Software Engineering: What and Why?

CS 2720

Lecture 1.2
Many software projects fail.

- Consider some statistics summarized from the Standish Group: 29% are successful, 22% are failures, 49% are “challenged”.
- It is not hard to find numerous instances of software failures (see, for instance, 1, 2, 3, 4, 5)
Why So Many Failures?

It is unlikely that the numerous failures are strictly caused by employees who can’t “code”.

- According to proprietary data from QSM, the “typical” software project today involves 11,414 effort hours from 6.9 FTE staff creating 29.6 KESLOC.
- “Coding” by itself does not include all the skills necessary to handle projects like these!
Define *software engineering*.

According to **ISO/IEC/IEEE 24765-2010**:

*Software Engineering:* 1. the systematic application of scientific and technological knowledge, methods, and experience to the design, implementation, testing, and documentation of software.

2. the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software
Breaking Down the Definition

Note some of the items in the ISO/IEC/IEEE definition:

- application of scientific and technological knowledge, methods, and experience
- systematic, disciplined, quantifiable approach
- development, operation, and maintenance
- design, implementation, testing, and documentation
Kung, in *Object-Oriented Software Engineering* (p. 3), lists the objectives of software engineering as:

- “...significantly increasing software productivity (P) ...”
- “...and quality (Q) ...”
- “...while reducing software production and operating costs (C) ...”
- “...and time to market (T).”
How can software engineering help us manage the differences between “personal” and “professional” software development in terms of PQCT?
If we treat a professional software project like a personal project, we can’t succeed.