8/27

* Process of designing software:
  + Requirements
  + Design (white board)
  + Prototyping (short) (IDE, libraries, compilers, Version control (Git) keeps track of versions)
  + Implementation
  + Testing (cycles back to implementation)
  + Deployment,

8/29

* Software: instructions that provide desired features, functions, and performance. Data structures that enable programs to adequately manipulate information. Documentation that describes the operation and use of the programs.
* Developed or engineered, not manufactured in the classical sense.
* Doesn’t wear out in the typical sense, but gets outdated. Needs to evolve.
* Industry is moving toward component based construction (self contained pieces of software that have their own functionality i.e. libraries, DLLs, etc.), but most is still custom-built
* Procedural: step by step
* OOP: interaction between different entities
* Wear vs. Deterioration
  + Idealized curve is high failure rate at the initial conception, but after time it works perfectly with no bugs
  + Actual curve, high failure at the start, as time goes on and users use it more they discover new errors, also environment changes over time causing other failures.
* Software Applications
  + System software (OS)
  + Application software
  + Engineering/scientific software (Difficult to develop and test, very close attention to detail since used to predict things like weather)
  + Embedded software (i.e. amazon echo)
  + Product-line software
  + WebApps (web applications)
  + AI software
* New categories
  + Open world computing aka the cloud
  + Ubiquitous computing-wireless networks
  + Netsourcing-web as computing engine
  + Open source-”free” source code open to the computing community
  + Also
    - Data mining
    - Grid computing
    - Cognitive machines
    - Software for nanotechnologies
* Legacy software
  + Why must in change?
    - Must be **adapted** to meet the needs of new environment or technology
    - Software must be **enhanced** to implement new business requirements
    - Software must be **extended to make it interoperable** with other more modern systems or databases
    - Software must be **re-architected** to make it viable within a network environment
* Characteristics of webapps
  + Network intensiveness: serves the needs of a diverse community of clients
  + Concurrency: a large # of users must be able to access it at one time
  + Unpredictable load: user base may vary vastly from day to day
  + Performance: must be fast
  + Availability: must be 24/7/365 basis
  + Data driven: use hypermedia to present content to end user
  + Content sensitive: must look pretty
  + Continuous evolution: must evolve fast
  + Immediacy: need to get software to market immediately
  + Security
  + Aesthetics
* Some realities
  + Understand what you are trying to create first
  + Design is a pivotal activity
  + Software should exhibit high quality
  + Software should be maintainable
* Seminal definition: Software engineering is the establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines
* IEEE definition: Software Engineering: (1) systematic, disciplined,
* Layered Approach
  + Top
    - Tools
    - Methods
    - Process model (generic, changes from company to company)
    - Quality focus
  + Base
* Process framework
  + Framework activities
    - Work tasks
    - Work products
    - Milestones & deliverables
    - QA Checkpoints
  + Umbrella activities
    - Project management
    - Formal technical reviews
    - Quality assurance
    - Software configuration management
    - Work product preparation and production
    - Reusability management
    - Measurement
    - Risk management
* Essence of Practice
  + Understand the problem
  + Plan a solution
  + Carry out the plan
  + Examine the result for accuracy
* Understanding the Problem
  + Who has a stake in the solution?
  + What are the unknowns?
  + Can the problem be compartmentalized?
  + Can the problem be represented graphically?
* Plan the solution
  + Have you seen similar problems before?
  + Has a similar problem been solved?
  + Can subproblems be defined?
* Carry out the plan
  + Does the solution conform to the plan?
  + Is each component part of the solution probably correct?
* Hooker’s General Principles
  + 1: The Reason it All Exists
  + 2: KISS
  + 3: Maintain the vision
  + 4: What you produce, others will consume
  + Be open to the future
  + Plan ahead for reuse
  + Think!
* Software Myths

9/5

* Identifying a task set
  + A task set defines the actual work to be done to accomplish the objectives of a software engineering action
    - A list of the tasks to be completed
    - A list of the work products to be produced
    - A list of the quality assurance filters to be applied
* Wicked problem: a social or cultural problem that is difficult or impossible to solve for as many as four reasons: incomplete or contradictory knowledge, the number of people and opinions involved, the large economic burden, and the interconnected nature of these problems with other problems.
* Process flows:
  + Linear
  + Iterative
  + Evolutionary
  + Parallel
* Process patterns:
  + Describes a process related problem that's encountered during software engineering work
  + Identifies the environment in which the problem has been encountered
  + suggest s one or more proven solutions to the problem
* Stated in more general terms, a process pattern provides you with a template--a consistent method for describing problem solutions within the context of the process problems.