I've tried to get dates right, but I do not claim to be giving the first published or operational occurrence of a big idea. Don't cite my dates.
The pictures do not represent what we ‘60’s & 70’s innovators look like today.
Uphill Both Ways '44-'51

- Hand-coding on paper
- In binary, octal for short
- Octal op codes
- Octal addresses
- Absolute addresses
- Transfer to paper tape, cards, punched-film
A Slow Insight

A Program

Programming System

Program Product

Programming System Product

The Mythical Man-Month
1. Program 1843

- Ada Lovelace
- Charles Babbage’s Analytical Engine
2. Software Product '51

- *Software Product* = A program to be used by other than the author
  - Generalization
  - Testing
  - Documentation
  - On-going maintenance
3. Software System

- **Software System** = A system of many separate programs working together
  - Interfaces
  - System integration and test
GM-NAA I-O (Batch) Operating System ’56

- Robert Patrick (GM), Owen Mock (NAA), George Ryckman (GM)
- **Components**
  - Input translator for cards, tape, multiple languages
  - SHARE Assembly program, later FORTRAN
  - Compute monitor (abort on errors, dump memory)
  - Accounting package
  - Output converter (to decimal) tape. (Hand-carried off)
- 10-fold improvement in jobs/hour
- Shorter turn-around on average
- Professional operators; programmers programmed
- No idle machine time
- 40 copies distributed; no support
A software system designed to be distributed to (and supported for) many users

- **SHARE Operating System '59**
  - Based on GM-NAA I-O Operating System
  - Distributed by IBM
  - Maintained by IBM
Software Engineering

The discipline of making software products

Name coined by Margaret Hamilton ~'66
Big Ideas of the '40's

• Programmable computers
  • Babbage 1843
  • Zuse '41
  • Aiken '44
  • Kilburn '48
  • Wilkes '49

• Stored Program—von Neumann
Stored Program

Conceived it
John von Neumann '46

Built it
Tom Kilburn '48, Maurice Wilkes '49
Big Ideas of '50's — 1

• Closed Subroutines
  • 'Wheeler Jump' calling sequence
• Input-Output Libraries
• Symbolic Assembler

• Sir Maurice Wilkes, F.R.S.
• David Wheeler, F.R.S.
• Stanley Gill

University of Cambridge
The Most Important Book in the History of Software '51

THE PREPARATION OF PROGRAMS FOR AN ELECTRONIC DIGITAL COMPUTER

by
MAURICE V. WILKES, F.R.S.
DAVID J. WHEELER
and
STANLEY GILL
SECOND EDITION
Big Ideas of '50's — 2

- Compilers
- Operating Systems
- Terminals and Communications
- Graphical Displays
- Block-structured programming
  - Algol '58
Compilers

FORTRAN '56
Optimized Run-time

FLOW-MATIC '59
An English-like Language

John Backus

RDML Grace Murray Hopper
Terminals, Communications, Graphical Displays '53

MIT Whirlwind I '51

Cape Cod prototype air defense system '53

SAGE Air Defense System '58
Big Ideas of the ‘60’s — 1

• Software engineering as engineering
  • NATO Conferences '68, '69

• Multiprogramming

• Time-sharing
The NATO Conferences ‘68,’69

- Fritz Bauer idea
- “Provocative name”
- ‘68 Conference
  - Enthusiastic participants
  - About software crisis
  - Enphasized management
- ‘69 Conference
  - Aimed to be more technical
  - Was much more fractious
Stretch Multiprogramming OS '58-9
Ted Codd

- Designed for efficient compute-I/O overlap, not yet time-sharing

- Enabled by STRETCH supervisory hardware:
  - Interruption
  - Clock
  - Memory protection
  - Privileged ops
Time Sharing OS’s

**ATLAS Supervisor ’62**
Tom Kilburn
- On Manchester ATLAS
- First memory paging OS
- Enabled interactive debugging

**MIT CTSS ’62**
Fernando Corbato
- On IBM 7090
- Precursor of MULTICS
  - On GE 645
  - Many important ideas
Big Ideas of the ‘60’s — 2

• Classes, inheritance
• Database systems
• Proofs and Axioms
Classes, Inheritance, Object-Orientation. '67

Simula 67

Kristen Nygaard

Ole-Johan Dahl
Database Systems '65

Integrated Data Store '65
Charles Bachman
Navigational DB Model

ORACLE '79
Ted Codd
Relational Model '70
Proofs and Axioms '67

Robert Floyd '67
"Assigning Meanings to Programs"

Sir Anthony Hoare '69
"An Axiomatic Basis for Computer Programming"
Big Ideas of the ‘70’s

- Information hiding, modules, abstract data types
- Top-down, incremental build, stepwise refinement
- Inspections
- Software engineering management
Information Hiding, Modules, Abstract Data Types

David Parnas '71
- Information hiding
- Modules

Barbara Liskov '74
- Abstract data types
Top-Down, Incremental Build, Stepwise Refinement '71

Harlan Mills '71

Niklaus Wirth '71
Inspections

A formal process
Outside team of several inspectors
Code reading line-by-line
Work against a set of specific requirements
Inspectors watch for defects and requirement failures

Michael E. Fagan
Software Engineering Management

• Several early papers

• The Mythical Man-Month

• Requirements verifying and validating
Requirements Verifying and Validating ‘79

- Verification: "Am I getting the requirements right?"
  - Completeness,
  - Consistency
  - Feasibility: Cost, Schedule
  - Testability

- Validation: "Am I building the right product?"

Barry Boehm

FPB 5/30/18
“No Silver Bullet” ’85, Refired

- Software is *Essentially* hard to build
  - Complexity is inherent
  - Conformity to hardware and world
  - Changeability (looks easy to change)
  - Invisibility

- “There is no single development in technology or management which alone promises a 10X gain in 10 years” is again true 30 years later
Resources

• Grady Booch ACM Webinar

• [https://learning.acm.org/webinars](https://learning.acm.org/webinars) “History of Software Engineering”

• Selby, *Software Engineering: Barry W. Boehm's Lifetime Contributions to Software Development, Management, and Research*

• Hoffman & Weiss, *Software Fundamentals: Collected Papers of David L. Parnas*