What is Software Engineering?

"It has been over thirty-five years since the first organized, formal discussion of software engineering as a discipline took place at the 1968 NATO Conference on Software Engin [Naur 69]. The term 'software engineering' is now widely used in industry, government academia: hundreds of thousands of computing professionals go by the title 'software engineer,' numerous publications, groups and organizations, and professional conferences use the term. However, there are still disagreements and differences of opinion meaning of the term. The following definitions provide several views of the meaning of software engineering. Nevertheless, they all possess a common thread, which states, implies that software engineering is more than just coding – it includes quality, schedule, cost, economics, and the knowledge and application of principles and discipline.

"Over the years, numerous definitions of the discipline of Software Engineering have been presented...

- 'The establishment and use of sound engineering principles (methods) in order for software that is reliable and works on real machine' [Bauer 72]
- 'Software engineering is that form of engineering that applies the principles of science and mathematics to achieving cost-effective solutions to software problems' [IEEE 90]
- 'The application of a systematic, disciplined, quantifiable approach to the develop, operation, and maintenance of software' [IEEE 1990].

"There are aspects of each of these definitions that contribute to the perspective of software engineering. One particularly important aspect is that software engineering builds on science and mathematics. But, in the engineering tradition, it goes beyond this technical draw upon a broader range of disciplines.

"These definitions clearly state that software engineering is about creating high-quality systematic, controlled, and efficient manner. Consequently, there are important emphases on analysis and evaluation, specification, design, and evolution of software. In addition, it issues related to management and quality, to novelty and creativity, to standards, to all skills, and to teamwork and professional practice that play a vital role in software engi

---

Excerpt from [IEEE 04]

References

[IEEE 04]

[Bauer 72]

[CMU 90]

[IEEE 90]
[Naur 69]
Naur, P. and Randell, B. (editors), Software Engineering: Report on a Conference Spon
NATO Science Committee, Oct. 7-11, 1968, Brussels, Scientific Affairs Division, NATO,
Why Software Engineering?
"Since the dawn of computing in the 1940s, the applications and uses of computers ha
a staggering rate. Software plays a central role in almost all aspects of daily life: in go
banking and finance, education, transportation, entertainment, medicine, agriculture, i
number, size, and application domains of computer programs have grown dramatically
result, hundreds of billions are being spent on software development, and the livelihoo
of most people depend on the effectiveness of this development. Software products ha
us to be more efficient and productive. They make us more effective problem solvers, .
provide us with an environment for work and play that is often safer, more flexible, an
confining. Despite these successes, there are serious problems in the cost, timeliness,
many software products. The reasons for these problems are many and include the foll

- Software products are among the most complex of man-made systems, and so
ever nature has intrinsic, essential properties (e.g., complexity, invisibility, and chan
gability) that are not easily addressed [Brooks 95].
- Programming techniques and processes that worked effectively for an individua
team to develop modest-sized programs do not scale-up well to the developme
complex systems (i.e., systems with millions of lines of code, requiring years of hun
dreds of software developers).
- The pace of change in computer and software technology drives the demand for
evolved software products. This situation has created customer expectations an
forces that strain our ability to produce quality of software within acceptable de
schedules."

-Excerpt from [IEEE 04]

References

[IEEE 04]
"Software Engineering 2004: Curriculum Guidelines for Undergraduate Degree Progran
Engineering", The Joint Task force on Computing Curricula, IEEE Computer Society an

[Brooks 95]
Addison-Wesley, 1995.
Software Engineering Course Descriptions

CSI 3471 Software Engineering I
Prerequisite(s): CSI 3334

Modeling, Design, and Testing. An introduction to object-oriented software developer and design. Topics include: iterative development, interpretation of requirements and documents into code; application of design notation in UML; and use of commonly-use-patterns. Introduction to second object-oriented programming language. Laboratory exercises and examples will be used to illustrate and reinforce concepts taught in the lectures.

CSI 3372 Software Engineering II
Prerequisite(s): CSI 3471

Analysis, Design, Project Management. Applying a development/design process to produce quality software. Topics include: identifying user requirements; performing analysis and design process; oriented documentation; using UML notation to create design models diagrams; investigating and applying design patterns, project management, configuration management. Completing software projects applying development processes using an oriented language.

CSI 3373 Software Quality Assurance and Testing
Prerequisite(s): CSI 3372 or concurrent enrollment

Quality, how to assure it and how to verify that it exists; the need for a culture of quality; avoiding errors; inspections and reviews; verification versus validation; testing verification validation techniques; process assurance and product assurance; quality process standards; problem analysis and reporting; statistical approaches to quality control.

CSI 3374 Software Project Management
Prerequisite(s): CSI 3372 or concurrent enrollment

Project planning, cost estimation and scheduling; project management tools; factors in productivity and success; productivity metrics; analysis of options and risks; planning management of expectations; release and configuration management; software process and process implementation; software contracts and intellectual property.

CSI 43C9 Capstone Design Project
Prerequisite(s): Senior standing and CSI 3335, 3372, 4321, and 4337.

A semester-long project course in which students will create a computing system. The requires applying information technology according to established design management including technical presentations (oral and written) by all students.

All Computer Science Course Descriptions
Other Related Required Courses

**ECO 3308 Engineering Economic Analysis**

Designed to provide an economic foundation for engineering decisions. The course stress for making optimum choices among engineering alternatives. Topics include cash flow value analysis; break-even analysis; the impact of taxes and inflation on investment decision methods for assigning costs; the treatment of risk in decision making; and capital budgeting.

**PHY 1425 General Physics I**

Prerequisite(s): MTH 1322 or concurrent enrollment.
Principals and applications of mechanics, wave motion, sound, and heat with emphasis on fundamental concepts, problem solving, notation, and units.

**PHY 2435 General Physics II**

Prerequisite(s): PHY 1408, 1422, or 1425; and MTH1322 or concurrent enrollment.
Principal and applications of electricity, magnetism, light, and modern physics with emphasis on fundamental concepts, problem solving, notation, and units.