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: computingethehumanexperience.com
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

[Signed]

Charles E. Hayden

[Seal]

Witnessed by

[Signature]

[Seal]
The term *systems engineering* dates back to Bell Telephone Laboratories in the early 1940s, with major applications of systems engineering during World War II.
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.
The first computers were human (and, for the most part, women).

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

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

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

“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.”
programming (1842)

Boolean algebra (1847)
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)
programming (1948) subroutine (1949) subroutine (1949) programming (1949) programming (1949)
imperative programming (1946)

flowchart (1947)

flowchart (1947)
operating system (1951) 

imperative programming (1960)

imperative programming (1960)

imperative programming (1960)
real time computing (1951)
program management (1957)
time sharing (1959)
programming services (1959)
Fred Brooks

project management (1964)

Larry Constantine

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

Edsger Dijkstra

structured programming (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.”

software inspection (1976)  
functional programming (1977)  
distributed computing (1978)
Booch method (1986)

OMT (1990)

Objectionary (1990)
object-oriented analysis (1988)
structured analysis (1989)
Responsibility driven design (1989)
object-oriented analysis and design (1990)

Defense Systems Software Development (1985)

Information engineering/CASE (1981)

Zachman framework (1987)

TOGAF®

1984

1993
Literate programming (1983)

free software (1983)

visual programming (1991)
1993

Design patterns (1994)  
software architecture (1996)
2001

Linus Torvalds: git (2005)
Jim Coplien: organizational patterns (2005)
Jeannette Wing: computational thinking (2006)
Joel Spolsky: Stackoverflow (2007)
Robert Martin: clean code (2008)
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.
Mathematical | Symbolic | Personal | Distributed & Connected | Imagined Realities
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/