ADVANCED SOFTWARE ENGINEERING

1.1 DEFINED: Software engineering is the study and application of engineering to the design, development, and maintenance of software

1.2 History

When the first digital computers appeared in the early 1940s,[4] the instructions to make them operate were wired into the machine. Practitioners quickly realized that this design was not flexible and came up with the "stored program architecture" or von Neumann architecture. Thus the division between "hardware" and "software" began with abstraction being used to deal with the complexity of computing.

Programming languages started to appear in the 1950s and this was also another major step in abstraction. Major languages such as Fortran, ALGOL, and COBOL were released in the late 1950s to deal with scientific, algorithmic, and business problems respectively. E.W. Dijkstra wrote his seminal paper, "Go To Statement Considered Harmful",[5] in 1968 and David Parnas introduced the key concept of modularity and information hiding in 1972[6] to help programmers deal with the ever increasing complexity of software systems.

The term "software engineering" was first used in 1968 as a title for the world's first conference on software engineering, sponsored and facilitated by NATO. The conference was attended by international experts on software who agreed on defining best practices for software grounded in the application of engineering. The result of the conference is a report that defines how software should be developed [i.e., software engineering foundations]. The original report is publicly available.[7]

The discipline of software engineering was created to address poor quality of software, get projects exceeding time and budget under control, and ensure that software is built systematically, rigorously, measurably, on time, on budget, and within specification. Engineering already addresses all these issues, hence the same principles used in engineering can be applied to software. The widespread lack of best practices for software at the time was perceived as a "software crisis".[8][9][10]

Barry W. Boehm documented several key advances to the field in his 1981 book, 'Software Engineering Economics'.[11] These include his Constructive Cost Model

(COCOMO), which relates software development effort for a program, in man-years T, to *source lines of code* (SLOC). $T = k * (SLOC)^{(1+x)}$

The book analyzes sixty-three software projects and concludes the cost of fixing errors escalates as we move the project toward field use. The book also asserts that the key driver of software cost is the capability of the software development team.

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. His 1989 book, Managing the Software Process,[12] asserts that the Software Development Process can and should be controlled, measured, and improved. 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).[13]

Sub-disciplines

Software engineering can be divided into ten subdisciplines. They are:[1]

- Requirements engineering: The elicitation, analysis, specification, and validation of requirements for software.
- Software design: The process of defining the architecture, components, interfaces, and other characteristics of a system or component. It is also defined as the result of that process.
- Software construction: The detailed creation of working, meaningful software through a combination of coding, verification, unit testing, integration testing, and debugging.
- Software testing: The dynamic verification of the behavior of a program on a finite set of test cases, suitably selected from the usually infinite executions domain, against the expected behavior.
- Software maintenance: The totality of activities required to provide cost-effective support to software.

- Software configuration management: The identification of the configuration of a system at distinct points in time for the purpose of systematically controlling changes to the configuration, and maintaining the integrity and traceability of the configuration throughout the system life cycle.
- Software engineering management: The application of management activities—planning, coordinating, measuring, monitoring, controlling, and reporting—to ensure that the development and maintenance of software is systematic, disciplined, and quantified.
- Software engineering process: The definition, implementation, assessment, measurement, management, change, and improvement of the software life cycle process itself.
- Software engineering tools and methods: The computer-based tools that are intended to assist the software life cycle processes (see Computer-aided software engineering) and the methods which impose structure on the software engineering activity with the goal of making the activity systematic and ultimately more likely to be successful.
- Software quality management: The degree to which a set of inherent characteristics fulfills requirements.

Education

Knowledge of computer programming is a prerequisite to becoming a software engineer. In 2004 the IEEE Computer Society produced the SWEBOK, which has been published as ISO/IEC Technical Report 1979:2004, describing the body of knowledge that they recommend to be mastered by a graduate software engineer with four years of experience.[14] 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 CCSE, and updated in 2004.[15] A number of universities have Software Engineering degree programs; as of 2010, there were 244 Campus programs, 70 Online programs, 230 Masters-level programs, 41 Doctorate-level programs, and 69 Certificate-level programs in the United States.[16]

For practitioners who wish to become proficient and recognized as professional software engineers, the IEEE offers two certifications that extend knowledge above level achieved by an academic degree: *Certified Software Development Associate* and *Certified Software Development Professional*.

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.

Profession

Legal requirements for the licensing or certification of professional software engineers vary around the world. In the UK, the British Computer Society licenses software engineers and members of the society can also become Chartered Engineers (CEng), while in some areas of Canada, such as Alberta, Ontario,[17] and Quebec, software engineers can hold the Professional Engineer (P.Eng) designation and/or the Information Systems Professional (I.S.P.) designation. In Canada, there is a legal requirement to have P.Eng when one wants to use the title "engineer" or practice "software engineering".

The United States, starting from 2013 offers an *NCEES Professional Engineer* exam for Software Engineering, thereby allowing Software Engineers to be licensed and recognized.[18] Mandatory licensing is currently still largely debated, and perceived as controversial. In some parts of the US such as Texas, the use of the term Engineer is regulated by law and reserved only for use by individuals who have a Professional Engineer license. The IEEE informs the professional engineer license is not required unless the individual would work for public where health of others could be at risk if the engineer was not fully qualified to required standards by the particular state. Professional engineer licenses are specific to the state which has awarded them, and have to be regularly retaken.

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. Currently, the SWEBOK v3 is being produced and will likely be released in mid-2013.[19] The IEEE also promulgates a "Software Engineering Code of Ethics".[20]

Employment

In 2004, the U. S. Bureau of Labor Statistics counted 760,840 software engineers holding jobs in the U.S.; in the same time period there were some 1.4 million practitioners employed in the U.S. in all other engineering disciplines combined.[21] 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 and have no engineering background whatsoever.[22]

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. Specializations include: in industry (analysts, architects, developers, testers, technical support, middleware analysts, managers) and in academia (educators, 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. Injuries in these occupations are rare. However, like other workers who spend long periods 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.[23]

The field's future looks bright according to Money Magazine and Salary.com, which rated Software Engineer as the best job in the United States in 2006.[24] In 2012, software engineering was again ranked as the best job in the United States, this time by CareerCast.com.[25]

Certification

The Software Engineering Institute offers certifications on specific topics like security, Process improvement and software architecture.[26] Apple, 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.[27] 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).[28] In 2008 they added an entry-level certification known as the Certified Software Development Associate (CSDA).[29] The ACM had a professional certification program in the early 1980s,[*citation needed*] 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.[30]

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 Institution of Engineering and Technology and so qualify for Chartered Engineer status. In Canada the Canadian Information Processing Society has developed a legally recognized professional certification called *Information Systems Professional (ISP)*.[31] 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.[32] 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.[33] Although statistics do not currently show a threat to software engineering itself; a related career, computer programming does appear to have been affected.[34][35] 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.[36] When North Americans are leaving work, Asians are just arriving to work. When Asians are leaving work, Europeans are arriving 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 which have been identified as geographical, temporal, cultural and communication (which includes the use of different languages and dialects of English in different locations).[37] 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 which 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.

Related fields

Software engineering is a direct sub-field of computer science and has some relations with management science. It is also considered a part of overall systems engineering and a direct sub-field of engineering.

Controversy

Over definition

Typical formal definitions of **software engineering** are:

- "the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software".[38]
- "an engineering discipline that is concerned with all aspects of software production"[39]
- "the establishment and use of sound engineering principles in order to economically obtain software that is reliable and works efficiently on real machines"[40]

The term has been used less formally:

- as the informal contemporary term for the broad range of activities that were formerly called computer programming and systems analysis;[41]

- as the broad term for all aspects of the *practice* of computer programming, as opposed to the *theory* of computer programming, which is called computer science;[42]
- 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.[43]

Criticism

Software Engineering sees its practitioners as individuals who follow well-defined engineering approaches to problem-solving. These approaches are specified in various software engineering books and research papers, always with the connotations of predictability, precision, mitigated risk and professionalism. This perspective has led to calls for licensing, certification and codified bodies of knowledge as mechanisms for spreading the engineering knowledge and maturing the field.

Software Craftsmanship has been proposed by a body of software developers as an alternative that emphasizes the coding skills and accountability of the software developers themselves without professionalism or any prescribed curriculum leading to ad-hoc problem-solving (craftsmanship) without engineering (lack of predictability, precision, missing risk mitigation, methods are informal and poorly defined). The Software Craftsmanship Manifesto extends the Agile Software Manifesto[44] and draws a metaphor between modern software development and the apprenticeship model of medieval Europe.

Software engineering extends engineering and draws on the engineering model, i.e. engineering process, engineering project management, engineering requirements, engineering design, engineering construction, and engineering validation. The concept 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.

One of the core issues in software engineering is that its approaches are not enough empirical 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" which is not true because engineering approaches in general can be applied to solve real-world problems more efficiently and effectively than craft. It is the higher education which is failing to promulgate the

true meaning and curriculum of software engineering because the focus of universities is to teach critical thinking and reading / research.

Dijkstra refuted the concepts of "software engineering" and "software maintenance," 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."[45]