CSE 112
Advanced Software Engineering

Winter 2015

Prof. Thomas A. Powell

Today

- Class Assumption and Aims
- Overview of the Approach
- Logistics
- Some Intro Ideas
- Amusements

Last time in CSE 112…

- I tend to prefer an applied approach to learning the theory and practices of software engineering.
- Experience is a great teacher and good experience is generally fraught with risk

Peter Venkman

- Our simulated customer
  - First year, Peter the Entrepreneurial Frankenstein with standard teams
  - Second year, Peter take two with two 60+ teams
    - A few words from our Tutors

This time in CSE 112...

- Peter is back

  He has succeed at his start-up and failed at retaining most of his team

    He can’t start over but he has to move forward and fast

    He has some old code someplace, but isn’t sure about it

    He has vowed he won’t let any of this happen ever again

    He demands rigor in his software engineering or else!

This time in CSE 112…

- Fixed tools this time
  - JavaScript only
  - Git (Github or Gitlab)
  - Slack
  - JetBrains Platform – WebStorm, TeamCity, YouTrack, UpSources
  - Heroku

Teams

- Teams
  - Assigned
  - Size: 10-14 will be even at first but may fluctuate for obvious reasons

Grading

- Attendance (2 misses allowed) - 5%
- General Participation / Communication - 5%
- 4 individual exercise grades - 20%
- Team Grade – 20%
    - ~8 team iteration grades
      (stand-up videos, reports, iteration deliveries)
- Midterm - 25%
- Final presentation - 25%

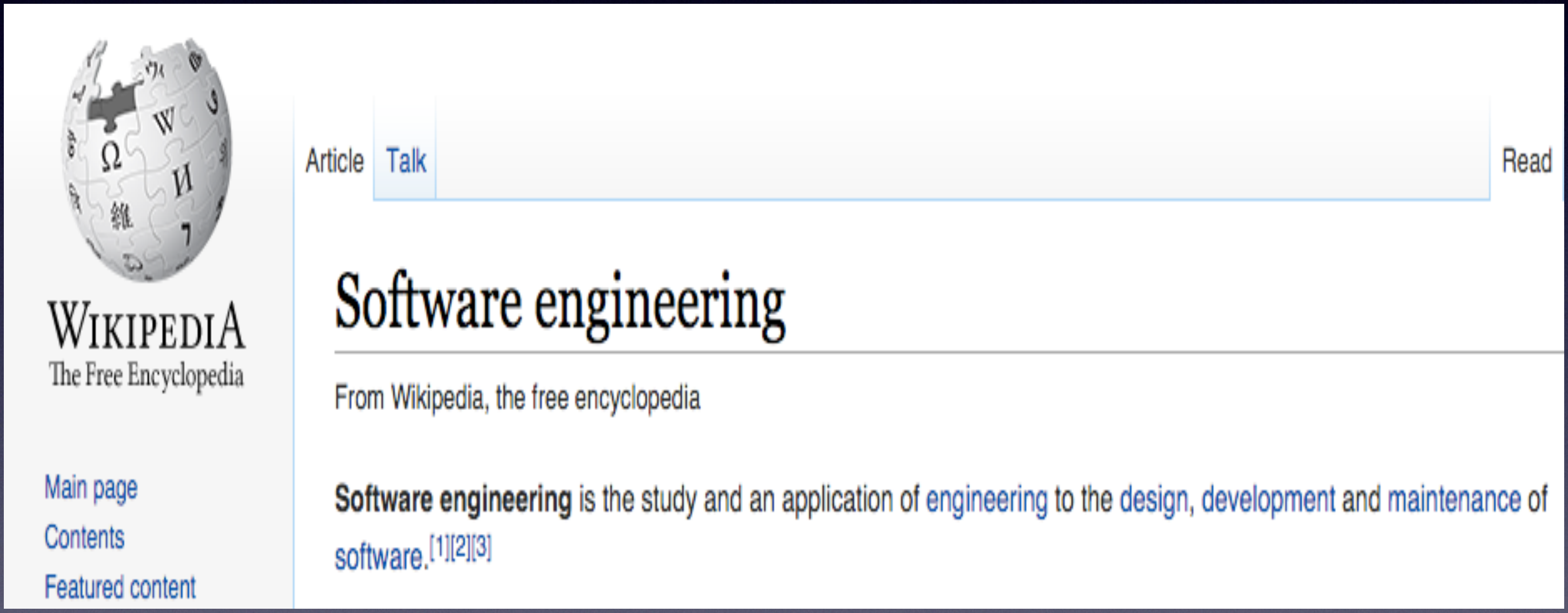
My Assumptions

- You know how to program
- You have taken an intro course in software engineering
- You have a reason you want to be here
  - There is some expected outcome I hope beyond the grade and units if I we both do our jobs right

My Class Goal

To help students gain working practical knowledge of how to produce software in an "engineered" fashion balancing theoretical aspirations with the practicalities imposed by the development environment that will be encountered out in the "real world"

What is Software Engineering?

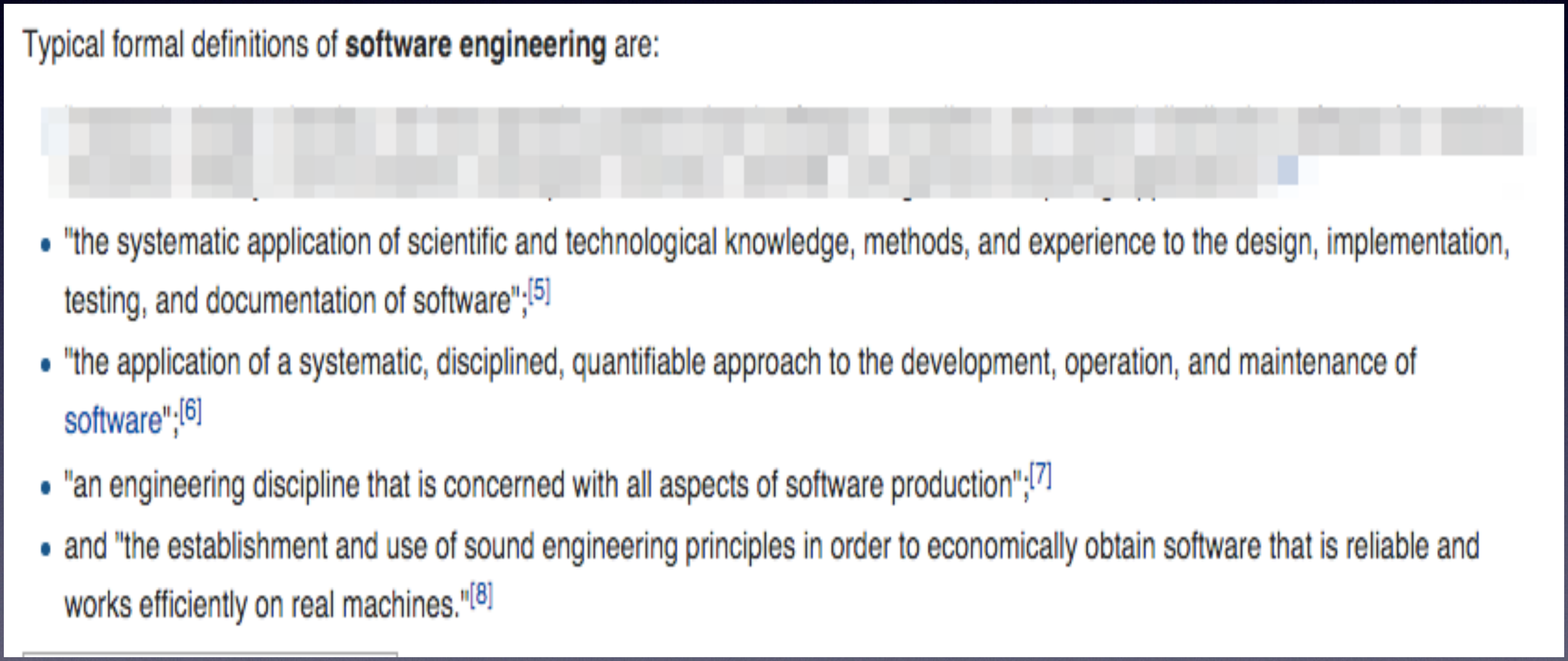


Software engineering

From Wikipedia, the free encyclopedia

Software engineering is the study and an application of engineering to the design, development and maintenance of software.[1][2][3]

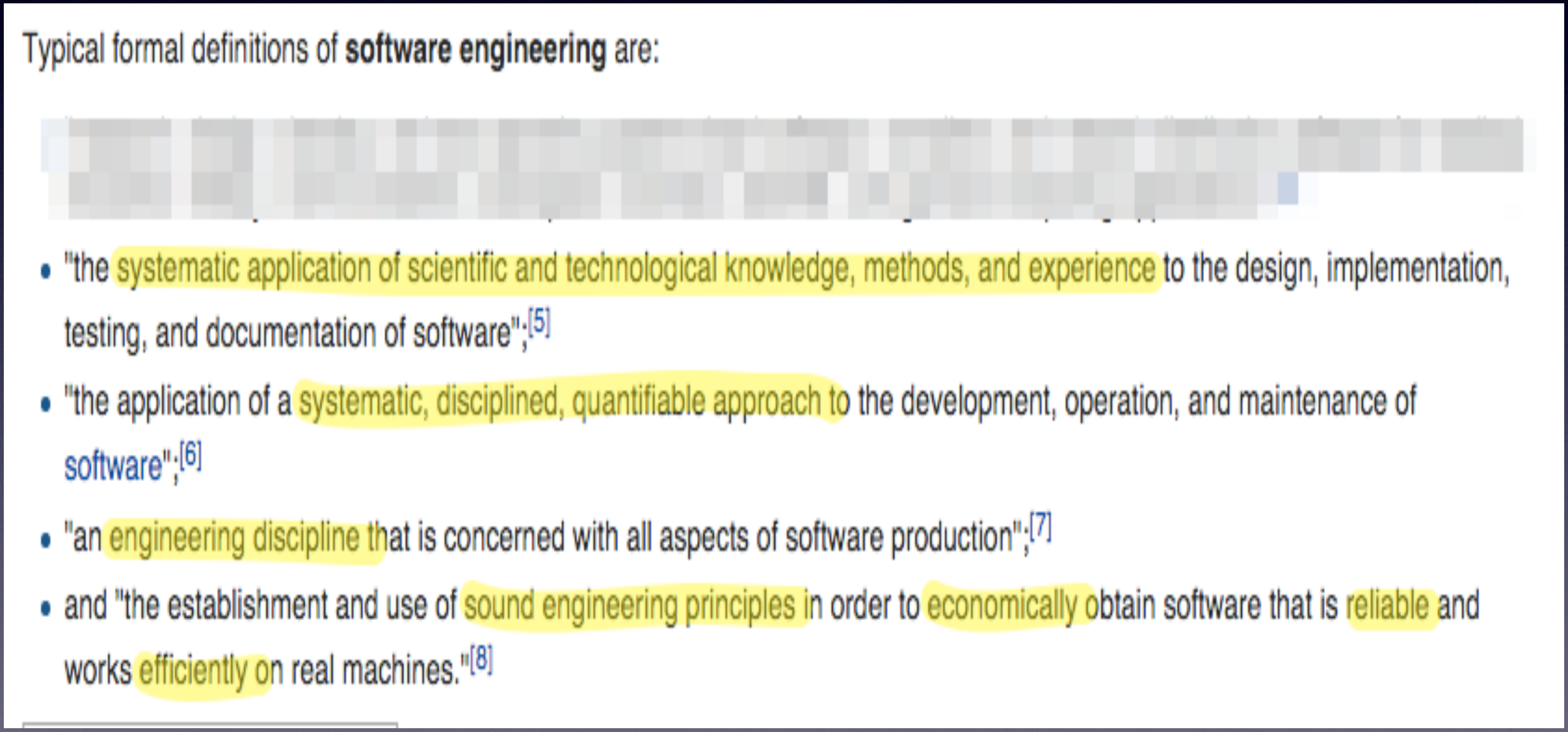
Getting more precise…


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

[redacted]

- "the systematic application of scientific and technological knowledge, methods, and experience to the design, implementation, testing, and documentation of software";[5]
- "the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software";[6]
- "an engineering discipline that is concerned with all aspects of software production";[7]
- and "the establishment and use of sound engineering principles in order to economically obtain software that is reliable and works efficiently on real machines."[8]

Operative Words



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

[redacted]

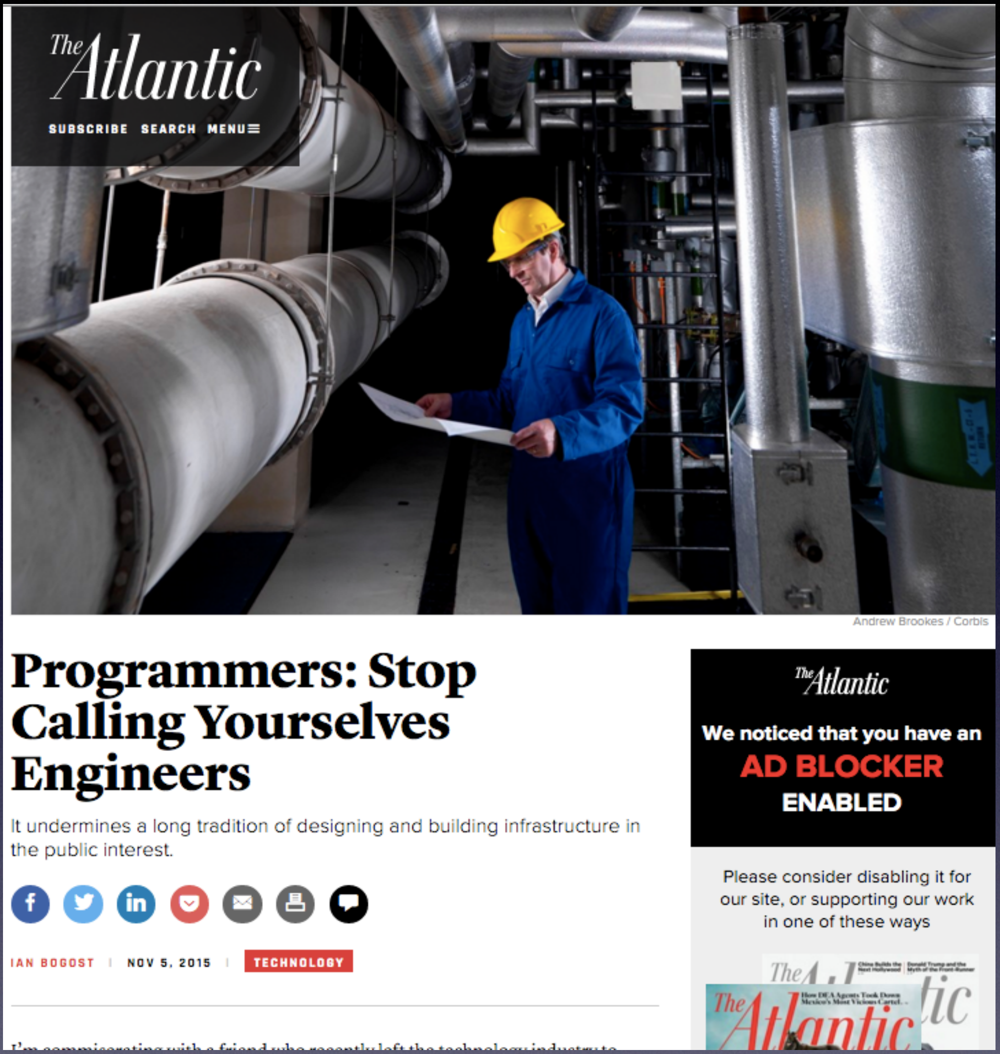
- "the systematic application of scientific and technological knowledge, methods, and experience to the design, implementation, testing, and documentation of software";[5]
- "the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software";[6]
- "an engineering discipline that is concerned with all aspects of software production";[7]
- and "the establishment and use of sound engineering principles in order to economically obtain software that is reliable and works efficiently on real machines."[8]

Engineering!

- *Engineering* is derived from the Latin *ingenium*, meaning "cleverness" and *ingeniare*, meaning "to contrive, devise”

  “The ***creative application of scientific principles to design or develop*** *structures, machines, apparatus, or manufacturing processes, or works utilizing them singly or in combination; or to construct or operate the same with* ***full cognizance*** *of their design; or to* ***forecast their behavior under specific operating conditions****; all as respects an* ***intended function****,* ***economics*** *of operation or* ***safety*** *to life and property”*

What!?



Programmers: Stop Calling Yourselves Engineers

It undermines a long tradition of designing and building infrastructure in the public interest.

IAN BOGOST | NOV 5, 2015 | TECHNOLOGY

The Atlantic
We noticed that you have an AD BLOCKER ENABLED
Please consider disabling it for our site, or supporting our work in one of these ways

http://www.theatlantic.com/technology/archive/2015/11/programmers-should-not-call-themselves-engineers/414271/

Homework #1 – Due Thursday Class Start

First read the Atlantic article

Second … THINK

Third ... write a single page document that indicates what we should do to become software engineers in the true sense of the world. This should include personal and field wise thinking (field wise mans as an industry) attempt.

Homework #1 Format

In the upper corner include a picture (headshot) of yourself and your name with the statement. ”

I __*yourname*___ intend to become a software engineer” to that end I need to:

Present bullets that start with things like “Learn…”, “Always”, ”Aspire to ...

This is not an assignment to debunk the point of the article. Assume they are correct that we aren’t engineers...yet and we want to become one