ACM Highlights

- Learning Center tools for professional development: http://learning.acm.org
  - **Safari Learning Platform**
    - 50,000+ trusted technical books, video courses, and O'Reilly conference videos
    - Hundreds of learning paths, online learning tools, case studies
  - **Skillsoft Learning Collections**
    - 1,800+ Skillsoft courses, virtual labs, test preps, live mentoring for software professionals covering programming, data management, DevOps, cybersecurity, networking, project management, more
    - 4,800+ 30,000+ task-based short videos for "just-in-time" learning
    - Training toward top vendor certifications (CEH, Cisco, CISSP, CompTIA, ITIL, PMI, etc.)
  - 1,200+ DRM—free books in CS on the **ScienceDirect** platform (including Morgan Kaufmann and Syngress titles)
  - Learning Webinars from thought leaders and top practitioners
  - Podcast interviews with innovators, entrepreneurs, and award winners
- Popular publications:
  - Flagship *Communications of the ACM (CACM)* magazine: http://cacm.acm.org/
  - *ACM Queue* magazine for practitioners: http://queue.acm.org/
- ACM Digital Library, the world's most comprehensive database of computing literature: http://dl.acm.org.
- International conferences that draw leading experts on a broad spectrum of computing topics: http://www.acm.org/conferences.
- Prestigious awards, including the ACM A.M. Turing and ACM Prize in Computing: http://awards.acm.org
- And much more… http://www.acm.org.

The History of Software Engineering

Grady Booch

IBM Fellow & Chief Scientist for Software Engineering

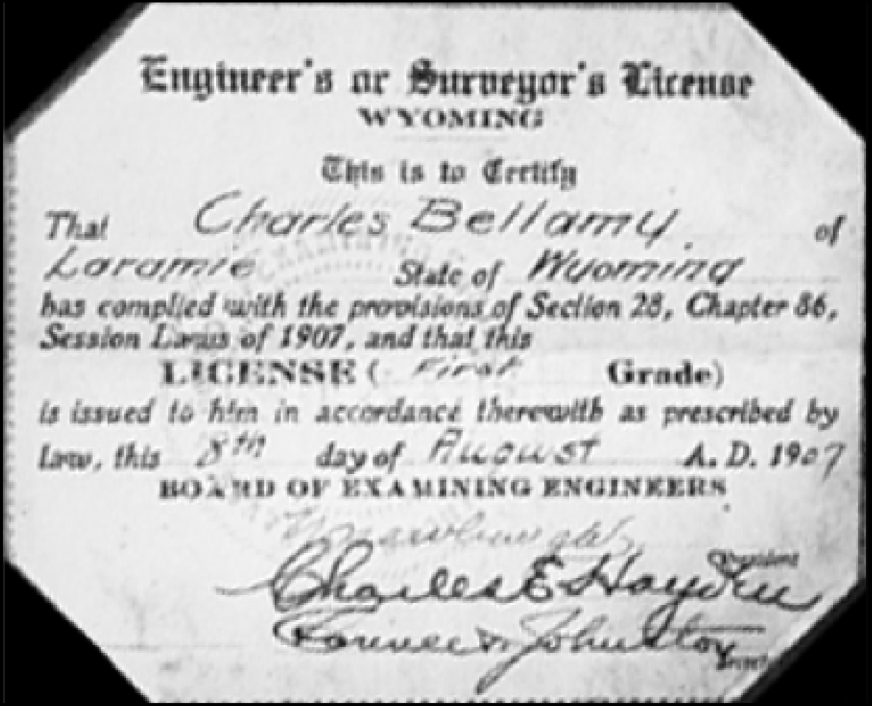
Email: gbooch@us.ibm.com
Twitter: @grady_booch
Web: computingthehumanexperience.com

V1.0

Imhotep is considered the first engineer; he lived in Egypt around the 27th century BCE, and served as the chancellor to the pharaoh Djoser, architect of the step pyramid, and high priest of the sun god Ra.

In the 19th century BCE, the Code of Hammurabi had this to say: *If a builder erect a house or a man and do not make its construction firm, and the house on which he built collapse and cause the death of the owner of the house, that builder shall be put to death.*

Ismail al-Jazari is another candidate for consideration as the first engineer; he lived in Turkey around the 12th century CE, during the Islamic Golden Age. Author of *The Book of Knowledge of Ingenious Mechanical Devices*, he is also considered the father of robotics.



Engineer's or Surveyor's License
WYOMING

This is to Certify

That Charles Bellamy of Laramie State of Wyoming has complied with the provisions of Section 28, Chapter 86, Session Laws of 1907, and that this

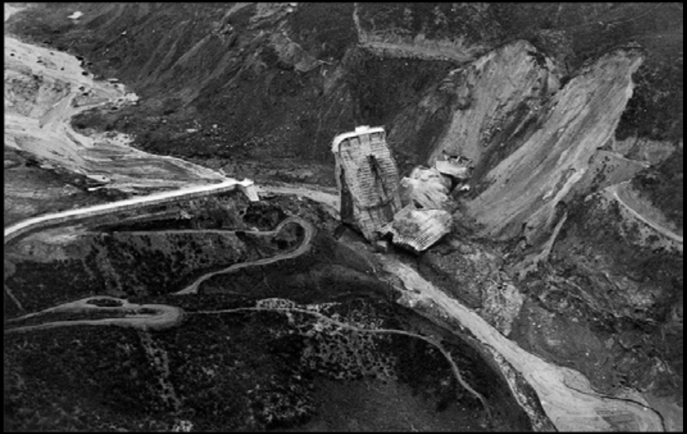
LICENSE (First **Grade)**

is issued to him in accordance therewith as prescribed by Law, this 8th day of August A. D. 1907

BOARD OF EXAMINING ENGINEERS

Charles E Hayden President

Clarence T. Johnston Secy.



The term *systems engineering* dates back to Bell Telephone Laboratories in the early 1940s, with major applications of systems engineering during World War II.

INCOSE

International Council on Systems Engineering ®

Worldwide, engineering is largely an occupational closure, requiring graduation from an accredited college or university, the passing of a standard examination, and experience working as an apprentice under other licensed engineers.

Annie Cannon

The first computers were human (and, for the most part, women).

George Stibitz

A pioneer in Boolean logic circuits, Stibitz coined the term *digital* around 1942.

John Tukey

Co-inventor of the Fast-Fourier Transform algorithm, Tukey coined the term *software* in 1952.

Prompted by the so-called software crisis - marked by the rapid rise of computational power together with the growing complexity of problems to be addressed - NATO held a Software Engineering Conference in 1968 and again in 1969. Bauer proposed the term *software engineering* to mean the “establishment and use of sound engineering principles to economically obtain software that is reliable and works on real machines efficiently.”

Frederich Bauer

SOFTWARE ENGINEERING

Report on a conference sponsored by the
NATO SCIENCE COMMITTEE
Garmisch, Germany, 7th to 11th October 1968

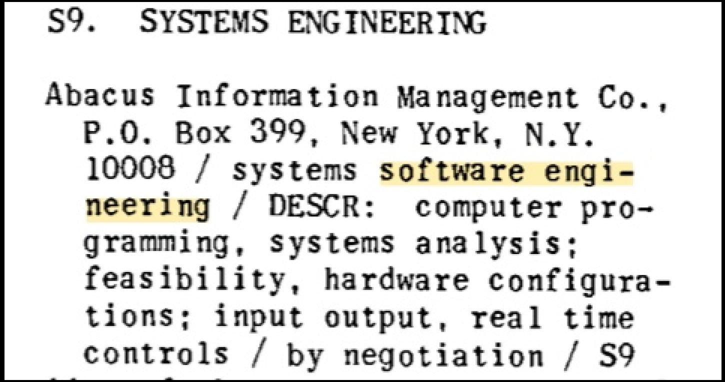
Chairman: Professor Dr. F. L. Bauer
Co-chairmen: Professor L. Bolliet, Dr. H. J. Helms

Editors: Peter Naur and Brian Randell

January 1969

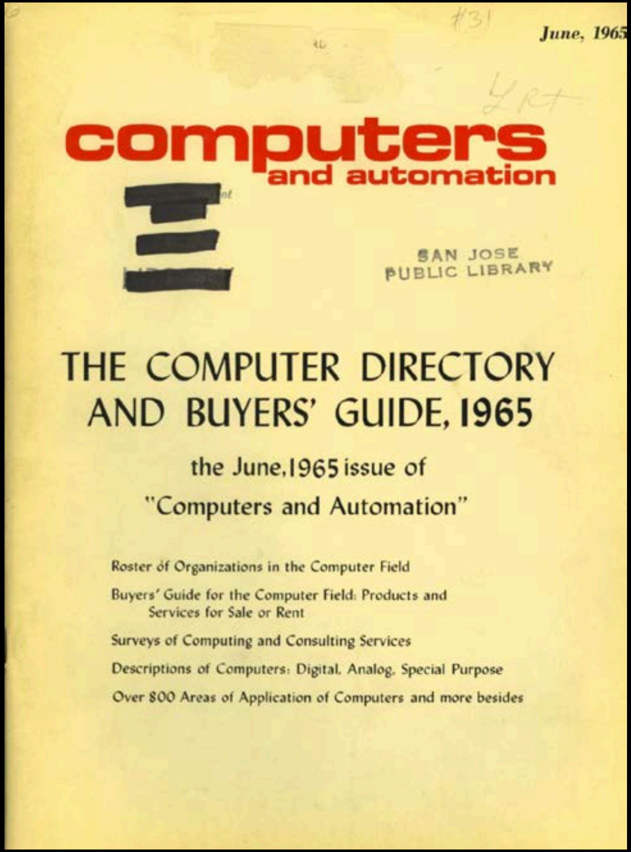
Anthony Oettinger

In the August 1966 issue of *Communications of the ACM*, Oettinger had this to say: “A concern with the *science* of computing and information processing, while undeniably of the utmost importance and an historic root of our organization is, alone, too exclusive. We must recognize ourselves as members of an *engineering* profession, be it hardware engineering or software engineering, a profession without artificial and irrelevant boundaries like that between ‘scientific’ and ‘business’ applications.”



S9. SYSTEMS ENGINEERING

Abacus Information Management Co.,
P.O. Box 399, New York, N.Y.
10008 / systems software engineering / DESCR: computer programming, systems analysis; feasibility, hardware configurations; input output, real time controls / by negotiation / S9



#31

June, 1965

Y R+

computers
and automation

[redacted]

THE COMPUTER DIRECTORY
AND BUYERS' GUIDE, 1965

the June, 1965 issue of
"Computers and Automation"

Roster of Organizations in the Computer Field

Buyers' Guide for the Computer Field: Products and Services for Sale or Rent

Surveys of Computing and Consulting Services

Descriptions of Computers: Digital, Analog, Special Purpose

Over 800 Areas of Application of Computers and more besides

Margaret Hamilton

First a developer for SAGE and then the lead developer for the Skylab and Apollo flight software, Hamilton coined the term *software engineering* around 1963 or 1964 while working at the Charles Stark Draper Laboratory at MIT.

Grace Hopper

“To me programming is more than an important practical art. It is also a gigantic undertaking in the foundations of knowledge.”

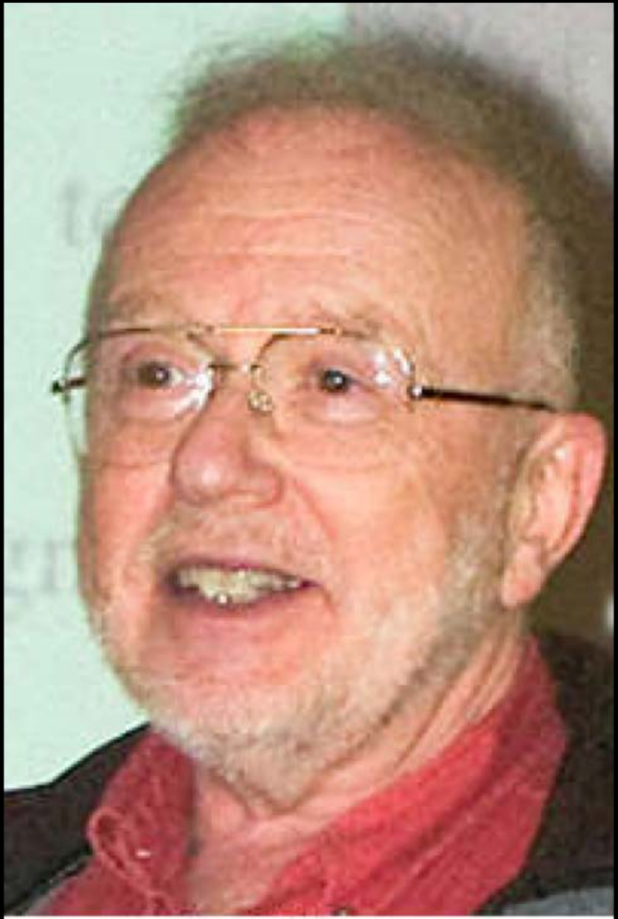
Edsger Dijkstra

“The art of programming is the art of organizing complexity.”

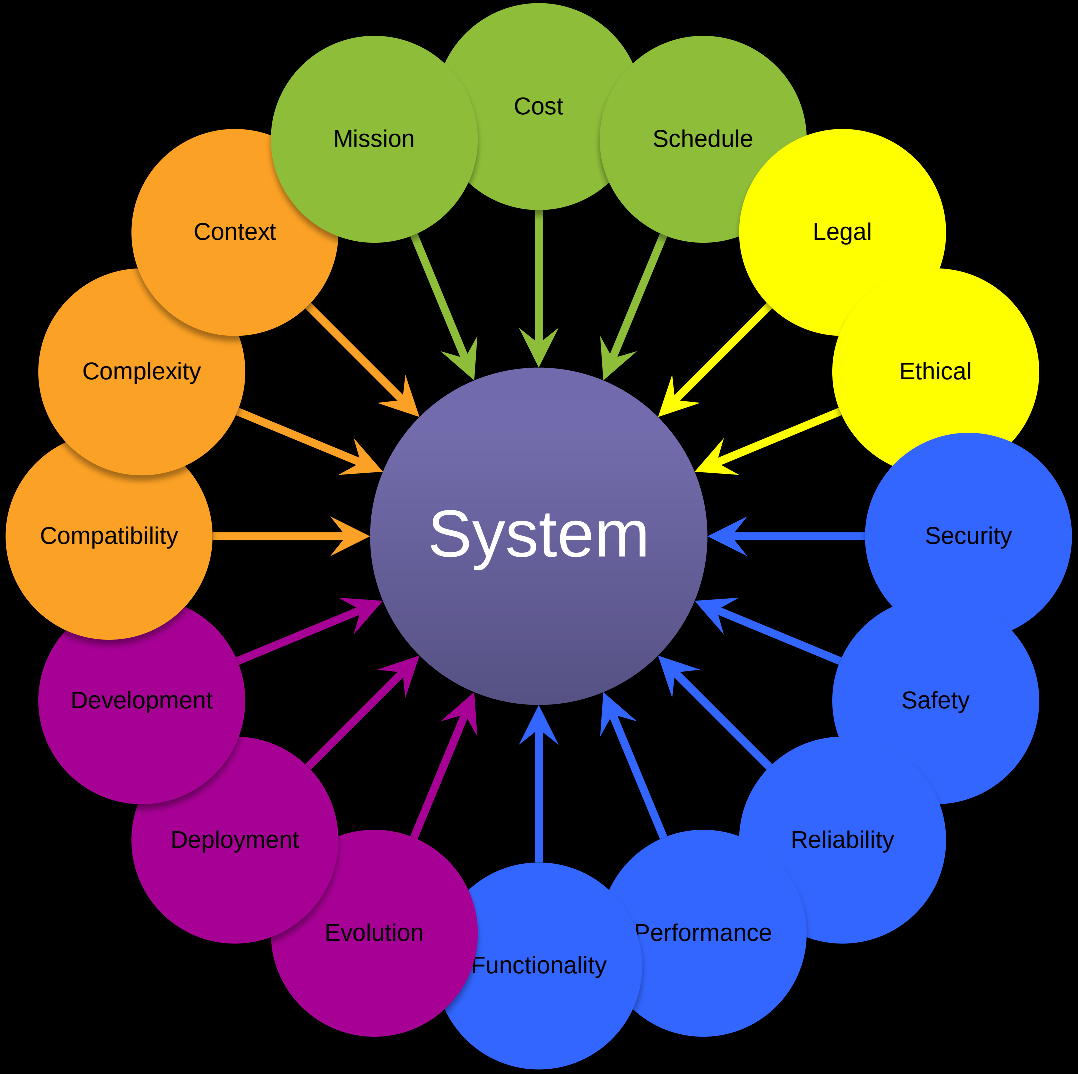
Donald Knuth

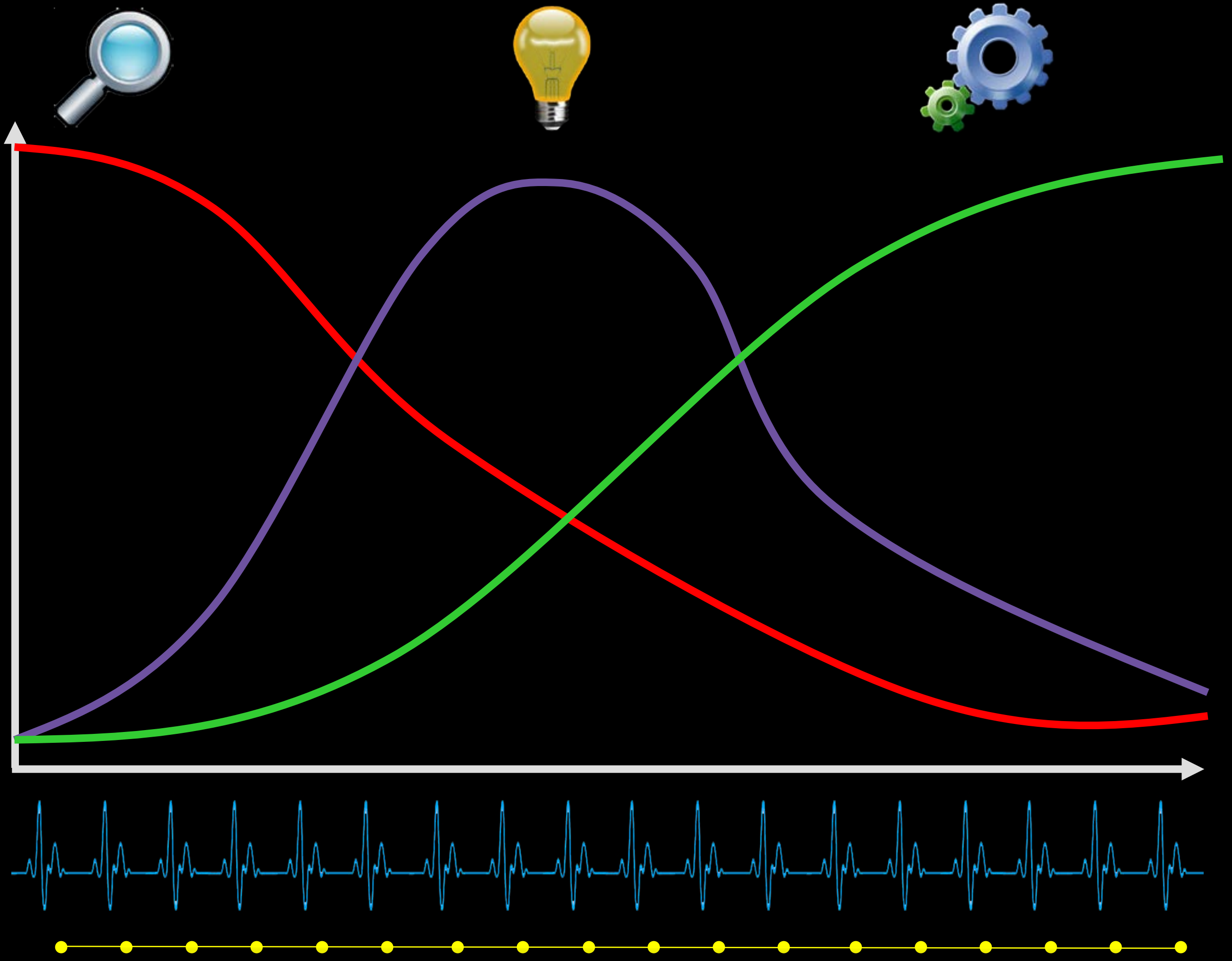
“Computer programming is an art, because it applies accumulated knowledge to the world, and especially because it produces objects of beauty.”

“Software engineering is often treated as a branch of computer science. This is akin to regarding chemical engineering as a branch of chemistry. We need both chemists and chemical engineers but they are very different. Chemists are scientists, chemical engineers are engineers. Software engineering and computer science have the same relationship.”

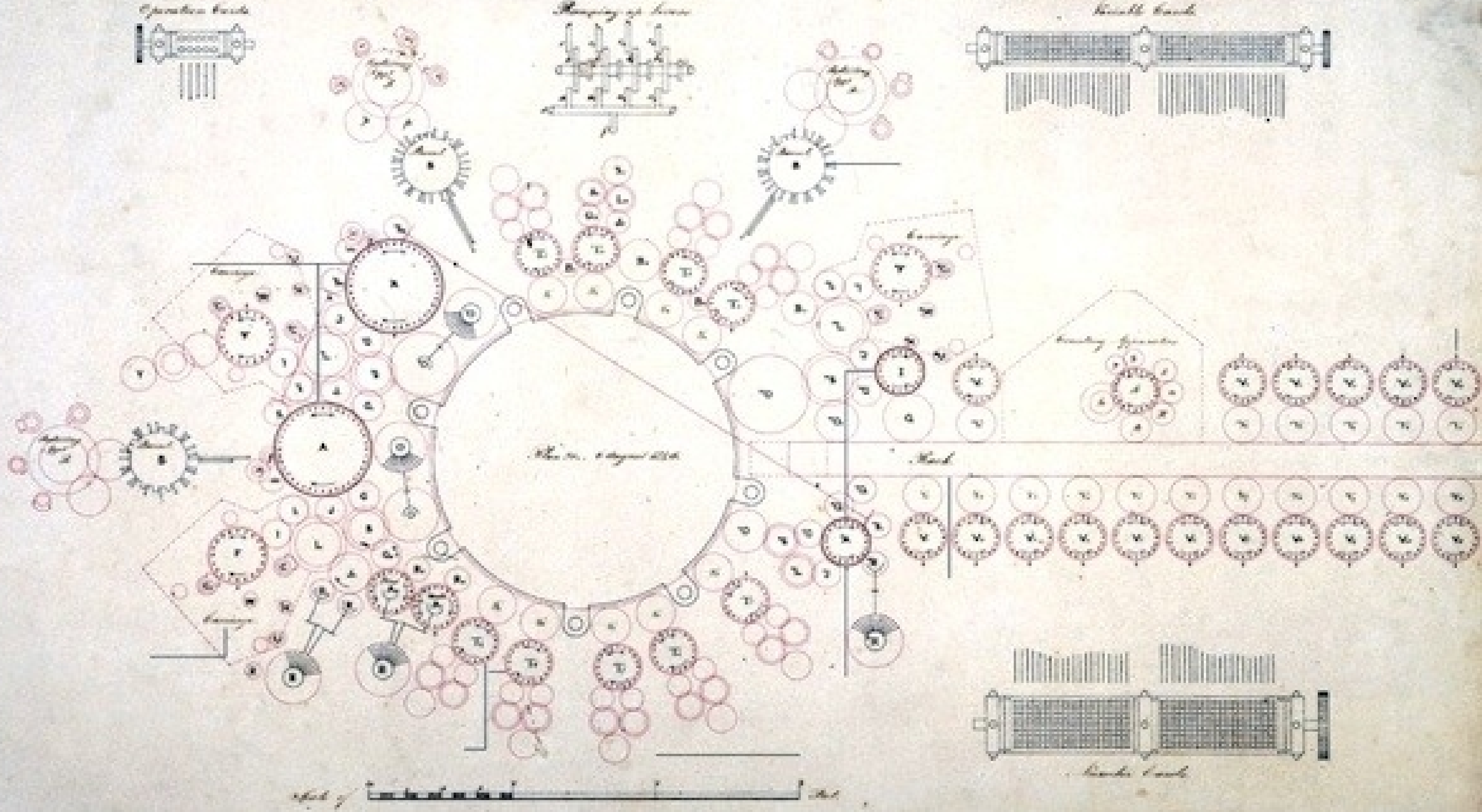


David Parnas


Cost
Mission
Schedule
Context
Legal
Complexity
Ethical
Compatibility
System
Security
Development
Safety
Deployment
Reliability
Evolution
Performance
Functionality



Brian Selic


Operation Cards
Variable Cards
Barrel
Carriage
Plan 25. 6 August 1840
Store
Counting Apparatus
Number Cards
Scale of
Feet

programming (1842)

Boolean algebra (1847)

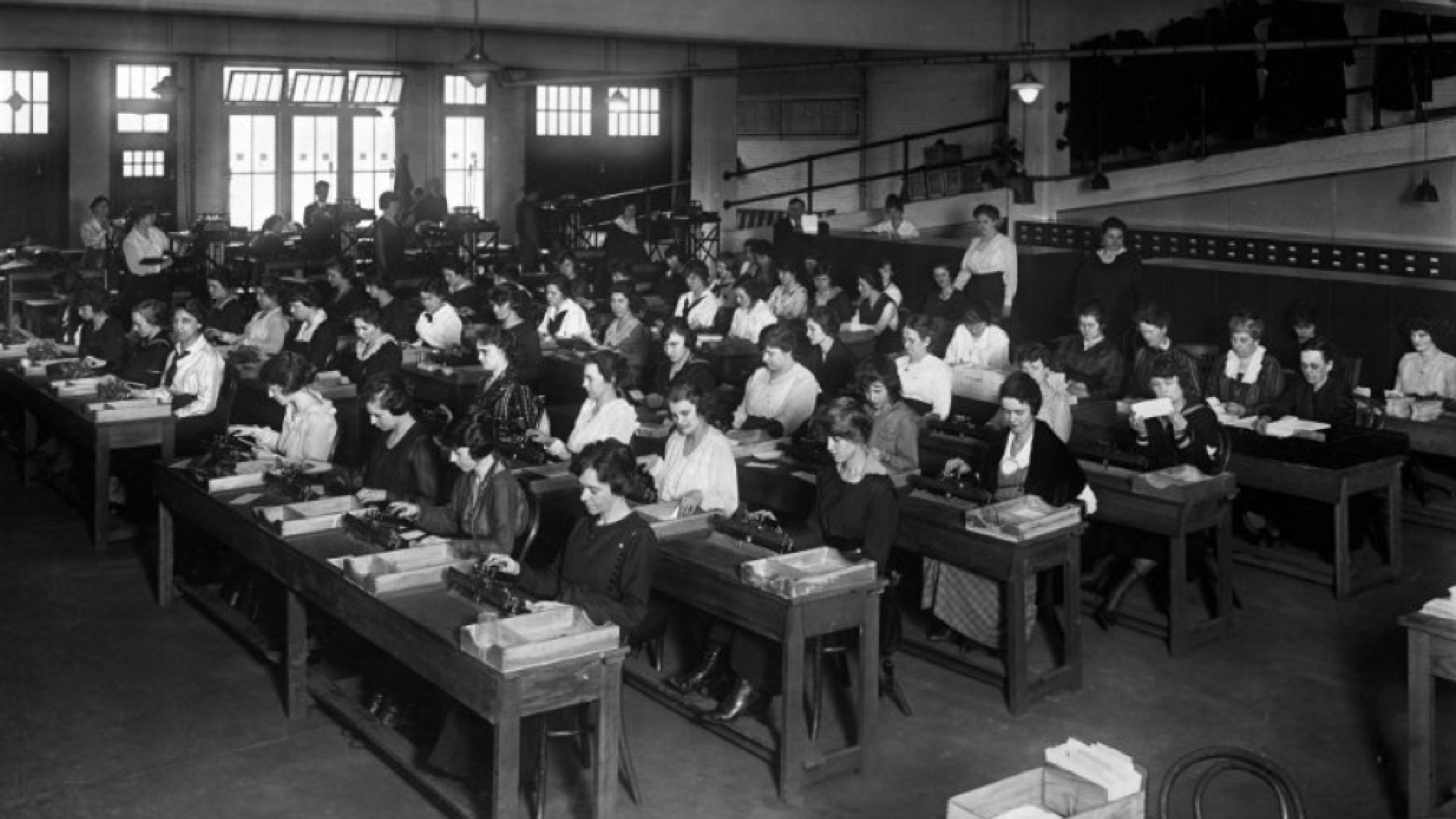
Dec. 1889

Annie Cannon

human computing (1896)

Henrietta Leavitt

human computing (1896)



process charts (1921)

analysis (1921)

human computing (1938)

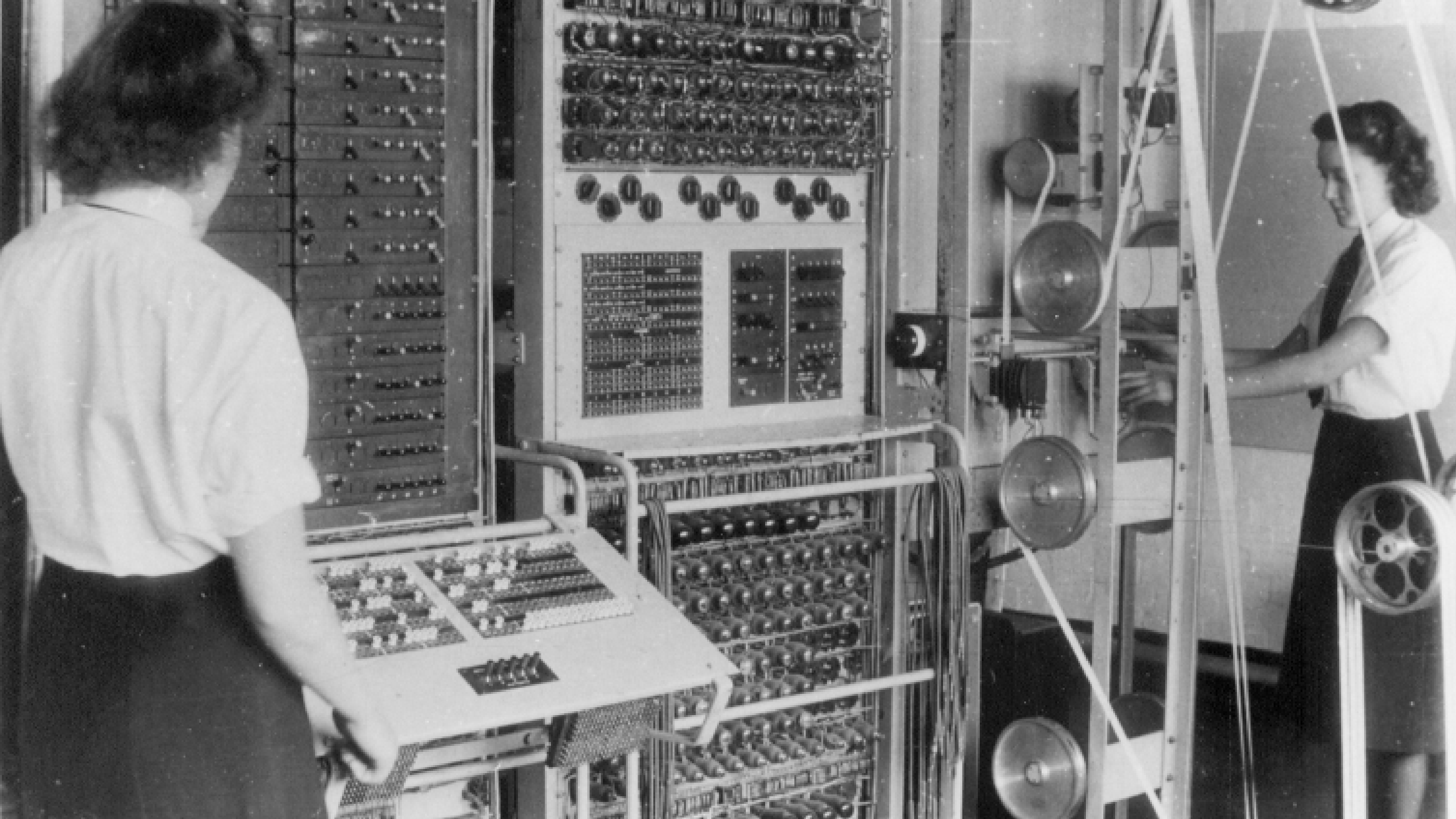
punch card methods (1940)

relay logic (1937)

theoretical computer science (1944)

electromechanical computation (1944)

machine-independent programming (1952)

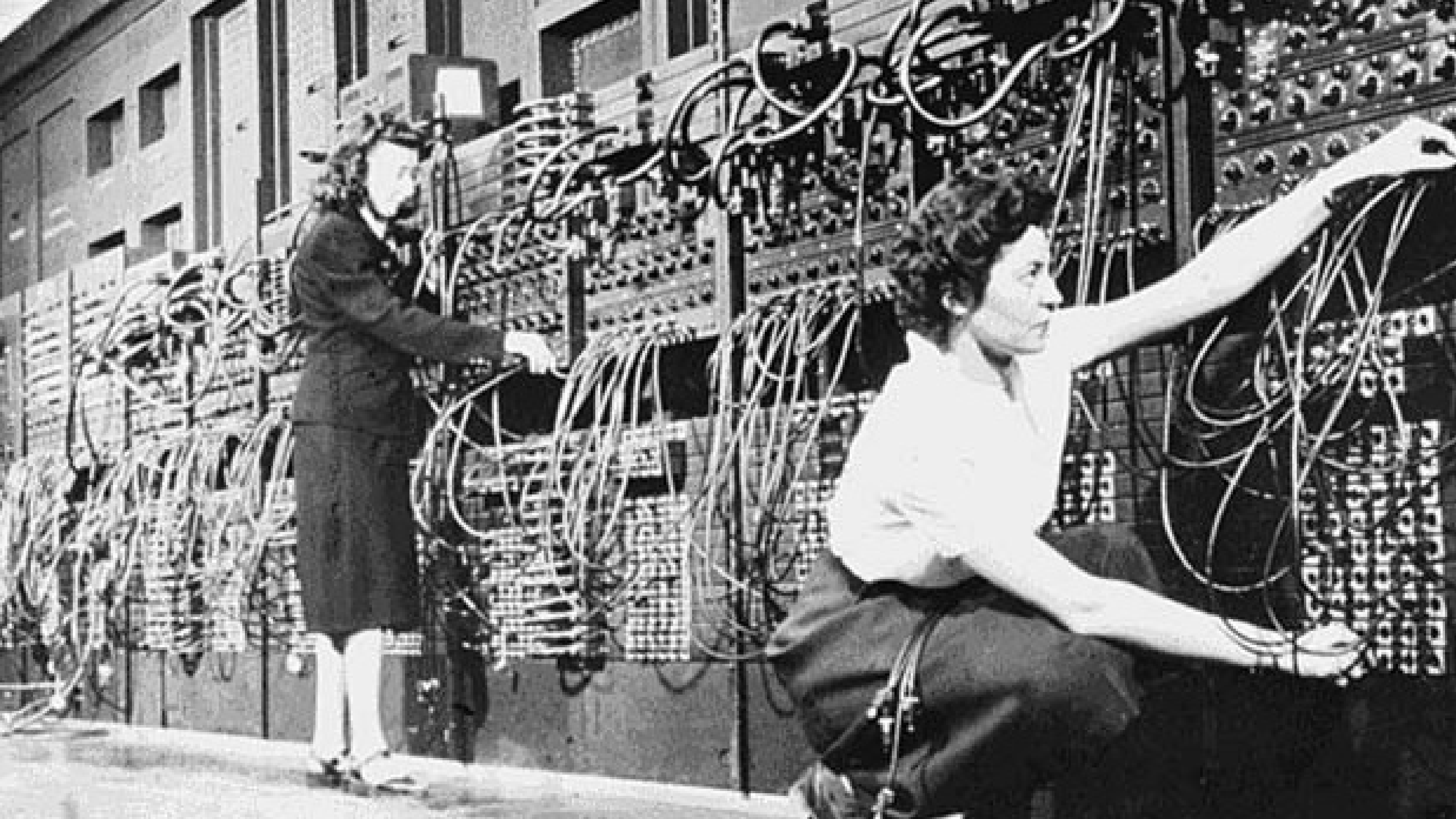


theoretical computer science (1936)

programmable computation (1943)

workflow (1943)

high order languages (1936)



Kay Antonelli

programming (1946)

Betty Snyder

programming (1946)

Frances Spence

programming (1946)

Ruth Teitelbaum

programming (1946)

Marlyn Wescoff

programming (1946)

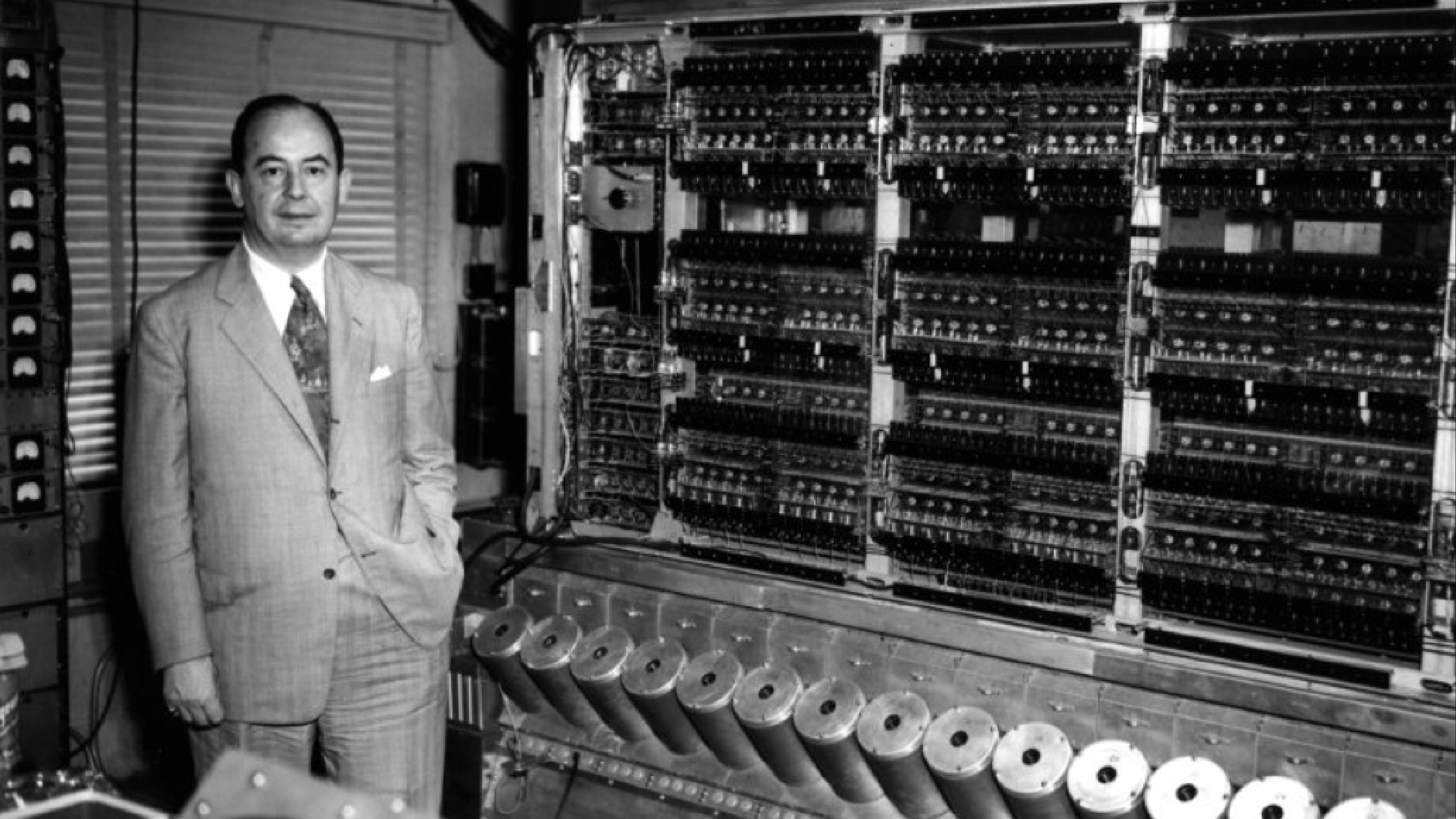
programming (1948)

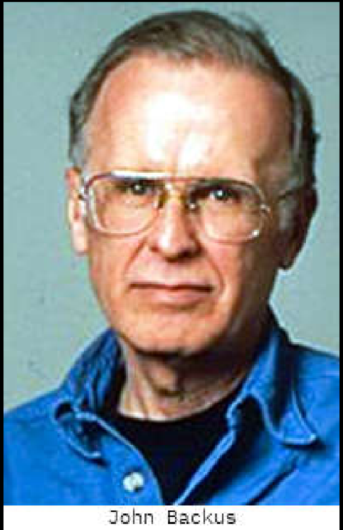
subroutine (1949)

subroutine (1949)

programming (1949)

programming (1949)





imperative
programming (1946)

flowchart (1947)

flowchart (1947)

IBM

SHARE

Educate · Network · Influence

1955

John Pinkerton

operating system (1951)

Grace Hopper

imperative programming (1960)

Robert Bemer

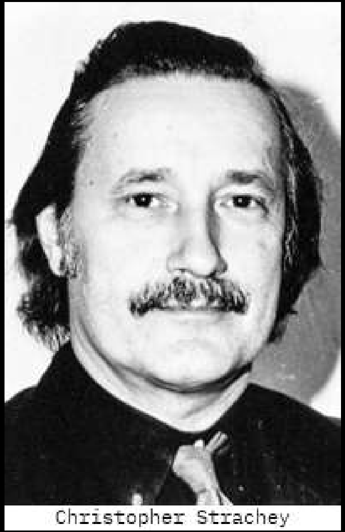
imperative programming (1960)

Jean Sammet

imperative programming (1960)

real time computing (1951)

program management (1957)



time sharing (1959)

programming services (1959)

IBM
System 360

Fred Brooks

project management (1964)

Larry Constantine

modular programming/coupling & cohesion/data flow (1968)

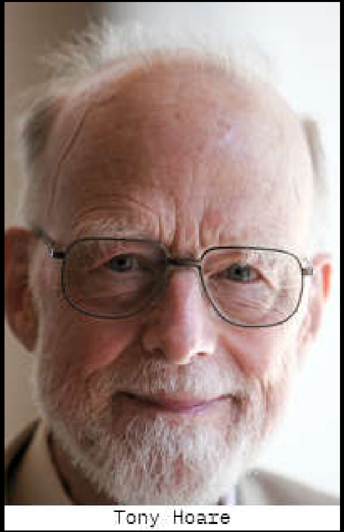
Edsger Dijkstra

structured programming (1969)

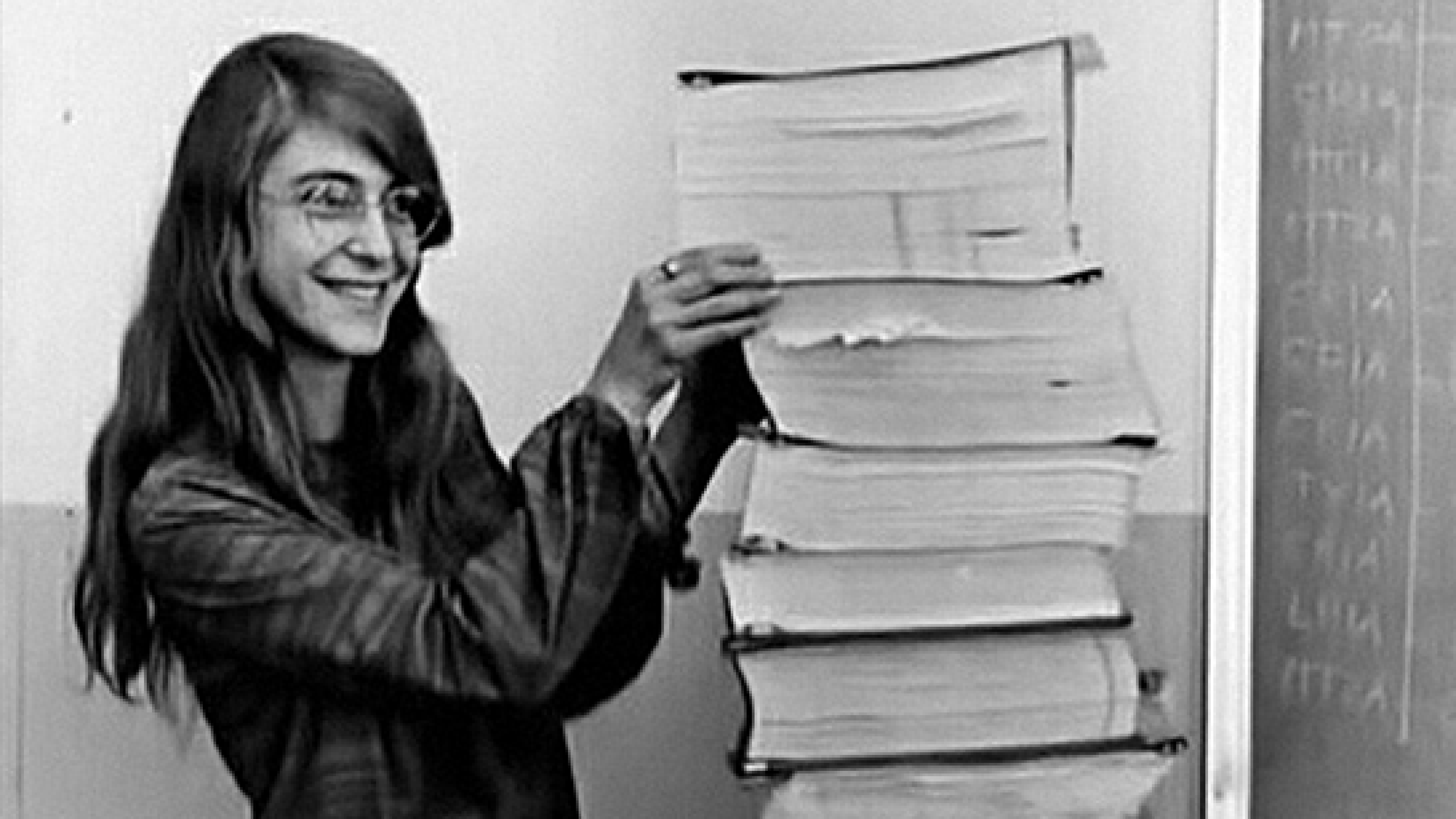
formal systems (1967)

object-oriented programming (1967)

object-oriented programming (1967)



formal systems (1969)



“Software during the early days of this project was treated like a stepchild and not taken as seriously as other engineering disciplines, such as hardware engineering; and it was regarded as an art and as magic, not a science. I had always believed that both art and science were involved in its creation, but at that time most thought otherwise. Knowing this, I fought to bring the software legitimacy so that it (and those building it) would be given its due respect and thus I began to use the term ‘software engineering’ to distinguish it from hardware and other kinds of engineering; yet, treat each type of engineering as part of the overall systems engineering process. When I first started using this phrase, it was considered to be quite amusing. It was an ongoing joke for a long time. They liked to kid me about my radical ideas. Software eventually and necessarily gained the same respect as any other discipline.”

https://medium.com/@verne/margaret-hamilton-the-engineer-who-took-the-apollo-to-the-moon-7d550c73d3fa

process (1970)

stepwise refinement/abstraction (1971/1976)

information hiding (1972)

abstract data types (1974)

entity-relationship modeling (1976)

SADT (1969)

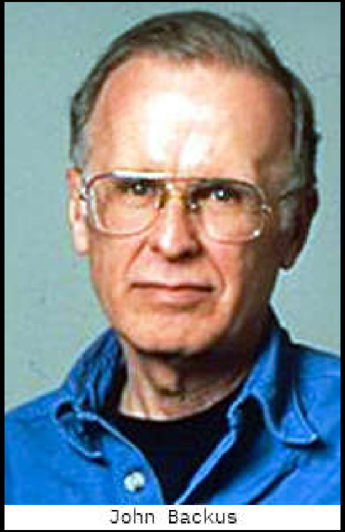
structured design (1972)

structured design (1972)

Jackson structured design (1975)

structured analysis and system specification (1978)

software inspection (1976)



functional programming (1977)

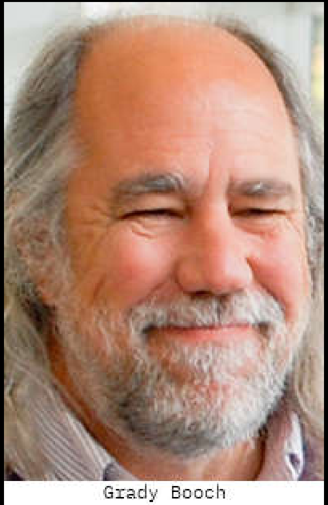
distributed computing (1978)

IBM

hello.

UNIFIED MODELING LANGUAGE™

1997


Grady Booch

Booch method (1986)

Jim Rumbaugh

OMT (1990)

Ivar Jacobson

Objectory (1990)

object-oriented
analysis (1988)

structured analysis (1989)

Responsibility
driven design (1989)

object-oriented
analysis and design (1990)

Barry Boehm

software engineering economics (1981)
spiral model (1988)

Victor Basili

empirical software engineering (1986)

Brad Cox

component based software engineering (1986)

Harlan Mills

clean room software engineering (1987)

Watts Humphrey

capability maturity model (1988)

Structured Systems
Analysis and Design
Methodologies (1981)

Defense Systems
Software Development
(1985)

Software Engineering Institute

1984

TOGAF®

1993

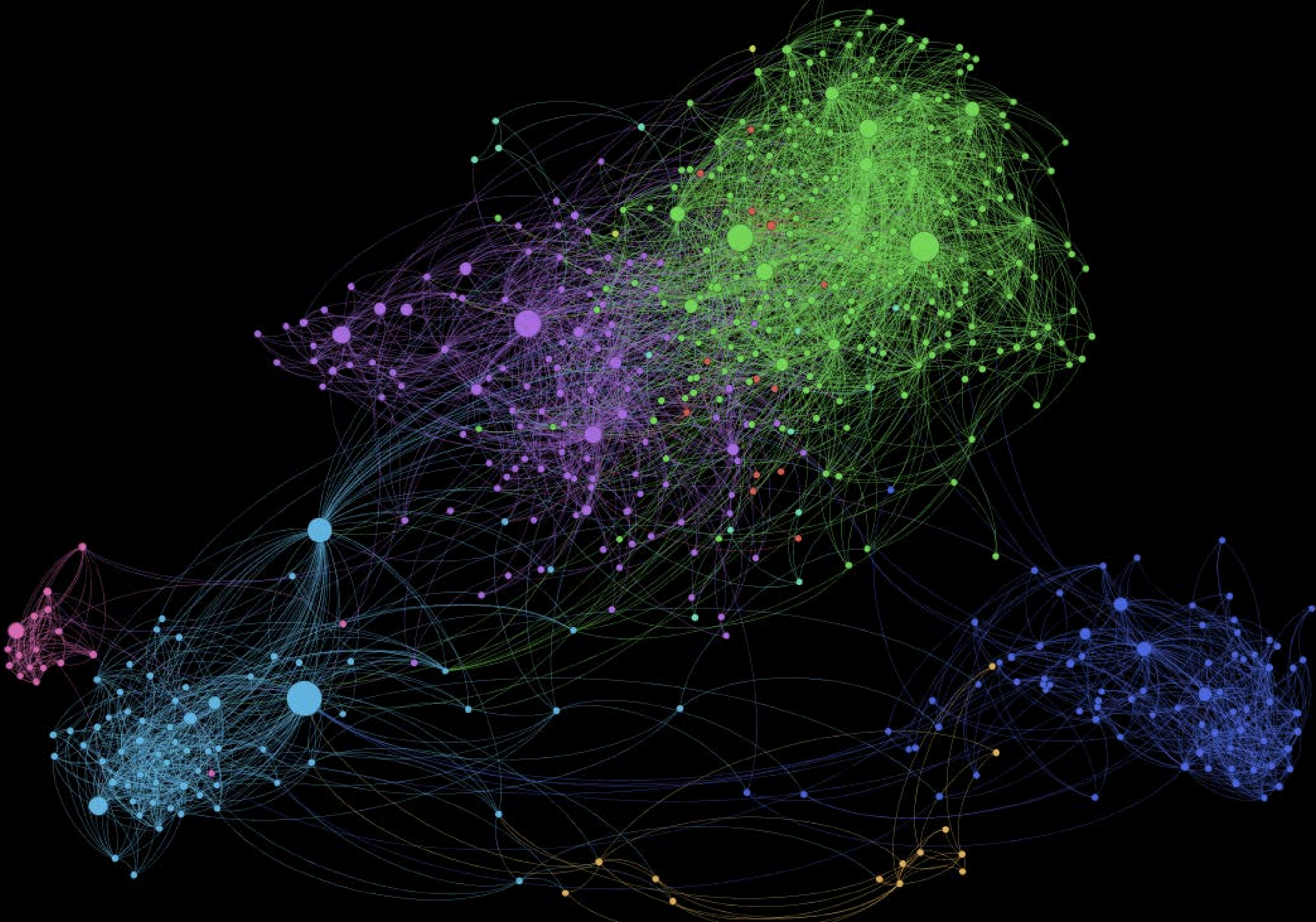
Information engineering/CASE
(1981)

Zachman framework (1987)

Literate programming (1983)

free software (1983)

visual programming (1991)



SCRUM (1995)

extreme programming (1996)

refactoring (1999)

Rational Unified Process (2000)

THE HILLSIDE GROUP

1993

Erich Gamma

Design patterns (1994)

Philippe Kruchten

Rational Unified Process/software architecture (1995)

Mary Shaw

software architecture (1996)

configuration management (1997)

open source (1997)

outsourcing (2001)

●●●○○ Bell LTE
5:15 PM
73%
Friday
26
Messages
Calendar
Photos
Camera
Weather
Clock
Maps
280
Videos
Notes
Reminders
Stocks
Game Center
Newsstand
iTunes Store
App Store
1
iBooks
Health
Passbook
Settings
Phone
Mail
Safari
Music

Agile Alliance

2001

Linus Torvalds

git (2005)

Jim Coplien

organizational patterns (2005)

Jeannette Wing

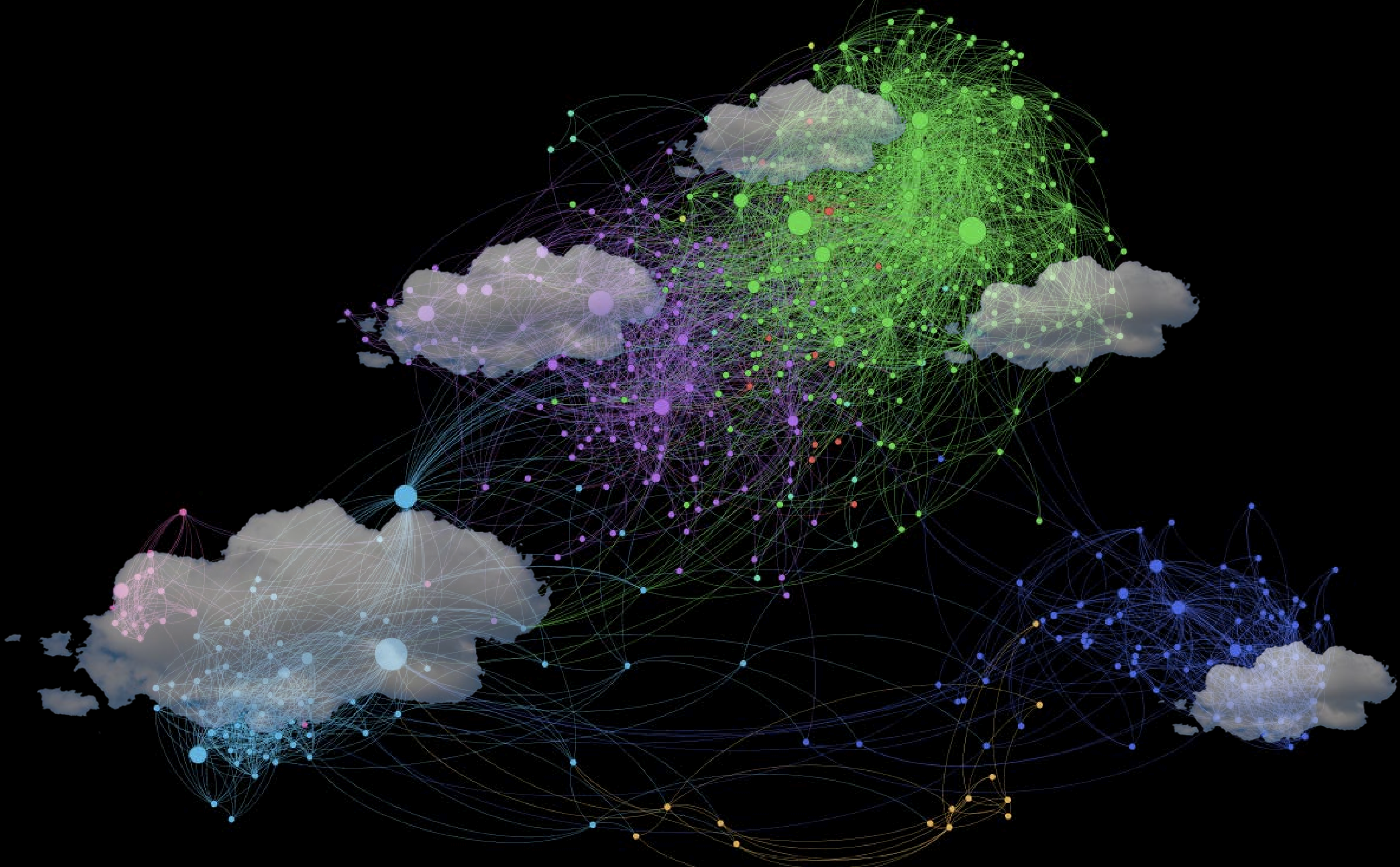
computational thinking (2006)

Joel Spolsky

Stackoverflow (2007)

Robert Martin

clean code (2008)

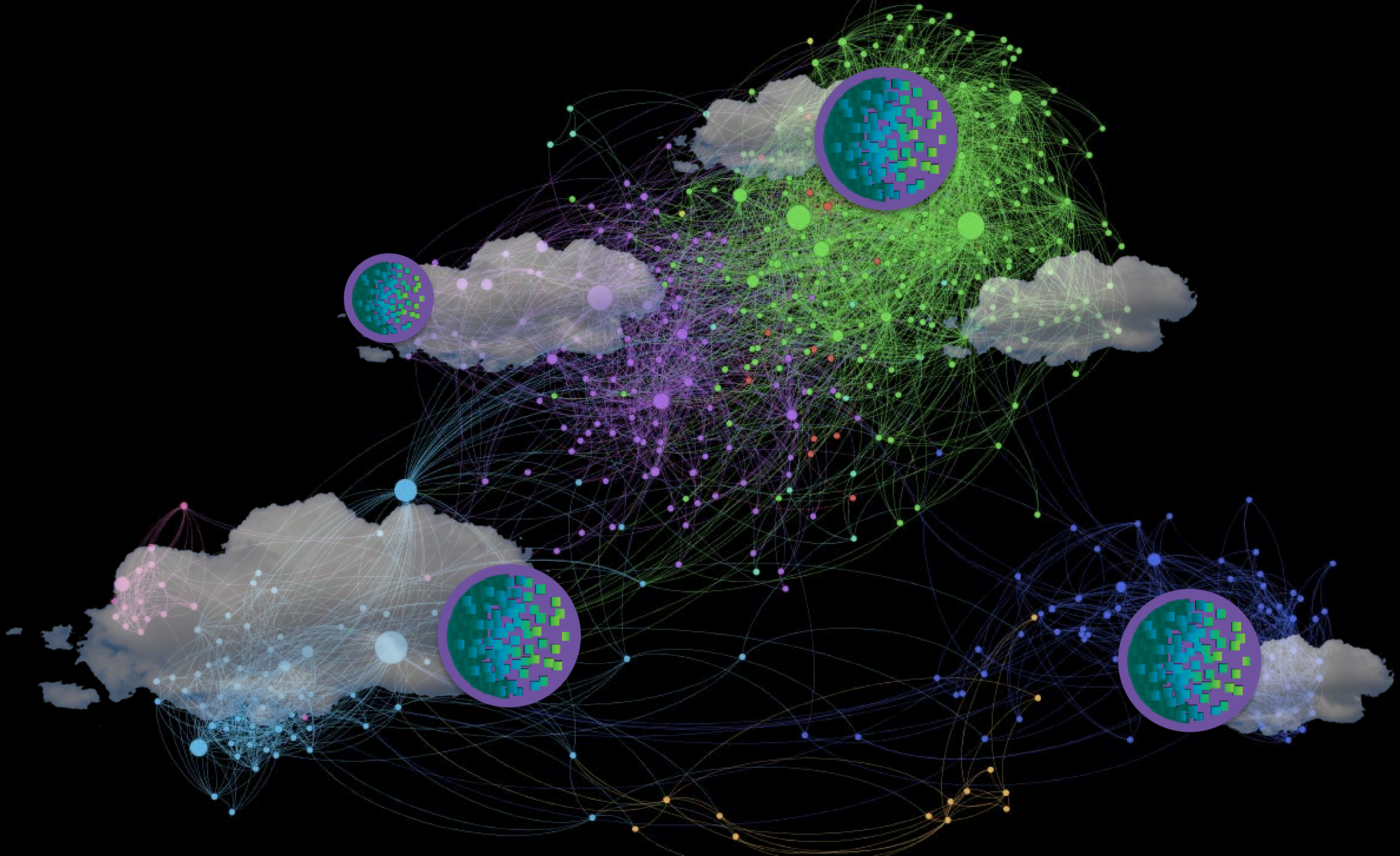


Andrew Shafer

devops (2008)

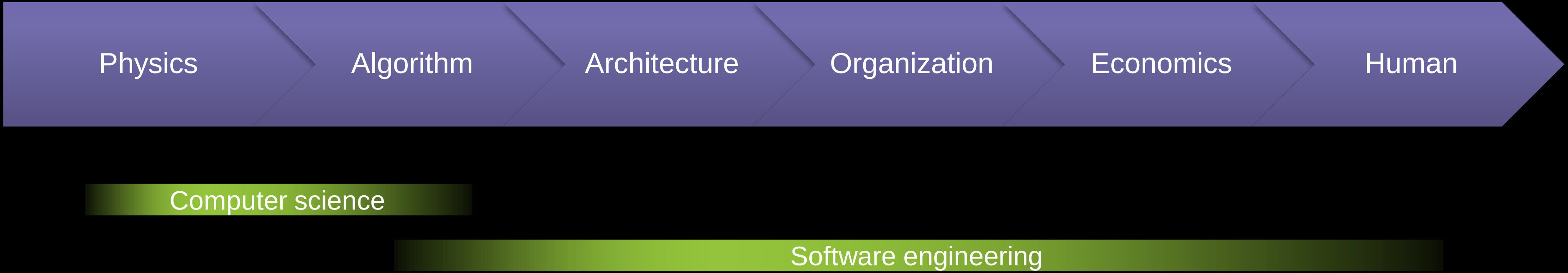
Patrick Debois

devops (2008)



platform computing (2000)

platform computing (2006)



Physics

Algorithm

Architecture

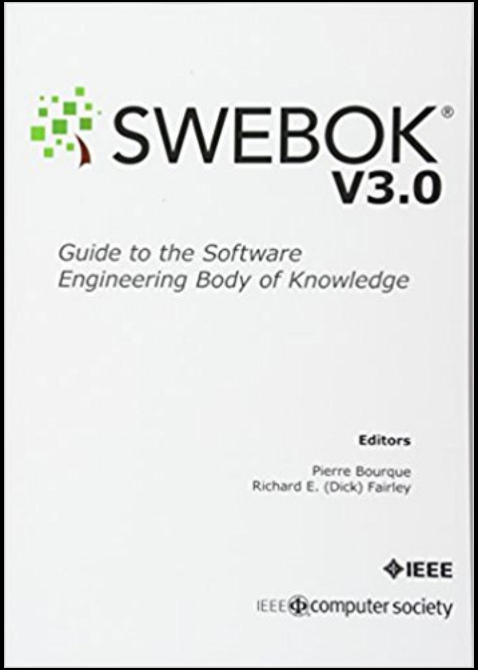
Organization

Economics

Human

Computer science

Software engineering



SWEBOK®
V3.0

Guide to the Software Engineering Body of Knowledge

Editors

Pierre Bourque
Richard E. (Dick) Fairley

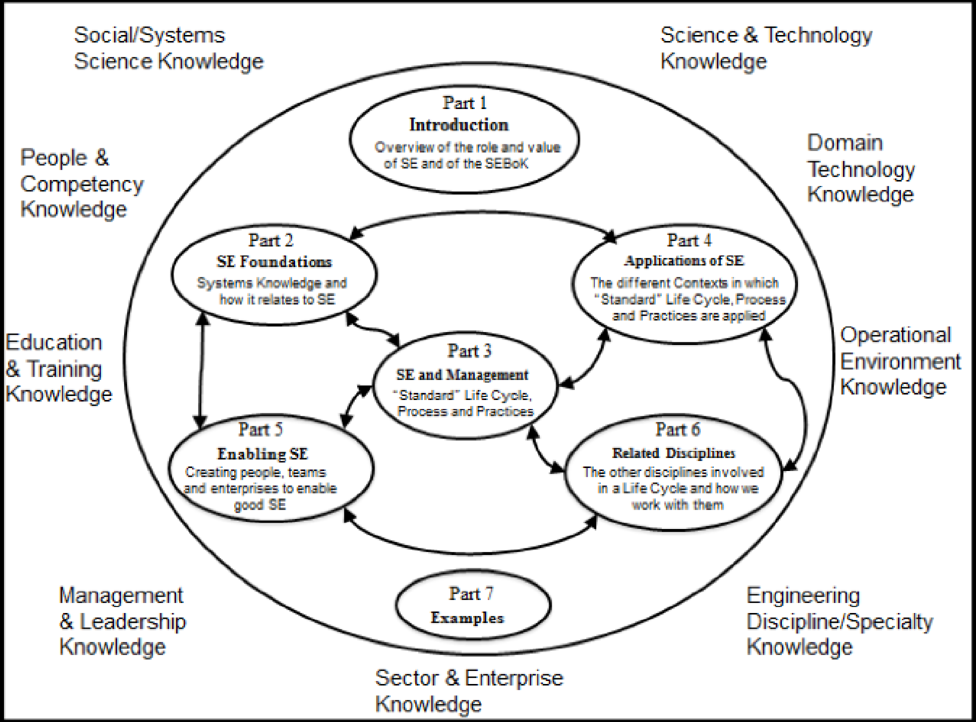
IEEE

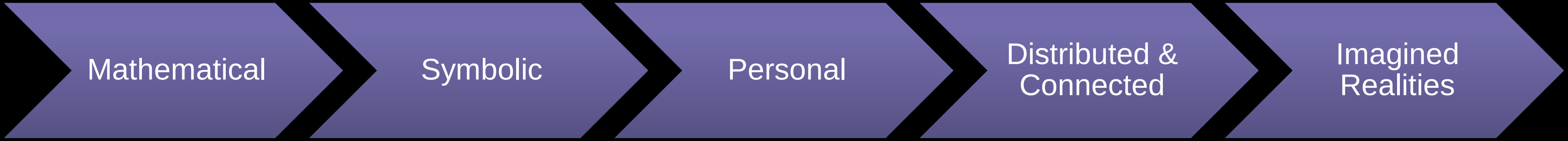
IEEE computer society

The *Software Engineering Body of Knowledge* was first released in 2004 (its current version was published in 2014), and addresses

- Software requirements
- Software design
- Software construction
- Software testing
- Software maintenance
- Software configuration management
- Software engineering management
- Software engineering process
- Software engineering models and methods

The *Systems Engineering Body of Knowledge* is an effort by the International Council of Systems Engineering (INCOSE), the Systems Engineering Research Center (SERC), and the IEEE Computer Society to codify the best practices of systems engineering.


Social/Systems Science Knowledge
Science & Technology Knowledge
People & Competency Knowledge
Domain Technology Knowledge
Education & Training Knowledge
Operational Environment Knowledge
Management & Leadership Knowledge
Engineering Discipline/Specialty Knowledge
Sector & Enterprise Knowledge
Part 1
Introduction
Overview of the role and value of SE and of the SEBoK
Part 2
SE Foundations
Systems Knowledge and how it relates to SE
Part 4
Applications of SE
The different Contexts in which "Standard" Life Cycle, Process and Practices are applied
Part 3
SE and Management
"Standard" Life Cycle, Process and Practices
Part 5
Enabling SE
Creating people, teams and enterprises to enable good SE
Part 6
Related Disciplines
The other disciplines involved in a Life Cycle and how we work with them
Part 7
Examples



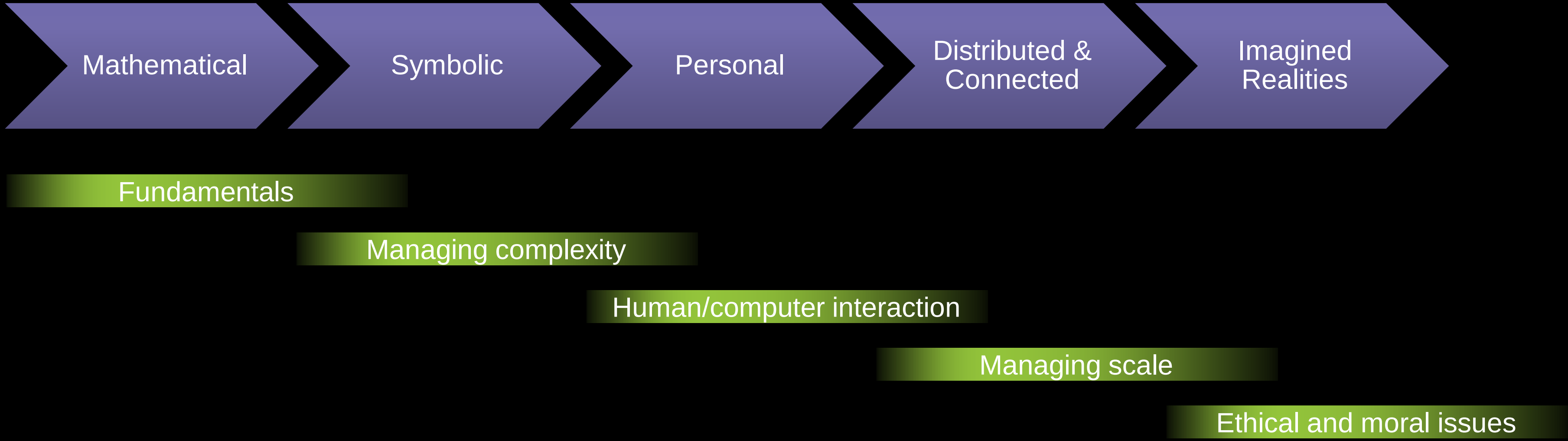
Mathematical

Symbolic

Personal

Distributed & Connected

Imagined Realities



Mathematical

Symbolic

Personal

Distributed & Connected

Imagined Realities

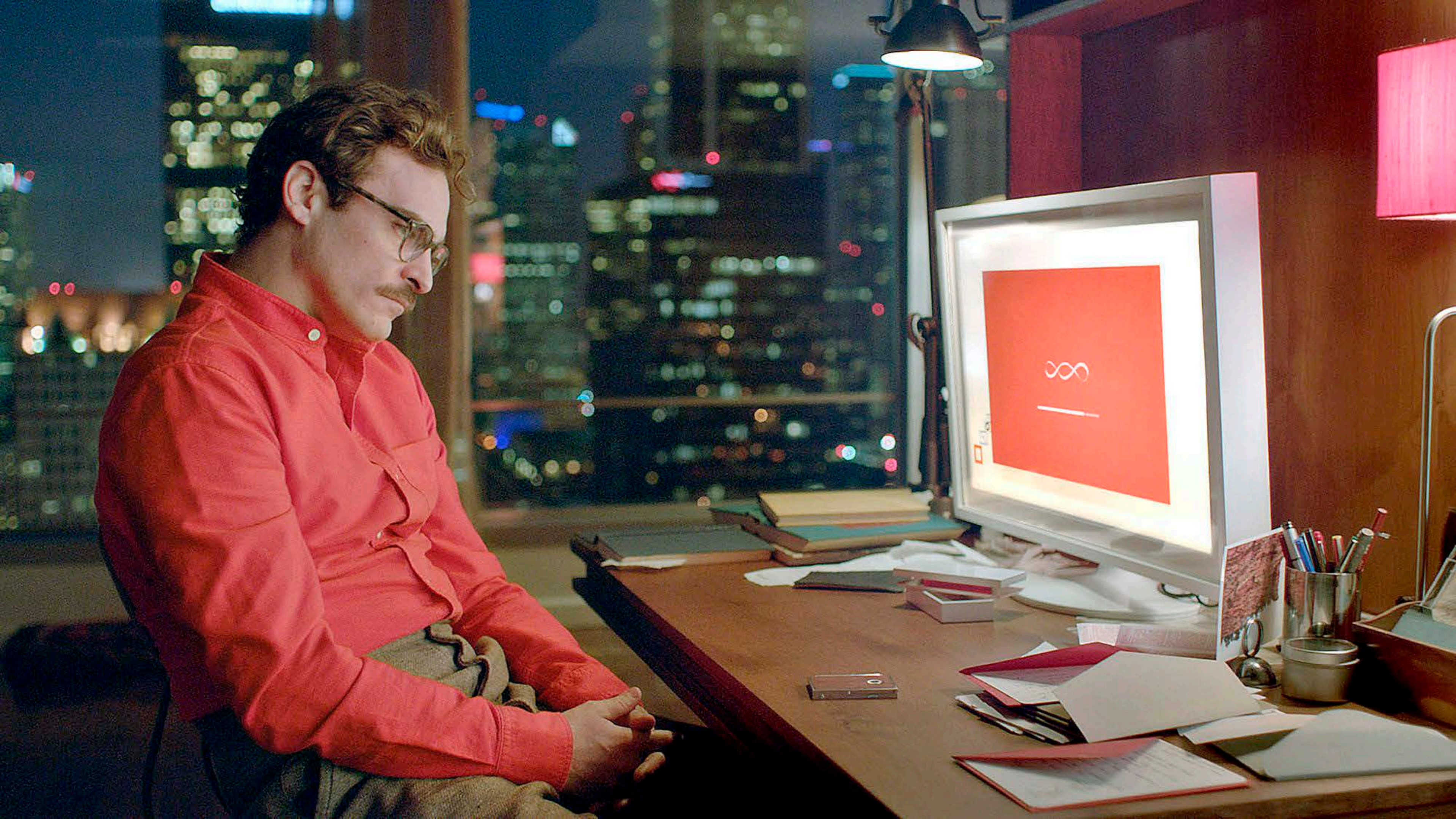
Fundamentals

Managing complexity

Human/computer interaction

Managing scale

Ethical and moral issues



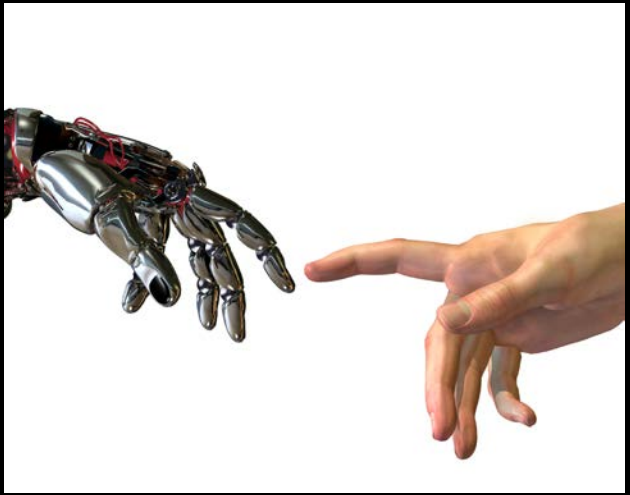
The fundamentals always apply:

- Crisp abstractions
- Clear separation of concerns
- Balanced distribution of responsibilities
- Simplicity

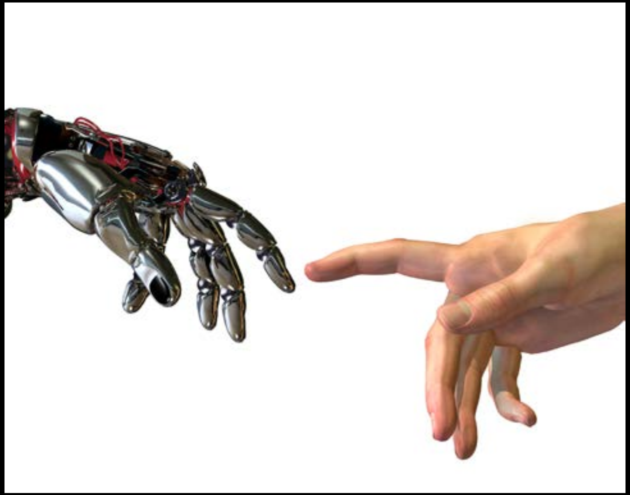
Grow a system through the iterative, incremental, and continuous release of its executable architecture.

Still, there is work to be done:

- Orchestrating hybrid symbolic, connectionist, and quantum models of computation
- The architectural pendulum
- The edge/cloud pendulum
- Scale, in the presence of untrusted components, legacy of considerable inertia, and the general public



Software is the invisible writing that whispers the stories of possibility to our hardware…



…and you are the storytellers.

Grady Booch

IBM Fellow & Chief Scientist for Software Engineering

Email: gbooch@us.ibm.com
Twitter: @grady_booch
Web: computingthehumanexperience.com

ACM: The Learning Continues…

- Questions/comments about this webcast? learning@acm.org
- ACM's Discourse Page: http://on.acm.org
- ACM Learning Webinars (on-demand archive): http://webinar.acm.org
- ACM Learning Center: http://learning.acm.org
- ACM Queue: http://queue.acm.org
- ACM AiDecentralized Conference: https://www.aidecentralized.com/