The future of Programming
Looking back to look ahead

1940s

PROGRAMMING

20xx
ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

By A. M. Turing.

[Received 28 May, 1936.—Read 12 November, 1936.]
1939: The Bombe
1942 Collosus
1945: Mercury Delay Lines
1945: CRT Memory
1945

Turing writes code.
* In binary. (base 32 reversed)
* Uses integer ‘add’ and logical ‘not’.
* Invents subroutine
* Invents and codes stack
* Invents floating point
* Etc.
“We shall need a great number of mathematicians of ability” because “there will probably be a good deal of work of this kind to be done.”
“One of our difficulties will be the maintenance of an appropriate discipline, so that we do not lose track of what we are doing.”
Mathematicians of Ability.
A Great Number.
Appropriate Discipline.
1945

Number of computers in the world: $O(1)$
1945

Number of programmers in the world: $O(1)$
1950s
1953 Fortran

C AREA OF A TRIANGLE WITH A STANDARD SQUARE ROOT FUNCTION
C INPUT - TAPE READER UNIT 5, INTEGER INPUT
C OUTPUT - LINE PRINTER UNIT 6, REAL OUTPUT
C INPUT ERROR DISPLAY ERROR OUTPUT CODE 1 IN JOB CONTROL LISTING
READ INPUT TAPE 5, 501, IA, IB, IC
501 FORMAT (3I5)
C IA, IB, AND IC MAY NOT BE NEGATIVE
C FURTHERMORE, THE SUM OF TWO SIDES OF A TRIANGLE
C IS GREATER THAN THE THIRD SIDE, SO WE CHECK FOR THAT, TOO
  IF (IA) 777, 777, 701
  701 IF (IB) 777, 777, 702
  702 IF (IC) 777, 777, 703
  703 IF (IA+IB-IC) 777,777,704
  704 IF (IA+IC-IB) 777,777,705
  705 IF (IB+IC-IA) 777,777,799
  777 STOP 1
C USING HERON'S FORMULA WE CALCULATE THE
C AREA OF THE TRIANGLE
799  S = FLOATF (IA + IB + IC) / 2.0
     AREA = SQRT( S * (S - FLOATF(IA)) * (S -
               FLOATF(IA)) * (S - FLOATF(IA)) *
               (S - FLOATF(IA))
WRITE OUTPUT TAPE 6, 601, IA, IB, IC, AREA
601 FORMAT (4H A= ,I5,5H B= ,I5,5H C= ,I5,8H AREA= ,F10.2,
+     13H SQUARE UNITS)
STOP
END
1958 - LISP

Functional Programming

DEFINE
(((RVRSE, (LAMBDA, (L)), (COND, ((NULL, L), NIL)),
    (T, (CONS, (RVRSE, (CDR, L)), (CONS, (CAR, L), NIL))))),
(RVDE, (LAMBDA, (L), (REV, L, NIL))),
(REV, (LAMBDA, (J, K), (COND, ((NULL, J), K),
    (T, (REV, (CDR, J), (CONS, (CAR, J), K))))))))
()
RVRSE  ((A, B, C, D, E))  ()
RVDE   ((A, B, C, D, E))  ()
1954-1960
IBM sold 140 model 70x computers.
Fortran Lisp.
1960

Number of computers in the world: \( O(1E2) \)
1960

Number of programmers in the world: $O(1E3)$
1950s

• Programmers were drawn from:
  • Engineers
  • Scientists
  • Mathematicians
1965

10,000 1401s.
Rented for $2,500/mo ($20K today).

This put the 1401 in reach of many businesses who eagerly bought them.
1965

Number of computers in the world: $O(1E4)$
Number of programmers in the world: $O(1E5)$
Looking back to look ahead

1945: 1

Programmers

I am 13

1965: 100,000

I am 13
1965

- Not Enough:
  - Engineers, Scientists, Mathematicians
- No CS Grads.
- Programmers drawn from:
  - Best and brightest
  - Accountants, Planners, etc.
Though not mathematicians, they were: Experienced, Disciplined, Professionals. Turing would likely have approved.
1966

1,000 360s every month.
1966 - Simula-67

Ole-Johan Dahl

Kristen Nygard

Object Orientation
Go To Statement Considered Harmful

Key Words and Phrases: go to statement, jump instruction, branch instruction, conditional clause, alternative clause, repetitive clause, program intelligibility, program sequencing

CR Categories: 4.22, 5.23, 5.24

EDITOR:

For a number of years I have been familiar with the observation that the quality of programmers is a decreasing function of the density of go to statements in the programs they produce. More recently I discovered why the use of the go to statement has such disastrous effects, and I became convinced that the go to statement should be abolished from all “higher level” programming languages (i.e. everything except, perhaps, plain machine code). At that time I did not attach too much importance to this discovery; I now submit my considerations for publication because...
1968 - C

Ken Thompson
Dennis Ritchie
1970

50,000 PDP8s
1970

Number of computers in the world:

\(O(1E5)\)
1970

Number of programmers in the world: $O(1E6)$
Looking back to look ahead

1945: 1

Programmers

1970: 1,000,000
1970

Hundreds of thousands of computers.

A million programmers.

Who were they?
1970

Tens of thousands of new CS & EE Grads.

They all had something in common.

We were all young.

We were almost all male.
At my first job there were ~24 programmers.

Most were in their thirties or forties;

and half were women.
1980

My employer had ~50 programmers. All in their twenties or early thirties. Two were women.
The demographics of programming had decisively shifted towards young men.

Hundreds of thousands of very young men. typically not what Turing would have thought of as “disciplined mathematicians”.

Source: National Science Foundation, American Bar Association, American Association of Medical Colleges
Credit: Quoctrung Bui/NPR
But businesses had to have programmers.

What very young men lack in discipline, they make up for in energy.
And they’re cheap.
Now remember:

• Up to now programmers were disciplined professionals.

• They didn’t need a lot of management or process.

• They knew how to manage their time, communicate, and work together.

• They understood deadlines & commitments. What to leave in and what to leave out.
Those Disciplined Professionals had worked miracles

- IBM 360 Virtual Memory OS.
- NASA: Mercury, Gemini, Apollo
- Structured, Functional, Object-Oriented.
- Fortran, Cobol, Algol, Lisp, C, Unix.
Those original programmers knew how to get big things done.
Agile:
The process used by disciplined professionals observed in the wild.
But hoards of young testosterone driven men need discipline imposed upon them from above.

They need close management and a fixed process.

They need:

Waterfall.
Looking back to look ahead

1945: None

Process

1970: Waterfall
Looking back to look ahead

The Waterfall era

1970

2000
1970-1995

Number of programmers doubles every five years.

Tens of millions of programmers.

Half of whom have less than five years experience.
1995

- Original cohort of disciplined professionals retires.
- The first wave of career programmers comes of age (well over 40).
- We foresee the need for change.
1995

- Scrum: Schwaber, Beedle, DeVos, et. al.
- XP: Beck, Cunningham
- FDD: Coad.
- Crystal: Cockburn
2001: Snowbird
Agile Manifesto
Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan

That is, while there is value in the items on the right, we value the items on the left more.
Agile requires Discipline

- Working in fixed time boxes.
- Estimating in relative units.
- Customer communication.
- Continuous integration.
- Collaboration.
- And so much more.
Extreme Programming

• The most technical disciplines
  • TDD
  • Refactoring
  • Simple Design
  • Acceptance Tests
  • Metaphor
Many of us felt that these technical disciplines were the glue that made the whole agile process work properly.

Without those technical - dare I say mathematical disciplines, the code being produced would grow and evolve in ways that made it harder and harder to work with.

“lose track of what we are doing”
Business understands Discipline
Rapid Adoption of Scrum
But Business doesn’t understand us.

- So they can’t evaluate, endorse, or approve any technical practices.
Worse, programers do not agree.
Flaccid Scrum
An efficient business discipline coupled to an undisciplined engineering team will very rapidly make a mess.
Agile Split

Agile: Business Practices

Craftsmanship: Technical Practices
Kent Beck at Snowbird stated a goal for Agile: Healing of the divide between business and programming.
Agile Split

FAIL!

Agile: Business Practices

Craftsmanship: Technical Practices
It’s time for us to:

- Grow Up.
- Define our profession.
- Choose our practices and disciplines.
- Reunify Agile/Craftsmanship.
Because we are headed towards a disaster
Civilization depends upon us.
in ways it doesn’t yet understand.
In ways we don’t yet understand.
WE RULE THE WORLD
TENS OF THOUSANDS DIE IN SOFTWARE CATASTROPHE
HOW COULD YOU...

...HAVE LET THIS HAPPEN?
CONGRESS PASSES SWEEPING SOFTWARE REGULATIONS
The Future of Programming

1940s

20xx
UNCLE BOB

unclebob@cleancoder.com

cleancoder.com

cleancoders.com - Videos

@unclebobmartin
The Professional’s Oath
I will not produce harmful code.
II

The code that I produce will always be my best work.

I will not knowingly release code that is defective either in behavior or structure.
I will provide, with each release, a quick, sure, and repeatable proof that every element of the code works.
I will make frequent, small, releases. I will not impede progress.
I will fearlessly and relentlessly improve the code at every opportunity.

I will never make the code worse.
VI

I will keep productivity high.
I will do nothing that decreases productivity.
I will continuously ensure that others can cover for me, and that I can cover for them.
VIII

I will produce estimates that are honest both in magnitude and precision.

I will not make promises without certainty.
IX

I will never stop learning and improving my craft.