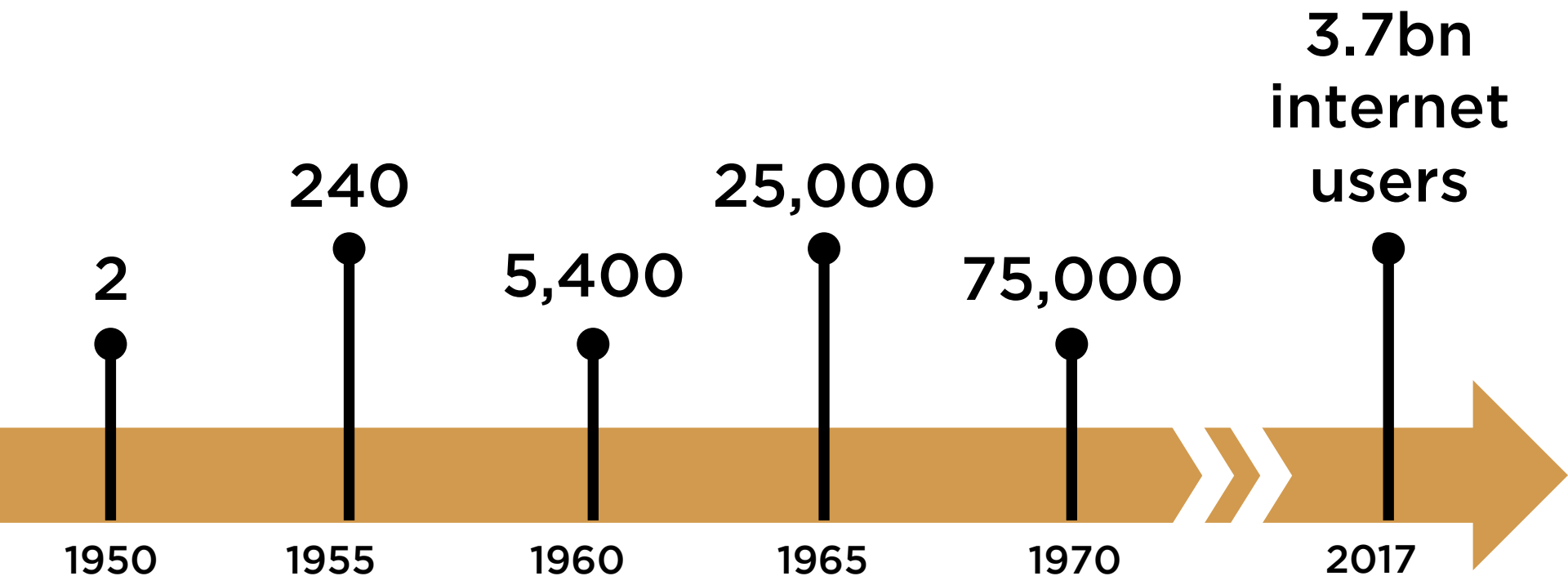
```
foreach run:
  foreach section:
    foreach branch:
      compute forces
      resolve forces
      output result
    end
  end
end
```

EXAMPLE

ROCKET

- For IBM 704 computer
- Running the SHARE operating system
- Primary languages: FORTRAN, COBOL
  - FORTRAN: FORmula TRANslator
  - COBOL: COmmon Business Oriented Language
- Users: aerospace engineers
- 6 months full-time effort:
  - 4 months architecting incl. prototypes
  - 1 month coding and integration
  - 1 month testing

Increasing demand for programmers

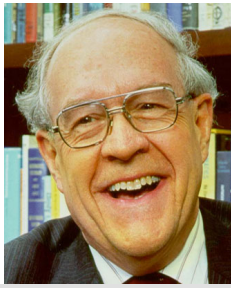


Number of computers on the market between 1950-1970

EXAMPLE

IBM System/360 and OS/360

- Family of systems, announced in 1964
- Cost $5bn
  IBM's annual revenue was $2.2bn...
- OS/360 operating system
  - Largest project to date!
  - Original budget $25m
    - Plan: 12 program designers, 60 developers
    - Reality: 1,000+ developers
- OS/360 Much-troubled project
  - Over budget
  - Over due
  - Far less capability than promised
- OS/360 project initially led by Fred Brooks
  - His lessons learned in "the Mythical Man-Month"



Fred Brooks


IBM

Programming as a gigantic problem

> As long as there were no machines, programming was no problem at all.
>
> When we had a few weak computers, programming became a mild problem;
>
> and now [that] we have gigantic computers, programming has become an equally gigantic problem.
>
> Edsger Dijkstra

Like the Wright Bros.

> We build software like the Wright brothers built airplanes:
>
> Build the whole thing, push it off a cliff, and start over again.
>
> Ronald Graham 1968

The Software Crisis

- Programming initially assumed to be “no problem”
- In reality, software systematically ended up:
  - Late
  - Incomplete / unsatisfactory
  - Over budget
- These 3 characteristics are together named the Software Crisis

The Software Crisis: other issues

- Software production by "amateurs" (even though hired as professionals!)
- Software development by tinkering or human wave ('million monkeys' approach)
- Existing software unreliable, needs permanent "maintenance" (fixing defects)
- Existing software messy, lacks transparency, prevents improvement or extension

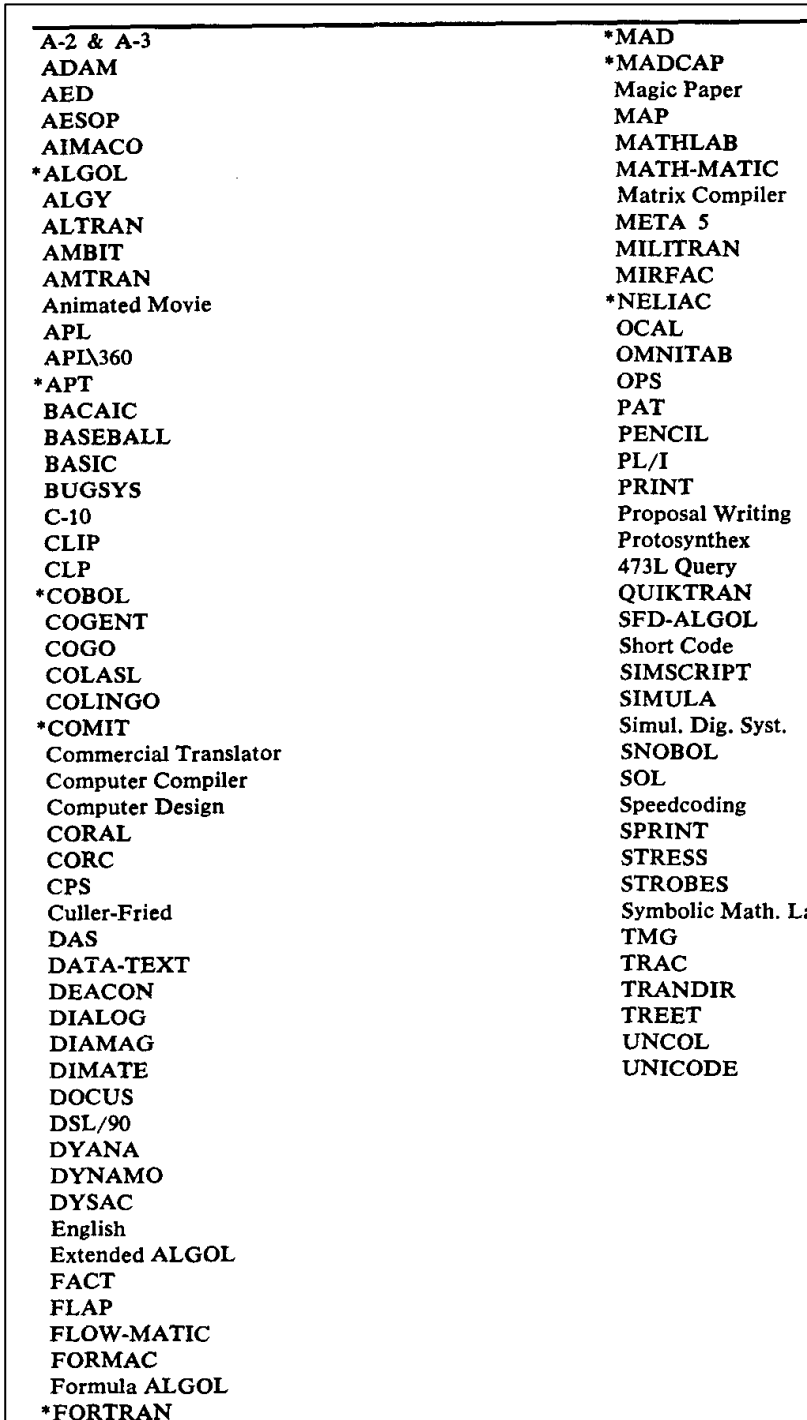
Programming too difficult

> Software engineering ... is the part of computer science that is too difficult for the computer scientists.
>
> Friedrich Bauer

Comp Sci → Soft Eng

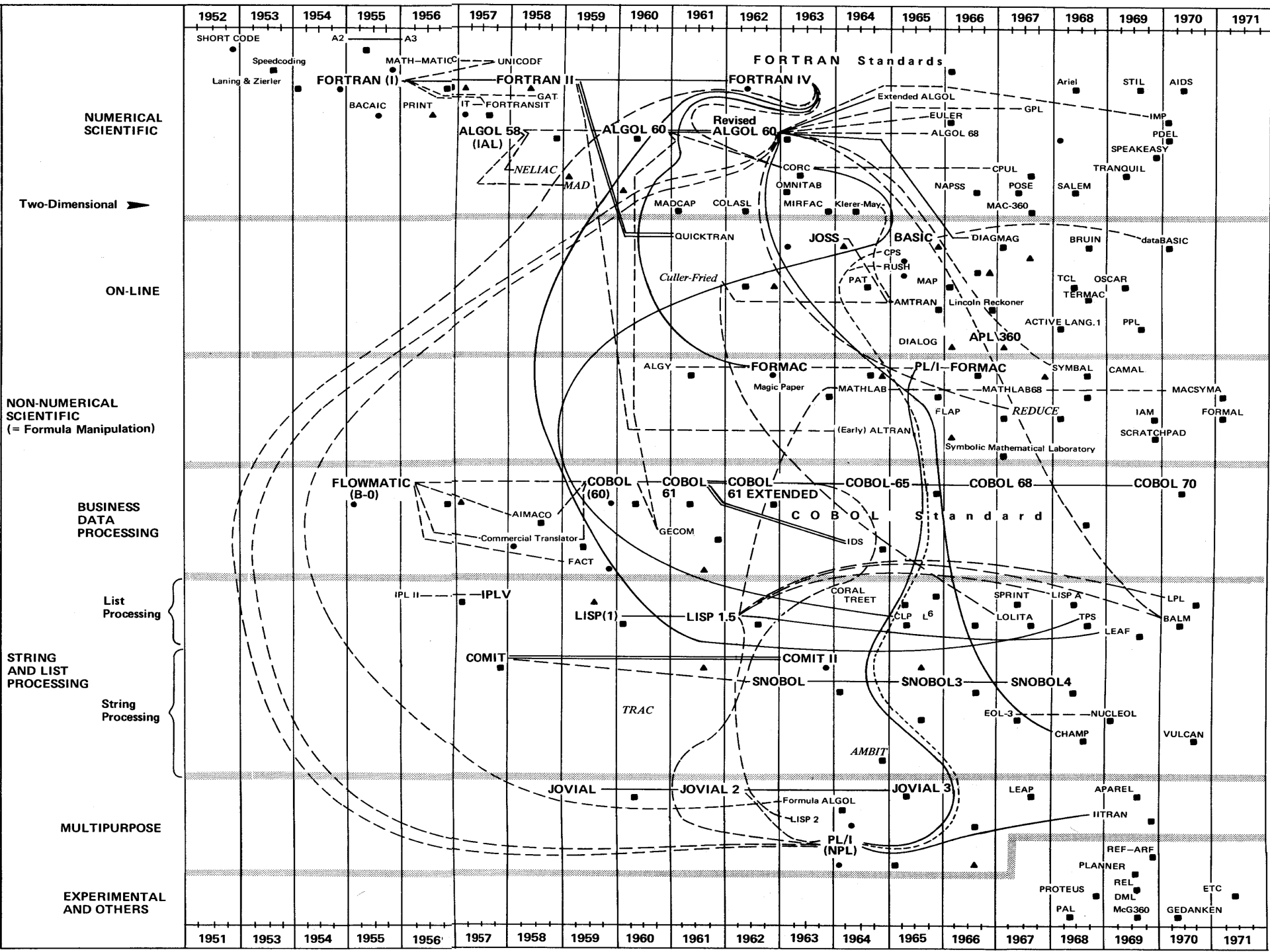
Early research in computer science focused primarily on:

- Programming languages
- Programming techniques



A-2 & A-3
ADAM
AED
AESOP
AIMACO
*ALGOL
ALGY
ALTRAN
AMBIT
AMTRAN
Animated Movie
APL
APL\360
*APT
BACAIC
BASEBALL
BASIC
BUGSYS
C-10
CLIP
CLP
*COBOL
COGENT
COGO
COLASL
COLINGO
*COMIT
Commercial Translator
Computer Compiler
Computer Design
CORAL
CORC
CPS
Culler-Fried
DAS
DATA-TEXT
DEACON
DIALOG
DIAMAG
DIMATE
DOCUS
DSL/90
DYANA
DYNAMO
DYSAC
English
Extended ALGOL
FACT
FLAP
FLOW-MATIC
FORMAC
Formula ALGOL
*FORTRAN
*MAD
*MADCAP
Magic Paper
MAP
MATHLAB
MATH-MATIC
Matrix Compiler
META 5
MILITRAN
MIRFAC
*NELIAC
OCAL
OMNITAB
OPS
PAT
PENCIL
PL/I
PRINT
Proposal Writing
Protosynthex
473L Query
QUIKTRAN
SFD-ALGOL
Short Code
SIMSCRIPT
SIMULA
Simul. Dig. Syst.
SNOBOL
SOL
Speedcoding
SPRINT
STRESS
STROBES
Symbolic Math. La
TMG
TRAC
TRANDIR
TREET
UNCOL
UNICODE

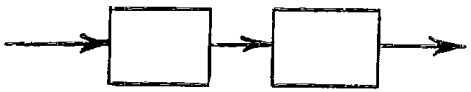
List of programming languages (fragment). *Source:* J.E. Sammet: *Programming Languages: History and Future*. Communications of the ACM, vol. 15, no. 7, 1972


1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
NUMERICAL SCIENTIFIC
Two-Dimensional
ON-LINE
NON-NUMERICAL SCIENTIFIC (= Formula Manipulation)
BUSINESS DATA PROCESSING
List Processing
STRING AND LIST PROCESSING
String Processing
MULTIPURPOSE
EXPERIMENTAL AND OTHERS
SHORT CODE
A2
A3
Speedcoding
Laning & Zierler
MATH-MATIC
UNICODE
FORTRAN (I)
FORTRAN II
FORTRAN IV
FORTRAN Standards
BACAIC
PRINT
IT
FORTRANSIT
GAT
ALGOL 58 (IAL)
ALGOL 60
Revised ALGOL 60
Extended ALGOL
GPL
EULER
ALGOL 68
NELIAC
MAD
CORC
CPUL
OMNITAB
NAPSS
POSE
SALEM
MADCAP
COLASL
MIRFAC
Klerer-May
MAC-360
QUICKTRAN
JOSS
BASIC
DIAGMAG
CPS
RUSH
PAT
MAP
Culler-Fried
AMTRAN
Lincoln Reckoner
TCL
OSCAR
TERMAC
ACTIVE LANG.1
PPL
DIALOG
APL\360
ALGY
FORMAC
PL/I-FORMAC
Magic Paper
MATHLAB
MATHLAB68
SYMBAL
CAMAL
MACSYMA
FLAP
REDUCE
IAM
FORMAL
(Early) ALTRAN
SCRATCHPAD
Symbolic Mathematical Laboratory
FLOWMATIC (B-0)
COBOL (60)
COBOL 61
COBOL 61 EXTENDED
COBOL-65
COBOL 68
COBOL 70
AIMACO
COBOL Standard
Commercial Translator
FACT
GECOM
IDS
IPL II
IPLV
CORAL
TREET
SPRINT
LISP A
LPL
LISP(1)
LISP 1.5
CLP
L6
LOLITA
TPS
BALM
LEAF
COMIT
COMIT II
SNOBOL
SNOBOL3
SNOBOL4
TRAC
EOL-3
NUCLEOL
CHAMP
VULCAN
AMBIT
JOVIAL
JOVIAL 2
JOVIAL 3
LEAP
APAREL
Formula ALGOL
LISP 2
IITRAN
PL/I (NPL)
REF-ARF
PLANNER
REL
PROTEUS
DML
ETC
PAL
McG360
GEDANKEN
Ariel
STIL
AIDS
IMP
PDEL
SPEAKEASY
TRANQUIL
dataBASIC

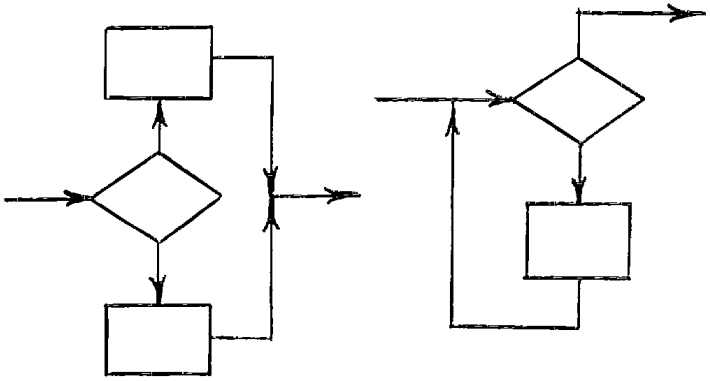
Programming techniques

- Study topic: Structured Programming
- Dijkstra's letter to the editor:
  "GOTO considered harmful" in Comm. ACM '68

Fig. 1. Jacopini's Base Diagrams



(a) Π (concatenation)



(b) Δ (selection) (c) Ω (repétition)

D-charts / D-structures:

Any program can be converted to an equivalent program using only 3 structures:

1. Concatenation / Sequential
2. Selection / Alternation
3. Repetition / Iteration

Comp Sci → Soft Eng

NATO Conference on Software Engineering

(report freely available online)

- Held in 1968 in Garmisch, Germany
- Considered first conference on SE
- Term Software Crisis coined

The roots of software engineering

Term Software Engineering
coined provocatively

> The whole trouble comes from the fact that there is so much tinkering with software.
>
> It is not made in a clean fabrication process, which it should be.
>
> What we need, is software engineering

Defining Software Engineering

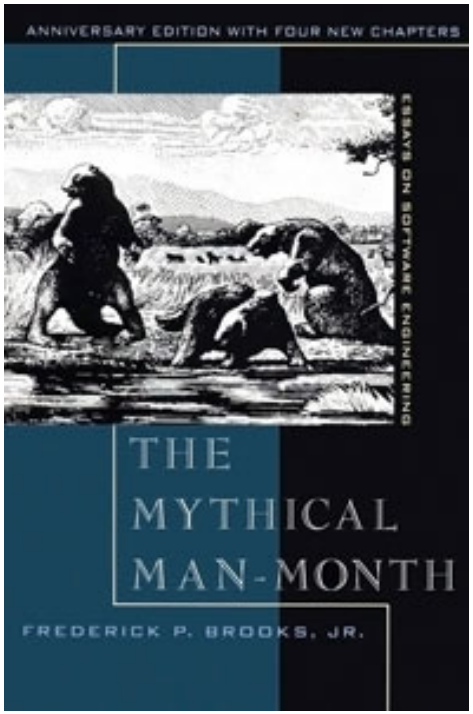
> *Practical application of scientific, engineering, and management skills to deliver a software system within budget in a timely manner that satisfies a user's needs.*

> *The establishment and use of sound engineering principles (methods) in order to obtain economically software that is reliable and works on real machines.*

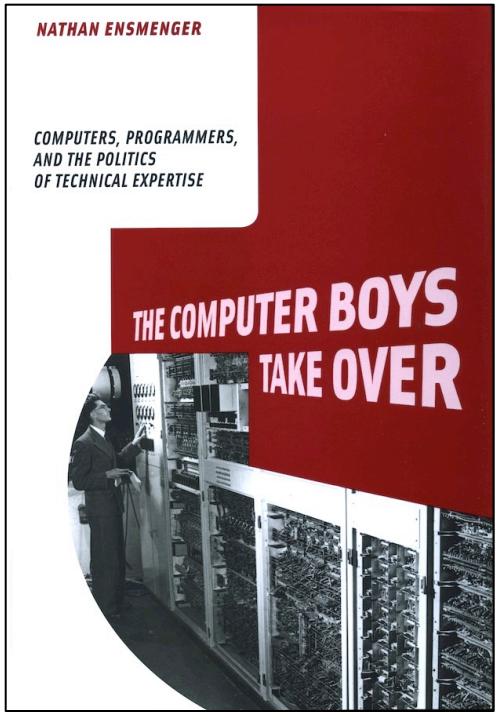
Summary

- Early days of computing hardware-focused:
  - Cost of hardware very high
  - Programming not considered a problem
  - Programming approached as hardware
- Term ‘software’ not used until late 1950s
- Term ‘Software Engineering’ coined provocatively
- Software Crisis: continuous struggle to deliver a satisfactory product in time within budget

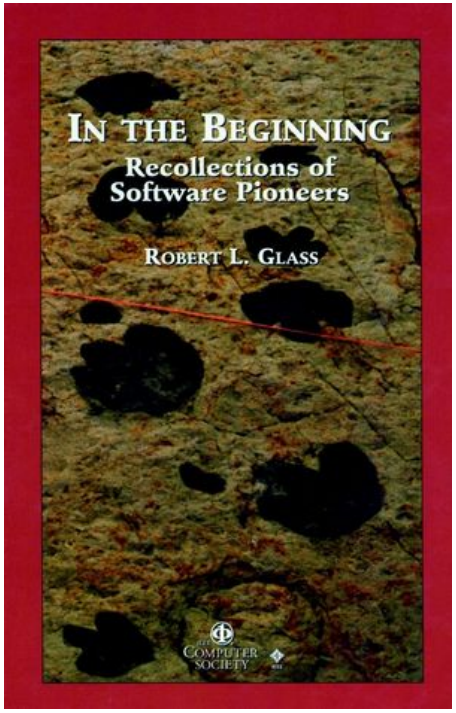
Recommended reading



Frederick Brooks, "The Mythical Man-Month"



Nathan Ensmenger, "The Computer Boys Take Over"



Robert Glass, "In the beginning: Recollections of Software Pioneers"

Thank you for your attention

Questions & suggestions can be sent to:
k.stol@cs.ucc.ie