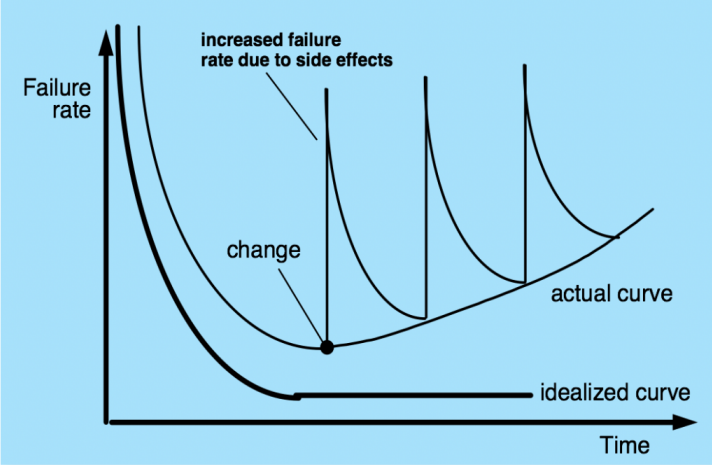
CAS CS 411
Lec 1

SE Defined

1. What is software?
    a. Software is instructions (computer programs) that when executed provide desired features, function, and performance
    b. Data structures that enable the programs to adequately manipulate information
    c. Documentation that describes the operation and use of the programs
    d. Software is developed or engineered to, it is not manufactured in a the classical sense
    e. Software doesn't "wear out" in the sense that material things do, but it can become obsolete
    f. Although the industry is moving towards component-based construction, most software continues to be built by hand, one program at a time
2. Everyone wants to use software
    a. In the modern world, we are obsessed with software, and everyone always wants to be running the latest version – False
3. Nobody wants to use software
    a. The reality is that we just want to do things, not run software
    b. From an engineering standpoint this means that the software must be robust enough to fade into the background
    c. This is the classic feature vs benefit problem
    d. Compare the approaches of
        i. Sony
            1. 
        ii. Apple
            1. 
        iii. They do the same thing (functionality equivalent)
        iv. Software should look like the Apple version (user experience based)

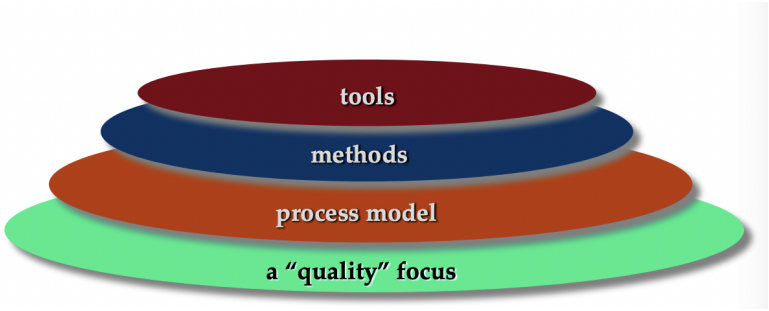
e. Wear vs Deterioration



i.
ii. Ideal: There are many bugs but we catch exceptions so it goes straight
iii. Reality: changes are required (more bugs are introduced)
iv. Opening and closing files can break things
v. Two solutions:
    1. Don't make changes
    2. Make changes after all bugs are fixed
vi. On the idealized curve, we might lose users (release cycle)
vii. On the other hand, if add too much changes, many bugs can be made → lose losers

f. Cloud Computing
    i. Saving data to a third party (extremely convenient but not a great idea)

g. Software Engineering
    i. Some goals
        1. A concerted effort should be made to understand the problem before a software solution is developed
        2. Design becomes a pivotal activity
        3. Software should exhibit high quality
        4. Software should be maintainable
    ii. The seminal defintion:
        1. Establishment and use of sound engineering principles in order to obtain software that is reliable and works efficiently on real machines
    iii. IEEE definition:
        1. The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software
    iv. Software engineering encompasses at least three things
        1. A set of process

2. A set of methods to implement the processes
3. A set of tools to facilitate the methods
4. 

4. Quality
    a. Every text on software engineering calais that 'quality' is a primary goal
    b. What does the word 'quality' mean?
        i. Software
    c. Is quality destroying the economy?
        i. Yes
        ii. Televisions, washers → can work for decades (but new products continue to come out to make users buy the new products)
5. Software isn't the goal
    a. Is a skill
    b. Writing software isn't our core activity
    c. Really we're solving business problems
    d. Software engineering is a discipline
    e. The code is just an expression of our design and decisions
6. A Process Framework
    a. Process framework is a set of guidelines, work products, and tools that attempt to facilitate a process
    b. For software engineering in general, the framework comprises
        i. Communication
        ii. Planning
        iii. Modeling
        iv. Construction
        v. Deployment
7. Process framework
    a. Set of guidelines, work products, and tools that attempt to facilitate a process
    b. For software engineering in general, framework compromises
        i. Communication
        ii. Planning
        iii. Modeling
        iv. Construction
        v. Deployment

8. Umbrella activities
    a. These are applied across all phases of the product
        i. Software project tracking and control
        ii. Risk management
        iii. Software quality assurance
        iv. Technical reviews
        v. Measurement
        vi. Software configuration management
        vii. Reusability management
        viii. Work product preparation and production
9. Polya (How to solve it)
    a. Understand the problem
        i. Who has a stake in the solution to the problem? That is, who are the stakeholders?
        ii. What are the unknowns? What data, functions, and features are required to properly solve the problem?
        iii. Can the problem be compartmentalized? Is it possible to represent smaller problems that may be easier to understand?
        iv. Can the problem be represented graphically? Can an analysis model be created?
    b. Plan the Solution
        i. Have you seen similar problems before? – patterns, existing softwares
        ii. Has a similar problem been solved? – if so, are elements of the solution reusable
        iii. Can subproblems be defined? If so, are solutions readily apparent for he subproblems?
        iv. Can you represent a solution in a manner that leads to effective implementations? Can a design model be created?
    c. Carry out the plan
        i. Does the solution conform to the plan? Is source code traceable to the design model?
        ii. Is each component part of the solution provably correct?
        iii. Is there a process in place to facilitate development?
    d. Examine the results for accuracy
        i. Is it possible to test each component part of the solution?
        ii. Does the solution produce results that conform to the data, functions, and features that are required?
        iii. What lessons can be learned?

10. Hooker's General Principles
    a. The reason it all exists
        i. To provide value to shareholders
    b. KISS (Keep it Simple, stupid)
        i. Designs should be as simple as possible but no simpler
    c. Maintain the Vision
        i. There should be a consistent, overlying architecture
    d. What you produce, others will consume
        i. Those 'others' must be able to understand how to use your product
    e. Be open to the future
        i. Never design for obsolescence… plan for expansion and change
    f. Plan ahead for reuse
        i. Create software building blocks that can be combined to create new products
11. Software isn't the goal
    a. One thing to keep in mind is that, for most companies, software projects are oly tools used to increase profits
    b. As software developers we sometimes lose sight of that
    c. Understanding how projects fit into a company's strategic goals is important in order to gain some context
    d. In this context, failure or success of a project has nearly nothing to do whether the code is correct or passes all of its tests
12. Bottom line
    a. Software is a product designed and built
    b. Typically a professional engineer works on software
    c. Benefits are more important than features
    d. It doesn't make sense for every software product to be created using its own process
    e. What we really want is a single process that we can repeat over and over for each project