Software engineering

Software engineering is an engineering approach to software development.[1][2][3] A practitioner, a **software engineer**, applies the engineering design process to develop software.

The terms *programmer* and *coder* overlap *software engineer*, but they imply only the construction aspect of typical software engineer workload.[4]

A software engineer applies a software development process,[1][5] which involves the definition, implementation, testing, management and maintenance of software systems and with development of the software development process itself.

History

Beginning in the 1960s, software engineering was recognized as a separate field of engineering.

The development of software engineering was seen as a struggle. It was difficult to keep up with the hardware which caused many problems for software engineers. Problems included software that was over budget, exceeded deadlines, required extensive debugging and maintenance, and unsuccessfully met the needs of consumers or was never even completed.

In 1968 NATO held the first Software Engineering conference where issues related to software were addressed: guidelines and best practices for the development of software were established.[6]

The origins of the term software engineering have been attributed to various sources. The term appeared in a list of services offered by companies in the June 1965 issue of "Computers and Automation"[7] and was used more formally in the August 1966 issue of Communications of the ACM (Volume 9, number 8) "letter to the ACM membership" by the ACM President Anthony A. Oettinger.[8][9] It is also associated with the title of a NATO conference in 1968 by Professor Friedrich L. Bauer.[10] Margaret Hamilton described the discipline of "software engineering" during the Apollo missions to give what they were doing legitimacy.[11] At the time there was perceived to be a "software crisis".[12][13][14] The 40th International Conference on Software Engineering (ICSE 2018) celebrates 50 years of "Software Engineering" with the Plenary Sessions' keynotes of Frederick Brooks[15] and Margaret Hamilton.[16]

In 1984, the Software Engineering Institute (SEI) was established as a federally funded research and development center headquartered on the campus of Carnegie Mellon University in Pittsburgh, Pennsylvania, United States. Watts Humphrey founded the SEI Software Process Program, aimed at understanding and managing the software engineering process. The Process Maturity Levels introduced would become the Capability Maturity Model Integration for Development(CMMI-DEV), which has defined how the US Government evaluates the abilities of a software development team.

Modern, generally accepted best-practices for software engineering have been collected by the ISO/IEC JTC 1/SC 7 subcommittee and published as the Software Engineering Body of Knowledge (SWEBOK).[5] Software engineering is considered one of the major computing disciplines.[17]

Terminology

Definition

Notable definitions of software engineering include:

- "The systematic application of scientific and technological knowledge, methods, and experience to the design, implementation, testing, and documentation of software"—The Bureau of Labor Statistics—IEEE *Systems and software engineering – Vocabulary*[18]
- "The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software"—IEEE *Standard Glossary of Software Engineering Terminology*[19]
- "an engineering discipline that is concerned with all aspects of software production"—Ian Sommerville[20]
- "the establishment and use of sound engineering principles in order to economically obtain software that is reliable and works efficiently on real machines"—Fritz Bauer[21]
- "a branch of computer science that deals with the design, implementation, and maintenance of complex computer programs"—Merriam-Webster[22]
- "'software engineering' encompasses not just the act of writing code, but all of the tools and processes an organization uses to build and maintain that code over time. [...] Software engineering can be thought of as 'programming integrated over time.'"—Software Engineering at Google[23]

The term has also been used less formally:

- as the informal contemporary term for the broad range of activities that were formerly called computer programming and systems analysis;[24]
- as the broad term for all aspects of the *practice* of computer programming, as opposed to the *theory* of computer programming, which is formally studied as a sub-discipline of computer science;[25]
- as the term embodying the *advocacy* of a specific approach to computer programming, one that urges that it be treated as an engineering discipline rather than an art or a craft, and advocates the codification of recommended practices.[26]

Etymology

Margaret Hamilton promoted the term "software engineering" during her work on the Apollo program. The term "engineering" was used to acknowledge that the work should be taken just as seriously as other contributions toward the advancement of technology. Hamilton details her use of the term:

> When I first came up with the term, no one had heard of it before, at least in our world. It was an ongoing joke for a long time. They liked to kid me about my radical ideas. It was a memorable day when one of the most respected hardware gurus explained to everyone in a meeting that he agreed with me that the process of building software should also be considered

> an engineering discipline, just like with hardware. Not because of his acceptance of the new "term" per se, but because we had earned his and the acceptance of the others in the room as being in an engineering field in its own right.[27]

Suitability

Individual commentators have disagreed sharply on how to define *software engineering* or its legitimacy as an engineering discipline. David Parnas has said that software engineering is, in fact, a form of engineering.[28][29] Steve McConnell has said that it is not, but that it should be.[30] Donald Knuth has said that programming is an art and a science.[31] Edsger W. Dijkstra claimed that the terms *software engineering* and *software engineer* have been misused in the United States.[32]

Workload

Requirements analysis

Requirements engineering is about elicitation, analysis, specification, and validation of requirements for software. Software requirements can be functional, non-functional or domain.

Functional requirements describe expected behaviors (i.e. outputs). Non-functional requirements specify issues like portability, security, maintainability, reliability, scalability, performance, reusability, and flexibility. They are classified into the following types: interface constraints, performance constraints (such as response time, security, storage space, etc.), operating constraints, life cycle constraints (maintainability, portability, etc.), and economic constraints. Knowledge of how the system or software works is needed when it comes to specifying non-functional requirements. Domain requirements have to do with the characteristic of a certain category or domain of projects.[33]

Design

Software design is the process of making high-level plans for the software. Design is sometimes divided into levels:

- Interface design plans the interaction between a system and its environment as well as the inner workings of the system
- Architectural design plans the major components of a system; including their responsibilities, properties, and interfaces between them.
- Detailed design plans internal elements; including their properties, relationships, algorithms and data structures.[34]

Construction

Software construction,[1][5] typically involves programming (a.k.a. coding), unit testing, integration testing, and debugging so as to implement the design. “Software testing is related to, but different from, ... debugging”.[5] Testing during this phase is generally performed by the programmer and with the purpose to verify that the code behaves per design and to know when the code is ready for next level testing.

Testing

Software testing[1][5] is an empirical, technical investigation conducted to provide stakeholders with information about the quality of the software under test.

When described separately from construction, testing typically is performed by test engineers or quality assurance instead of the programmers who wrote it and is performed at the system level and is considered an aspect of software quality.

Program analysis

Program analysis is the process of analyzing computer programs with respect to an aspect such as performance, robustness, and security.

Maintenance

Software maintenance[1][5] refers to supporting the software after release. It may include but is not limited to: error correction, optimization, deletion of unused and discarded features, and enhancement of existing features.

Usually, maintenance takes up about 40% to 80% of project cost, therefore, focusing on maintenance can reduce development costs.[35]

Education

Knowledge of computer programming is a prerequisite for becoming a software engineer. In 2004, the IEEE Computer Society produced the SWEBOK, which has been published as ISO/IEC Technical Report 1979:2005, describing the body of knowledge that they recommend to be mastered by a graduate software engineer with four years of experience.[36] Many software engineers enter the profession by obtaining a university degree or training at a vocational school. One standard international curriculum for undergraduate software engineering degrees was defined by the Joint Task Force on Computing Curricula of the IEEE Computer Society and the Association for Computing Machinery, and updated in 2014.[37] A number of universities have Software Engineering degree programs; as of 2010, there were 244 Campus Bachelor of Software Engineering programs, 70 Online programs, 230 Masters-level programs, 41 Doctorate-level programs, and 69 Certificate-level programs in the United States.

In addition to university education, many companies sponsor internships for students wishing to pursue careers in information technology. These internships can introduce the student to interesting real-world tasks that typical software engineers encounter every day. Similar experience can be gained through military service in software engineering.

Software engineering degree programs

Half of all practitioners today have degrees in computer science, information systems, or information technology. A small, but growing, number of practitioners have software engineering degrees. In 1987, the Department of Computing at Imperial College London introduced the first three-year software engineering Bachelor's degree in the UK and the world; in the following year, the University of Sheffield established a similar program.[38] In 1996, the Rochester Institute of Technology established the first software engineering bachelor's degree program in the United States, however, it did not obtain ABET accreditation until 2003, the same time as Rice University, Clarkson University, Milwaukee School of Engineering and Mississippi State University obtained theirs.[39] In 1997, PSG College of Technology in Coimbatore, India was the first to start a five-year integrated Master of Science degree in Software Engineering.

Since then, software engineering undergraduate degrees have been established at many universities. A standard international curriculum for undergraduate software engineering degrees, SE2004, was defined by a steering committee between 2001 and 2004 with funding from the Association for Computing Machinery and the IEEE Computer Society. As of 2004, in the U.S., about 50 universities offer software engineering degrees, which teach both computer science and engineering principles and practices. The first software engineering Master's degree was established at Seattle University in 1979. Since then graduate software engineering degrees have been made available from many more universities. Likewise in Canada, the Canadian Engineering Accreditation Board (CEAB) of the Canadian Council of Professional Engineers has recognized several software engineering programs.

In 1998, the US Naval Postgraduate School (NPS) established the first doctorate program in Software Engineering in the world. Additionally, many online advanced degrees in Software Engineering have appeared such as the Master of Science in Software Engineering (MSE) degree offered through the Computer Science and Engineering Department at California State University, Fullerton. Steve McConnell opines that because most universities teach computer science rather than software engineering, there is a shortage of true software engineers.[40] ETS (École de technologie supérieure) University and UQAM (Université du Québec à Montréal) were mandated by IEEE to develop the Software Engineering Body of Knowledge (SWEBOK), which has become an ISO standard describing the body of knowledge covered by a software engineer.[5]

Profession

Legal requirements for the licensing or certification of professional software engineers vary around the world. In the UK, there is no licensing or legal requirement to assume or use the job title Software Engineer. In some areas of Canada, such as Alberta, British Columbia, Ontario,[41] and Quebec, software engineers can hold the Professional Engineer (P.Eng) designation and/or the Information Systems Professional (I.S.P.) designation. In Europe, Software Engineers can obtain the European Engineer (EUR ING) professional title. Software Engineers can also become professionally qualified as a Chartered Engineer through the British Computer Society.

In the United States, the NCEES began offering a Professional Engineer exam for Software Engineering in 2013, thereby allowing Software Engineers to be licensed and recognized.[42] NCEES ended the exam after April 2019 due to lack of participation.[43] Mandatory licensing is currently still largely debated, and perceived as controversial.

The IEEE Computer Society and the ACM, the two main US-based professional organizations of software engineering, publish guides to the profession of software engineering. The IEEE's *Guide to the Software Engineering Body of Knowledge – 2004 Version*, or SWEBOK, defines the field and describes the

knowledge the IEEE expects a practicing software engineer to have. The most current SWEBOK v3 is an updated version and was released in 2014.[5] The IEEE also promulgates a "Software Engineering Code of Ethics".[44]

Employment

There are an estimated 26.9 million professional software engineers in the world as of 2022, up from 21 million in 2016.[45][46]

Many software engineers work as employees or contractors. Software engineers work with businesses, government agencies (civilian or military), and non-profit organizations. Some software engineers work for themselves as freelancers. Some organizations have specialists to perform each of the tasks in the software development process. Other organizations require software engineers to do many or all of them. In large projects, people may specialize in only one role. In small projects, people may fill several or all roles at the same time. Many companies hire interns, often university or college students during a summer break, or externships. Specializations include analysts, architects, developers, testers, technical support, middleware analysts, project managers, software product managers, educators, and researchers.

Most software engineers and programmers work 40 hours a week, but about 15 percent of software engineers and 11 percent of programmers worked more than 50 hours a week in 2008.[47] Potential injuries in these occupations are possible because like other workers who spend long periods sitting in front of a computer terminal typing at a keyboard, engineers and programmers are susceptible to eyestrain, back discomfort, and hand and wrist problems such as carpal tunnel syndrome.[48]

United States

The U. S. Bureau of Labor Statistics (BLS) counted 1,365,500 software developers holding jobs in the U.S. in 2018.[49] Due to its relative newness as a field of study, formal education in software engineering is often taught as part of a computer science curriculum, and many software engineers hold computer science degrees.[50] The BLS estimates from 2014 to 2024 that computer software engineering would increase by 17% .[51] This is down from the 2012 to 2022 BLS estimate of 22% for software engineering.[52][51] And, is further down from their 30% 2010 to 2020 BLS estimate.[53] Due to this trend, job growth may not be as fast as during the last decade, as jobs that would have gone to computer software engineers in the United States would instead be outsourced to computer software engineers in countries such as India and other foreign countries.[54][47] In addition, the BLS Job Outlook for Computer Programmers, the U.S. Bureau of Labor Statistics (BLS) Occupational Outlook predicts a decline of -7 percent from 2016 to 2026, a further decline of -9 percent from 2019 to 2029, a decline of -10 percent from 2021 to 2031.[54] and then a decline of -11 percent from 2022 to 2032.[54] Since computer programming can be done from anywhere in the world, companies sometimes hire programmers in countries where wages are lower.[54][55][56] Furthermore, women in many software fields has also been declining over the years as compared to other engineering fields.[57] Then there is the additional concern that recent advances in Artificial Intelligence might impact the demand for future generations of Software Engineers.[58][59][60][61][62][63][64] However, this trend may change or slow in the future as many current software engineers in the U.S. market flee the profession or age out of the market in the next few decades.[54]

Certification

The Software Engineering Institute offers certifications on specific topics like security, process improvement and software architecture.[65] IBM, Microsoft and other companies also sponsor their own certification examinations. Many IT certification programs are oriented toward specific technologies, and managed by the vendors of these technologies.[66] These certification programs are tailored to the institutions that would employ people who use these technologies.

Broader certification of general software engineering skills is available through various professional societies. As of 2006, the IEEE had certified over 575 software professionals as a Certified Software Development Professional (CSDP).[67] In 2008 they added an entry-level certification known as the Certified Software Development Associate (CSDA).[68] The ACM had a professional certification program in the early 1980s, which was discontinued due to lack of interest. The ACM examined the possibility of professional certification of software engineers in the late 1990s, but eventually decided that such certification was inappropriate for the professional industrial practice of software engineering.[69]

In the U.K. the British Computer Society has developed a legally recognized professional certification called *Chartered IT Professional (CITP)*, available to fully qualified members (*MBCS*). Software engineers may be eligible for membership of the British Computer Society or Institution of Engineering and Technology and so qualify to be considered for Chartered Engineer status through either of those institutions. In Canada the Canadian Information Processing Society has developed a legally recognized professional certification called *Information Systems Professional (ISP)*.[70] In Ontario, Canada, Software Engineers who graduate from a *Canadian Engineering Accreditation Board (CEAB)* accredited program, successfully complete PEO's (*Professional Engineers Ontario*) Professional Practice Examination (PPE) and have at least 48 months of acceptable engineering experience are eligible to be licensed through the *Professional Engineers Ontario* and can become Professional Engineers P.Eng.[71] The PEO does not recognize any online or distance education however; and does not consider Computer Science programs to be equivalent to software engineering programs despite the tremendous overlap between the two. This has sparked controversy and a certification war. It has also held the number of P.Eng holders for the profession exceptionally low. The vast majority of working professionals in the field hold a degree in CS, not SE. Given the difficult certification path for holders of non-SE degrees, most never bother to pursue the license.

Impact of globalization

The initial impact of outsourcing, and the relatively lower cost of international human resources in developing third world countries led to a massive migration of software development activities from corporations in North America and Europe to India and later: China, Russia, and other developing countries. This approach had some flaws, mainly the distance / time zone difference that prevented human interaction between clients and developers and the massive job transfer. This had a negative impact on many aspects of the software engineering profession. For example, some students in the developed world avoid education related to software engineering because of the fear of offshore outsourcing (importing software products or services from other countries) and of being displaced by foreign visa workers.[72] Although statistics do not currently show a threat to software engineering itself; a related career, computer programming does appear to have been affected.[73][74] Nevertheless, the ability to smartly leverage offshore and near-shore resources via the follow-the-sun workflow has improved the overall operational capability of many organizations.[75] When North Americans leave work, Asians are just arriving to work. When Asians are leaving work, Europeans arrive to work. This provides a continuous ability to have human oversight on business-critical processes 24 hours per day, without paying overtime compensation or disrupting a key human resource, sleep patterns.

While global outsourcing has several advantages, global – and generally distributed – development can run into serious difficulties resulting from the distance between developers. This is due to the key elements of this type of distance that have been identified as geographical, temporal, cultural and communication (that includes the use of different languages and dialects of English in different locations).[76] Research has been carried out in the area of global software development over the last 15 years and an extensive body of relevant work published that highlights the benefits and problems associated with the complex activity. As with other aspects of software engineering research is ongoing in this and related areas.

Prizes

There are several prizes in the field of software engineering:[77]

- ACM SIGSOFT Outstanding Research Award, selected for individual(s) who have made "significant and lasting research contributions to the theory or practice of software engineering."[78]
- The Codie awards is a yearly award issued by the Software and Information Industry Association for excellence in software development within the software industry.
- Jolt Awards are awards in the software industry.
- Stevens Award is a software engineering award given in memory of Wayne Stevens.
- Harlan Mills Award for "contributions to the theory and practice of the information sciences, focused on software engineering".

Criticism

Some call for licensing, certification and codified bodies of knowledge as mechanisms for spreading the engineering knowledge and maturing the field.

Some claim that the concept of software engineering is so new that it is rarely understood, and it is widely misinterpreted, including in software engineering textbooks, papers, and among the communities of programmers and crafters.

Some claim that a core issue with software engineering is that its approaches are not empirical enough because a real-world validation of approaches is usually absent, or very limited and hence software engineering is often misinterpreted as feasible only in a "theoretical environment."

Edsger Dijkstra, a founder of many of the concepts in software development today, rejected the idea of "software engineering" up until his death in 2002, arguing that those terms were poor analogies for what he called the "radical novelty" of computer science:

> A number of these phenomena have been bundled under the name "Software Engineering". As economics is known as "The Miserable Science", software engineering should be known as "The Doomed Discipline", doomed because it cannot even approach its goal since its goal is self-contradictory. Software engineering, of course, presents itself as another worthy cause, but that is

eyewash: if you carefully read its literature and analyse what its devotees actually do, you will discover that software engineering has accepted as its charter "How to program if you cannot."[79]

See also

Study and practice

- Computer science
- Data engineering
- Software craftsmanship
- Software development
- Release engineering

Roles

- Programmer
- Systems analyst
- Systems architect

Professional aspects

- Bachelor of Science in Information Technology
- Bachelor of Software Engineering
- List of software engineering conferences
- List of computer science journals (including software engineering journals)
- Software Engineering Institute

References

Citations

1. Abran et al. 2004
2. ACM (2007). "Computing Degrees & Careers" (http://computingcareers.acm.org/?page_id=12). ACM. Retrieved 2010-11-23.
3. Laplante, Phillip (2007). *What Every Engineer Should Know about Software Engineering* (https://books.google.com/books?id=pFHYk0KWAEgC&q=What%20Every%20Engineer%20Should%20Know%20about%20Software%20Engineering.&pg=PA1). Boca Raton: CRC. ISBN 978-0-8493-7228-5. Retrieved 2011-01-21.
4. "Programmers: Stop Calling Yourselves Engineers" (https://www.theatlantic.com/technology/archive/2015/11/programmers-should-not-call-themselves-engineers/414271/). *The Atlantic*. 5 November 2015.
5. Pierre Bourque; Richard E. (Dick) Fairley, eds. (2014). *Guide to the Software Engineering Body of Knowledge Version 3.0 (SWEBOK)* (https://www.computer.org/web/swebok/v3). IEEE Computer Society.

6. "The history of coding and software engineering" (https://www.hackreactor.com/blog/the-history-of-coding-and-software-engineering). *www.hackreactor.com*. Retrieved 2021-05-06.
7. "Computers and Automation: The Computer Directory and Buyers' Guide, 1965" (http://www.bitsavers.org/magazines/Computers_And_Automation/196506.pdf) (PDF). *bitsavers.org*. Retrieved 15 July 2023.
8. Oettinger, A. G. (1966). "President's Letter to the ACM Membership" (https://doi.org/10.1145%2F365758.3291288). *Commun. ACM*. **9** (8). Association for Computing Machinery: 545–546. doi:10.1145/365758.3291288 (https://doi.org/10.1145%2F365758.3291288). ISSN 0001-0782 (https://www.worldcat.org/issn/0001-0782). S2CID 53432801 (https://api.semanticscholar.org/CorpusID:53432801).
9. "The origin of "software engineering" " (https://bertrandmeyer.com/2013/04/04/the-origin-of-software-engineering/). 4 April 2013. Retrieved 17 November 2017.
10. Randall, Brian. "The 1968/69 NATO Software Engineering Reports" (http://homepages.cs.ncl.ac.uk/brian.randell/NATO/NATOReports/). Retrieved 17 November 2017.
11. Software Magazine (5 October 2018). "What to Know About the Scientist who Invented the Term "Software Engineering" " (https://publications.computer.org/software-magazine/2018/06/08/margaret-hamilton-software-engineering-pioneer-apollo-11/). Archived (https://web.archive.org/web/20181124103748/https://publications.computer.org/software-magazine/2018/06/08/margaret-hamilton-software-engineering-pioneer-apollo-11/) from the original on November 24, 2018. Retrieved February 12, 2019.
12. Sommerville, Ian (2016). *Software Engineering* (10 ed.). England: Pearson Education Limited. p. 19. ISBN 978-1-292-09613-1.
13. Peter, Naur; Randell, Brian (7–11 October 1968). *Software Engineering: Report of a conference sponsored by the NATO Science Committee* (http://homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1968.PDF) (PDF). Garmisch, Germany: Scientific Affairs Division, NATO. Retrieved 2008-12-26.
14. Randell, Brian (10 August 2001). "The 1968/69 NATO Software Engineering Reports" (http://homepages.cs.ncl.ac.uk/brian.randell/NATO/NATOReports/index.html). *Brian Randell's University Homepage*. The School of the Computer Sciences, Newcastle University. Retrieved 2008-10-11. "The idea for the first NATO Software Engineering Conference, and in particular that of adopting the then practically unknown term "software engineering" as its (deliberately provocative) title, I believe came originally from Professor Fritz Bauer."
15. 2018 International Conference on Software Engineering celebrating its 40th anniversary, and 50 years of Software engineering (31 May 2018). "ICSE 2018 – Plenary Sessions – Fred Brooks" (https://www.youtube.com/watch?v=StN49re9Nq8&t=67s). *YouTube*. Retrieved 9 August 2018.
16. 2018 International Conference on Software Engineering celebrating its 40th anniversary, and 50 years of Software engineering (31 May 2018). "ICSE 2018 – Plenary Sessions – Margaret Hamilton" (https://www.youtube.com/watch?v=ZbVOF0Uk5lU). *YouTube*. Retrieved 9 August 2018.
17. "The Joint Task Force for Computing Curricula 2005" (http://www.acm.org/education/curric_vols/CC2005-March06Final.pdf) (PDF). 2014-10-21. Archived (https://web.archive.org/web/20141021153204/http://www.acm.org/education/curric_vols/CC2005-March06Final.pdf) (PDF) from the original on 2014-10-21. Retrieved 2020-04-16.
18. *Systems and software engineering – Vocabulary*, ISO/IEC/IEEE std 24765:2010(E), 2010.
19. *IEEE Standard Glossary of Software Engineering Terminology*, IEEE std 610.12-1990, 1990.

20. Sommerville, Ian (2007) [1982]. "1.1.2 What is software engineering?" (http://www.pearsoned.co.uk/HigherEducation/Booksby/Sommerville/). *Software Engineering* (8th ed.). Harlow, England: Pearson Education. p. 7. ISBN 978-0-321-31379-9. "Software engineering is an engineering discipline that is concerned with all aspects of software production from the early stages of system specification to maintaining the system after it has gone into use. In this definition, there are two key phrases:
1. *Engineering discipline* Engineers make things work. They apply theories, methods and tools where these are appropriate [. . .] Engineers also recognize that they must work to organizational and financial constraints. [. . .]
2. *All aspects of software production* Software engineering is not just concerned with the technical processes of software development but also with activities such as software project management and with the development of tools, methods and theories to support software production."
21. "Software Engineering". *Information Processing*. **71**: 530–538.
22. "Definition of SOFTWARE ENGINEERING" (https://www.merriam-webster.com/dictionary/software+engineering). *www.merriam-webster.com*. Retrieved 2019-11-25.
23. Winters, Titus; Manshrec, Tom; Wright, Hyrum (2020). "Preface, Programming Over Time". *Software Engineering at Google*. O'Reilly Media, Inc. pp. xix–xx, 6–7. ISBN 978-1-492-08279-8. "We propose that "software engineering" encompasses not just the act of writing code, but all of the tools and processes an organization uses to build and maintain that code over time. What practices can a software organization introduce that will best keep its code valuable over the long term? How can engineers make a codebase more sustainable and the software engineering discipline itself more rigorous?"
24. Akram I. Salah (2002-04-05). "Engineering an Academic Program in Software Engineering" (http://www.micsymposium.org/mics_2002/SALAH.PDF) (PDF). 35th Annual Midwest Instruction and Computing Symposium. Retrieved 2006-09-13.: "For some, software engineering is just a glorified name for programming. If you are a programmer, you might put 'software engineer' on your business card—never 'programmer' though."
25. Mills, Harlan D., J. R. Newman, and C. B. Engle, Jr., "An Undergraduate Curriculum in Software Engineering," in Deimel, Lionel E. (1990). *Software Engineering Education: SEI Conference 1990, Pittsburgh, Pennsylvania, USA, April 2–3,...* Springer. ISBN 978-0-387-97274-9.,p. 26 (https://books.google.com/books?id=ZuWbyy2blMEC&pg=PA26): "As a practical matter, we regard software engineering as the necessary preparation for the practicing, software development and maintenance professional. The Computer Scientist is preparing for further theoretical studies..."
26. David Budgen; Pearl Brereton; Barbara Kitchenham; Stephen Linkman (2004-12-14). "Realizing Evidence-based Software Engineering" (https://web.archive.org/web/20061217013922/http://evidence.cs.keele.ac.uk/rebse.html). Archived from the original (http://evidence.cs.keele.ac.uk/rebse.html) on 2006-12-17. Retrieved 2006-10-18.: "We believe that software engineering can only advance as an engineering discipline by moving away from its current dependence upon advocacy and analysis,...."
27. Lawrence, Snyder (2017). *Fluency with information technology : skills, concepts, & capabilities* ([Seventh edition] ed.). NY, NY. ISBN 978-0134448725. OCLC 960641978 (https://www.worldcat.org/oclc/960641978).
28. Parnas, David L. (1998). "Software Engineering Programmes are not Computer Science Programmes" (http://citeseer.ist.psu.edu/parnas98software.html). *Annals of Software Engineering*. **6**: 19–37. doi:10.1023/A:1018949113292 (https://doi.org/10.1023%2FA%3A1018949113292). S2CID 35786237 (https://api.semanticscholar.org/CorpusID:35786237)., p. 19: "Rather than treat software engineering as a subfield of computer science, I treat it as an element of the set, {Civil Engineering, Mechanical Engineering, Chemical Engineering, Electrical Engineering,....}."

29. Parnas, David L. (1998). "Software Engineering Programmes are not Computer Science Programmes" (http://citeseer.ist.psu.edu/parnas98software.html). *Annals of Software Engineering*. **6**: 19–37. doi:10.1023/A:1018949113292 (https://doi.org/10.1023%2FA%3A1018949113292). S2CID 35786237 (https://api.semanticscholar.org/CorpusID:35786237)., p. 20: "This paper argues that the introduction of accredited professional programs in software engineering, programmes that are modelled on programmes in traditional engineering disciplines will help to increase both the quality and quantity of graduates who are well prepared, by their education, to develop trustworthy software products."
30. McConnell, Steve (August 2003). *Professional Software Development: Shorter Schedules, Better Projects, Superior Products, Enhanced Careers* (https://archive.org/details/professionalsoft00mcco_0). Boston, MA: Addison-Wesley. ISBN 0-321-19367-9., p. 39: "In my opinion, the answer to that question is clear: Professional software development should be engineering. Is it? No. But should it be? Unquestionably, yes. "
31. Knuth, Donald (1974). "Computer Programming as an Art" (http://disciplinas.lia.ufc.br/matdis061/arquivos/knuth-turingaward.pdf) (PDF). *Communications of the ACM*. **17** (12): 667–673. doi:10.1145/361604.361612 (https://doi.org/10.1145%2F361604.361612). S2CID 207685720 (https://api.semanticscholar.org/CorpusID:207685720).Transcript of the 1974 Turing Award lecture.
32. Dijkstra, Edsger W; transcribed by Mario Béland (November 23, 2004) [First published December 3, 1993]. "There is still a war going on (manuscript Austin, 3 December 1993)" (http://www.cs.utexas.edu/users/EWD/transcriptions/EWD11xx/EWD1165.html). *E. W. Dijkstra Archive*. The University of Texas at Austin, Department of Computer Sciences. Retrieved February 17, 2007. "When the term was coined in 1968 by F.L. Bauer of the Technological University of Munich, I welcomed it. [. . .] I interpreted the introduction of the term "software engineering" as an apt reflection of the fact that the design of software systems was an activity par excellence for the mathematical engineer. [. . .]. As soon the term arrived in the USA, it was relieved of all its technical content. It had to be so for in its original meaning it was totally unacceptable [. . .] In the meantime, software engineering has become an almost empty term, as was nicely demonstrated by Data General who overnight promoted all its programmers to the exalted rank of "software engineer"!"
33. "Software Engineering | Classification of Software Requirements" (https://www.geeksforgeeks.org/software-engineering-classification-of-software-requirements/). *GeeksforGeeks*. 2018-06-19. Retrieved 2021-05-06.
34. "Software Engineering | Software Design Process" (https://www.geeksforgeeks.org/software-engineering-software-design-process/). *GeeksforGeeks*. 2019-05-24. Retrieved 2021-05-06.
35. "What is Software Maintenance? Definition of Software Maintenance, Software Maintenance Meaning" (https://economictimes.indiatimes.com/definition/software-maintenance). *The Economic Times*. Retrieved 2021-05-06.
36. Alain Abran; James W. Moore; Pierre Bourque; Robert Dupuis; Leonard L. Tripp, eds. (2005) [2004]. "Chapter 1: Introduction to the Guide" (https://web.archive.org/web/20160509154355/https://www.computer.org/portal/web/swebok). *Guide to the Software Engineering Body of Knowledge* (http://www.computer.org/portal/web/swebok). IEEE Computer Society. Archived from the original (http://www.computer.org/portal/web/swebok/html/ch1) on 2016-05-09. Retrieved 2010-09-13. "The total volume of cited literature is intended to be suitable for mastery through the completion of an undergraduate education plus four years of experience."
37. "SE2014 Software Engineering Curriculum" (https://www.acm.org/binaries/content/assets/education/se2014.pdf) (PDF). Retrieved 7 April 2023.
38. Cowling, A. J. 1999. The first decade of an undergraduate degree program in software engineering. Ann. Softw. Eng. 6, 1–4 (Apr. 1999), 61–90.

39. "ABET Accredited Engineering Programs" (https://web.archive.org/web/20100619233414/http://abet.org/accrediteac.asp). April 3, 2007. Archived from the original (http://www.abet.org/accrediteac.asp) on June 19, 2010. Retrieved April 3, 2007.
40. McConnell, Steve (July 10, 2003). *Professional Software Development: Shorter Schedules, Higher Quality Products, More Successful Projects, Enhanced Careers*. ISBN 978-0-321-19367-4.
41. Williams, N.S.W. (19–21 February 2001). "Professional Engineers Ontario's approach to licensing software engineering practitioners". *Software Engineering Education and Training, 2001 Proceedings. 14th Conference on*. Charlotte, NC: IEEE. pp. 77–78.
42. "NCEES Software Engineering Exam Specifications" (https://web.archive.org/web/20130827220334/http://cdn1.ncees.co/wp-content/uploads/2012/11/Exam-specifications_PE-Software-Apr-2013.pdf) (PDF). Archived from the original (http://cdn1.ncees.co/wp-content/uploads/2012/11/Exam-specifications_PE-Software-Apr-2013.pdf) (PDF) on 2013-08-27. Retrieved 2012-04-01.
43. "NCEES discontinuing PE Software Engineering exam" (https://ncees.org/ncees-discontinuing-pe-software-engineering-exam/). National Council of Examiners for Engineering and Surveying. 13 March 2018. Retrieved 6 August 2018.
44. "*Software Engineering Code of Ethics*" (http://www.computer.org/cms/Computer.org/Publications/code-of-ethics.pdf) (PDF). Retrieved 2012-03-25.
45. Labs, Qubit (29 November 2022). "How Many Programmers are there in the World and in the US? [2023]" (https://qubit-labs.com/how-many-programmers-in-the-world/). *Qubit Labs*. Retrieved 7 February 2023.
46. "Global Developer Population and Demographic Study 2016 V2" (http://evansdata.com/reports/viewRelease.php?reportID=9). Evans Data Corporation. Retrieved 19 January 2017.
47. Rosenthal, Rachel (August 4, 2020). "Tech Companies Want You to Believe America Has a Skills Gap" (https://www.bloomberg.com/opinion/articles/2020-08-04/big-tech-wants-you-to-believe-america-has-a-skills-gap). *Bloomberg*. Retrieved October 8, 2021.
48. "Computer Software Engineers and Computer Programmers" (http://www.bls.gov/oco/ocos303.htm#training). Retrieved 2009-12-17.
49. "Software Developers" (https://www.bls.gov/ooh/computer-and-information-technology/software-developers.htm). *Occupational Outlook Handbook*. U. S. Bureau of labor Statistics. 4 September 2019. Retrieved 11 December 2019.
50. "Computing Disciplines and Majors" (https://www.acm.org/binaries/content/assets/education/computing-disciplines.pdf) (PDF). ACM. Retrieved 6 September 2019.
51. "Software Developers: Occupational Outlook Handbook" (http://www.bls.gov/ooh/computer-and-information-technology/software-developers.htm). U.S. Bureau of Labor Statistics.
52. "Computer Software Engineer" (https://web.archive.org/web/20130726002354/http://www.bls.gov/k12/computers04.htm). Bureau of Labor Statistics. March 19, 2010. Archived from the original (http://www.bls.gov/k12/computers04.htm) on July 26, 2013. Retrieved July 20, 2012.
53. "Software Developers" (http://www.bls.gov/ooh/computer-and-information-technology/software-developers.htm). Bureau of Labor Statistics. January 8, 2014. Retrieved July 21, 2012.
54. "Computer Programmers : Occupational Outlook Handbook: : U.S. Bureau of Labor Statistics" (https://www.bls.gov/ooh/computer-and-information-technology/computer-programmers.htm). Archived (https://web.archive.org/web/20190503144645/https://www.bls.gov/ooh/computer-and-information-technology/computer-programmers.htm) from the original on 3 May 2019. Retrieved 17 January 2017.
55. "Archive By Publication : Beyond the Numbers: U.S. Bureau of Labor Statistics" (https://www.bls.gov/opub/btn/archive/publication.htm#regional-reports). *www.bls.gov*.

56. "The Soon-to-Be-Extinct Embedded Software Engineer" (https://www.designnews.com/design-hardware-software/soon-be-extinct-embedded-software-engineer). *designnews.com*. May 10, 2018.
57. "hp's Developer Portal | HP International Women's Week: Women in Computer Science dropping since 1980s" (https://developers.hp.com/public/blog/hp-international-womens-week-women-computer-science-dropping-1980s). *developers.hp.com*.
58. "Software engineer jobs in danger due to ChatGPT-like tools? Here's what Google CEO Sundar Pichai has to say" (https://www.businesstoday.in/technology/news/story/software-engineer-jobs-in-danger-due-to-chatgpt-like-tools-heres-what-google-ceo-sundar-pichai-has-to-say-376341-2023-04-06). *Business Today*. 2023-04-06. Retrieved 2023-05-12.
59. "ChatGPT could make these jobs obsolete" (https://nypost.com/2023/01/25/chat-gpt-could-make-these-jobs-obsolete/). 2023-01-25. Retrieved 2023-05-12.
60. Kay, Grace. "Software engineers are panicking about being replaced by AI" (https://www.businessinsider.com/software-engineers-tech-panicking-golden-age-over-chatgpt-ai-blind-2023-4). *Business Insider*. Retrieved 2023-05-12.
61. Fowler, Gary. "Council Post: How Will ChatGPT Affect Jobs?" (https://www.forbes.com/sites/forbesbusinessdevelopmentcouncil/2023/03/16/how-will-chatgpt-affect-jobs/). *Forbes*. Retrieved 2023-05-12.
62. Ito, Aki. "ChatGPT spells the end of coding as we know it" (https://www.businessinsider.com/chatgpt-ai-technology-end-of-coding-software-developers-jobs-2023-4). *Business Insider*. Retrieved 2023-05-12.
63. Zinkula, Aaron Mok, Jacob. "ChatGPT may be coming for our jobs. Here are the 10 roles that AI is most likely to replace" (https://www.businessinsider.com/chatgpt-jobs-at-risk-replacement-artificial-intelligence-ai-labor-trends-2023-02). *Business Insider*. Retrieved 2023-05-12.
64. Cohen, Mikaela (15 April 2023). "These are the tech jobs most threatened by ChatGPT and A.I." (https://www.cnbc.com/2023/04/15/these-are-the-tech-jobs-most-threatened-by-chatgpt-and-ai.html) *CNBC*. Retrieved 2023-05-12.
65. "SEI certification page" (http://www.sei.cmu.edu/certification/). Sei.cmu.edu. Retrieved 2012-03-25.
66. Wyrostek, Warren (March 14, 2008). "The Top 10 Problems with IT Certification in 2008" (http://www.informit.com/articles/article.aspx?p=1180991). *InformIT*. Retrieved 2009-03-03.
67. IEEE Computer Society. "2006 IEEE computer society report to the IFIP General Assembly" (http://www.ifip.org/minutes/GA2006/Tab18b-US-IEEE.pdf) (PDF). Retrieved 2007-04-10.
68. IEEE. "CSDA" (http://www.computer.org/portal/web/certification/csda). Retrieved 2010-04-20.
69. ACM (July 17, 2000). "A Summary of the ACM Position on Software Engineering as a Licensed Engineering Profession" (https://web.archive.org/web/20080517201804/http://www.cs.wm.edu/~coppit/csci690-spring2004/papers/selep_main.pdf) (PDF). Association for Computing Machinery (ACM). Archived from the original (http://www.cs.wm.edu/~coppit/csci690-spring2004/papers/selep_main.pdf) (PDF) on May 17, 2008. Retrieved 2009-03-03. "At its meeting in May 2000, the Council further concluded that the framework of a licensed professional engineer, originally developed for civil engineers, does not match the professional industrial practice of software engineering. Such licensing practices would give false assurances of competence even if the body of knowledge were mature; and would preclude many of the most qualified software engineers from becoming licensed."
70. Canadian Information Processing Society. "I.S.P. Designation" (http://www.cips.ca/standards/isp). Retrieved 2007-03-15.
71. "Professional Engineers Ontario: Welcome to PEO's website" (http://www.peo.on.ca). Peo.on.ca. Retrieved 2012-03-25.

72. Thibodaux, Patrick (2006-05-05). "As outsourcing gathers steam, computer science interest wanes" (http://www.computerworld.com/article/2555175/it-careers/as-outsourcing-gathers-steam--computer-science-interest-wanes.html). Computerworld.com. Retrieved 2016-12-06.
73. "Computer Programmers" (http://www.bls.gov/oco/ocos110.htm#outlook). Bls.gov. Retrieved 2012-03-25.
74. Mullins, Robert (2007-03-13). "Software developer growth slows in North America" (https://web.archive.org/web/20090404033214/http://www.infoworld.com/article/07/03/13/HNslowsoftdev_1.html). *InfoWorld*. Archived from the original (http://www.infoworld.com/article/07/03/13/HNslowsoftdev_1.html) on 2009-04-04. Retrieved 2012-03-25.
75. "Gartner Magic Quadrant" (http://www.cognizant.com/html/content/news/GartnerMQ_Cognizant.pdf) (PDF). Cognizant.com. Retrieved 2012-03-25.
76. Casey, Valentine (2010-08-20). "Virtual software team project management" (http://eprints.dkit.ie/116/1/VCaseyRevisedVersion.doc). *Journal of the Brazilian Computer Society*. **16** (2): 83–96. doi:10.1007/s13173-010-0013-3 (https://doi.org/10.1007%2Fs13173-010-0013-3). S2CID 14383734 (https://api.semanticscholar.org/CorpusID:14383734).
77. Some external links:
    - SIGSOFT Awards (http://www.sigsoft.org/awards/index.htm)
    - ICSE's Most Influential Paper Award (http://www.sigsoft.org/awards/mostInfPapAwd.htm)
    - A list of various Software Engineering (and SE-related) Awards (http://people.engr.ncsu.edu/txie/seawards.html)
78. "Outstanding Research Award" (https://www2.sigsoft.org/awards/outstandingresearch). *SIGSOFT*. Retrieved 1 April 2024.
79. Dijkstra, E. W. (1988). "On the cruelty of really teaching computing science" (http://www.cs.utexas.edu/~EWD/transcriptions/EWD10xx/EWD1036.html). Retrieved 2014-01-10.

Sources

- Pierre Bourque; Richard E. (Dick) Fairley, eds. (2014). *Guide to the Software Engineering Body of Knowledge Version 3.0 (SWEBOK)* (https://www.computer.org/web/swebok/v3). IEEE Computer Society.
- Alain Abran; James W. Moore; Pierre Bourque; Robert Dupuis; Leonard L. Tripp, eds. (2004). *Guide to the Software Engineering Body of Knowledge* (https://web.archive.org/web/20160509154355/https://www.computer.org/portal/web/swebok). IEEE Computer Society. Archived from the original (http://www.computer.org/portal/web/swebok) on 2016-05-09. Retrieved 2010-09-13.
- Sommerville, Ian (2010). *Software Engineering* (https://books.google.com/books?id=PqsWaBkFh1wC) (7th ed.). Pearson Education. ISBN 978-81-7758-530-8. Retrieved 10 January 2013.

Further reading

- Pierre Bourque; Richard E. (Dick) Fairley, eds. (2014). *Guide to the Software Engineering Body of Knowledge Version 3.0 (SWEBOK)* (https://www.computer.org/web/swebok/v3). IEEE Computer Society.
- Pressman, Roger S (2009). *Software Engineering: A Practitioner's Approach* (7th ed.). Boston, Mass: McGraw-Hill. ISBN 978-0-07-337597-7.
- Sommerville, Ian (2010) [2010]. *Software Engineering* (http://www.pearsoned.co.uk/HigherEducation/Booksby/Sommerville/) (9th ed.). Harlow, England: Pearson Education. ISBN 978-

0-13-703515-1.
- Jalote, Pankaj (2005) [1991]. *An Integrated Approach to Software Engineering* (https://www.springer.com/gp/book/9780387208817) (3rd ed.). Springer. ISBN 978-0-387-20881-7.
- Bruegge, Bernd; Dutoit, Allen (2009). *Object-oriented software engineering : using UML, patterns, and Java* (https://archive.org/details/objectorientedso0000brue) (3rd ed.). Prentice Hall. ISBN 978-0-13-606125-0.
- Oshana, Robert (2019-06-21). *Software engineering for embedded systems : methods, practical techniques, and applications* (Second ed.). Kidlington, Oxford, United Kingdom. ISBN 978-0-12-809433-4.

External links

- Pierre Bourque; Richard E. Fairley, eds. (2004). *Guide to the Software Engineering Body of Knowledge Version 3.0 (SWEBOK)*, https://www.computer.org/web/swebok/v3. IEEE Computer Society.
- The Open Systems Engineering and Software Development Life Cycle Framework (http://opensdlc.org/) Archived (https://web.archive.org/web/20100718114646/http://opensdlc.org/) 2010-07-18 at the Wayback Machine OpenSDLC.org the integrated Creative Commons SDLC
- Software Engineering Institute (http://www.sei.cmu.edu/) Carnegie Mellon