Software Engineering Midterm Study Guide

Powerpoints (DONE)

Class 01: None

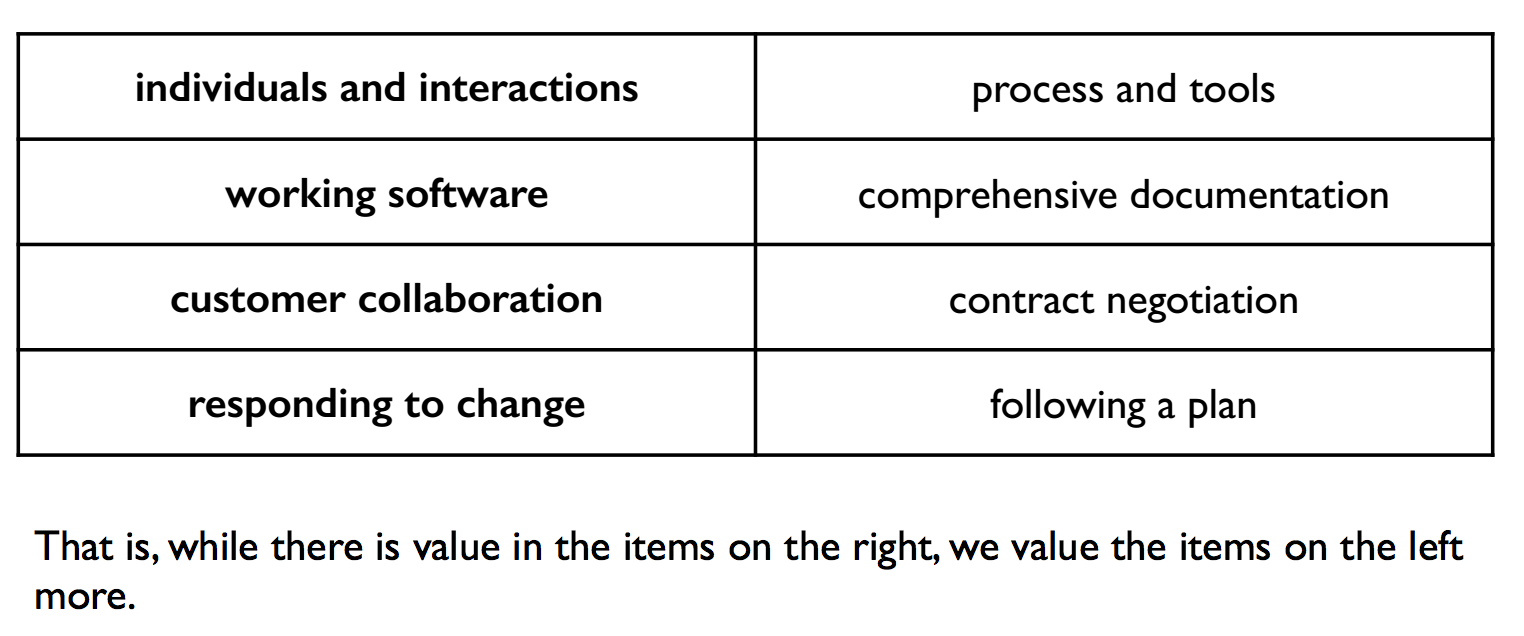
Class 02:

* Software Engineering - “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. 2) The study of approaches as in (1)”
* Science extends our knowledge of laws in nature
* Engineering applies the laws to build useful artifacts
* Doghouse: plan // get supplies // build in a weekend…
  + Programming in the small has individual program // single programmer // does one simple task well // is quickly completed.
* Skyscraper: plan // ??? // profit…
  + Programming in the large has multiple programs // many programmers // many modules // may take years to complete
* REQUIREMENTS // DESIGN // CONSTRUCTION // TESTING

Class 03 (Processes):

* “The process by which user needs are translated into a software product. The process involves translating user needs into software requirements, transforming the software requirements into design, implementing the design in code, testing the code, and sometimes, installing and checking out the software for operational use.”
* Waterfall vs. Incremental
* Waterfall = predictive // Incremental = adaptive
* Waterfall is “A model of the software development process in which the constituent activities, typically a concept phase, requirements phase, design phase, implementation phase, test phase, and installation and checkout phase, are performed in that order, possibly with overlap but with little or no iteration.”
  + Criticisms: We learn a lot as we go, but cannot apply it. 45% of features are never used. Typical software projects experience 25% to 50% change in requirements.
* Incremental is “A software development technique in which requirements definition, design, implementation, and testing occur in an overlapping, iterative (rather than sequential) manner, resulting in incremental completion of the overall software product.”

Class 04 (Agile):

* 
* “Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.”
* “Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.”
* “Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.”
* “Business people and developers must work together daily throughout the project.”
* “Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.”
* “The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.”
* “Working software is the primary measure of progress.”
* “Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.”
* “Continuous attention to technical excellence and good design enhances agility.”
* “Simplicity -- the art of maximizing the amount of work not done -- is essential.”
* “The best architectures, requirements, and designs emerge from self-organizing teams.”
* “At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.”

Class 05 (Happy):

* Iteration planning. Gather high level requirements with info like:
  + Priority (customer sets priority)
  + Time-to-complete (you estimate)

Class 06 (Planning):

* Iteration period is set by us. Maybe 2 to 3 weeks.
* Build plan based on time and priority.
  + High priority first.
  + Place dependent features in the same iteration.

Class 07 (Requirements):

* Software development stakeholders:
  + Users
  + Customer
  + Software engineers
* Requirements elicitation: "the process through which the acquirers (customers or users) and the suppliers (contractor) of a system discover, review, articulate, understand, and document the users' needs and the constraints on the system and the development activity"
  + Ask good questions
  + Blue-sky with customer
  + Role play
  + Observe
  + Heavy customer involvement

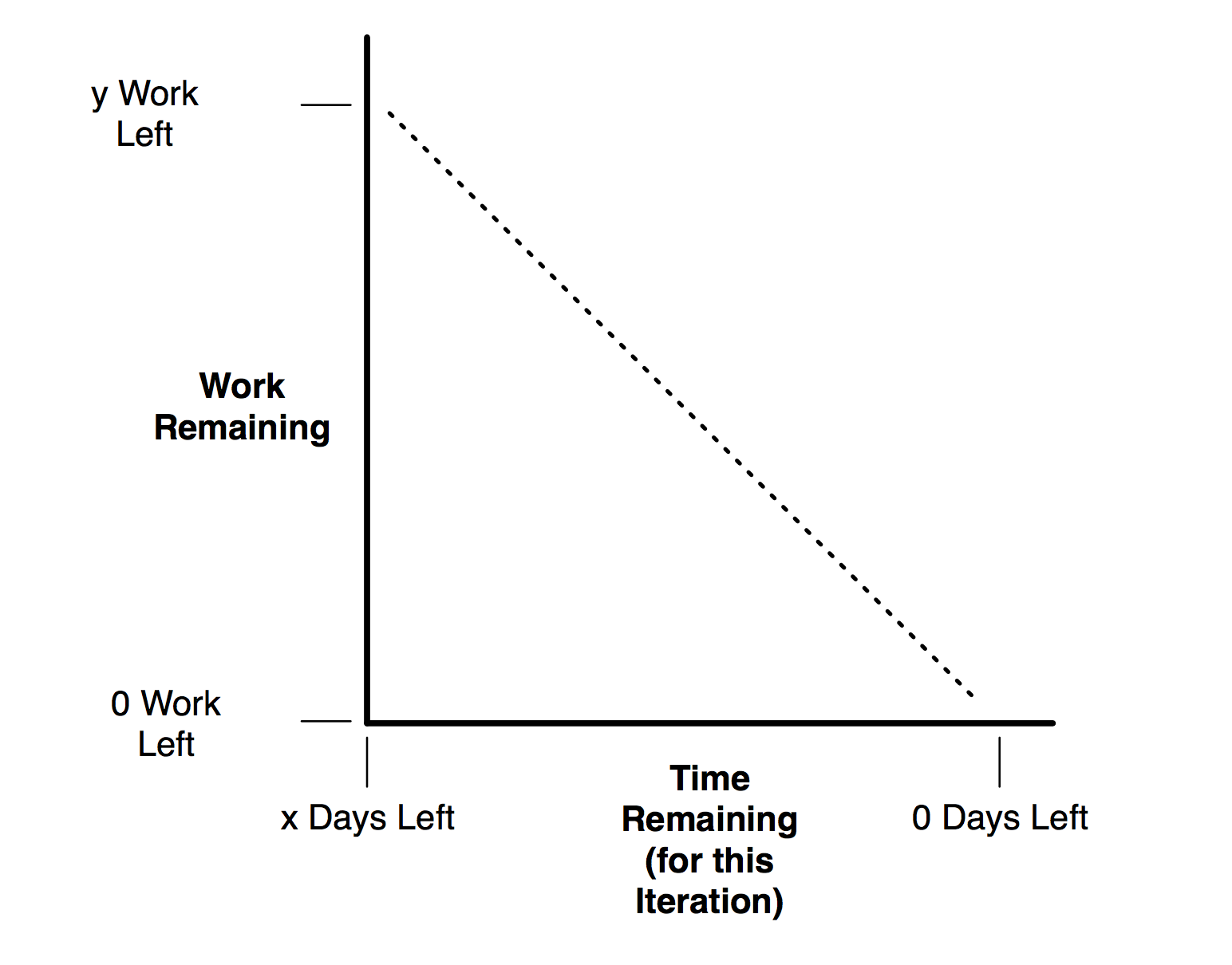
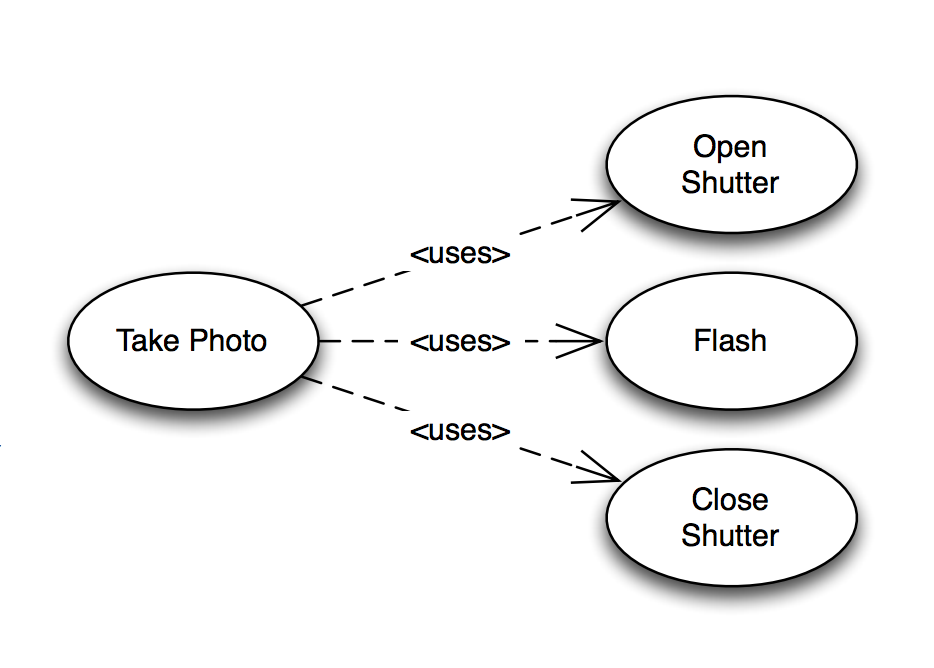
Class 08 (Requirements):

* A requirement should have these attributes:
  + Behavior
    - One thing the software must do. MUST BE VERIFIABLE
    - Written in normal language. Customer can read it.
  + Priority
  + Status
    - Waiting // In-progress // Complete
  + Time estimate
    - Planning poker. All developers vote on time estimate. Come to a consensus. Remove all assumptions.
    - Surviving assumptions are risks.
* Types of requirements:
  + Functional (things the software will execute) vs. Nonfunctional (constrain the solution)
  + Product (behavior) vs. Process (constraints on the development of the software)
  + Quantifiable
* Most important property of a requirement is that it is verifiable!

Class 09 (VCS):

* Git and Mercurial are distributed revision control systems.

Class 10 (Design):

* 
* "The use of scientific principles, technical information, and imagination in the definition of a software system to perform pre-specified functions with maximum economy and efficiency"
* Design: “How will it be accomplished?”
* Input is requirements
* Output is “models and artifacts that record major design decisions.”
* Two steps:
  + Architectural design
  + Detailed design
* Architectural Design - "the process of defining a collection of hardware and software components and their interfaces to establish the framework for the development of a computer system"
* Detailed Design - "the process of refining and expanding the preliminary design of a system or component to the extent that the design is sufficiently complete to be implemented"
* Use Case Diagrams
  + Very abstract
  + Not written by customer
  + Represents functionality
  + Actors // system // use case // relationships
* 
* <uses> X is a subtask of Y

Two Books (HFSWD First 4 // Peopleware first 6) (DONE)

HFSWD 1:

—Most projects have two limitations: time and money

—Big Bang approach to development doesn’t end well. It implies seeing customer at beginning and end of process. Hopefully you got it right!

—Great software development results in **what is needed, on time, and on budget.**

—The secret is **iteration.** Check in with the customer much more often.

—The end of each iteration should be some sort of working software you can show the customer.

—Each iteration is like a little cycle of the Big Bang approach. So each iteration has requirements gathering, design, coding, and testing.

—Plan out an balance iterations based on length of time and priority for features.

HFSWD 2:

—Gather requirements: each one is a single thing that the software has to do.

—Talk with your customer go gather more details and forgotten requirements.

—Blueskying, role playing, and observation

—User stories: describe one thing // written so that the customer understands // written by the customer // be short.

—Think of this process for requirements: getting basic ideas // brainstorm // make user stories // find holes and add clarity // have resulting clear, customer-focused user stories // play planning poker // get any missing information, break up large user stories // have final estimates.

—Estimates are crucial. Get rid of assumptions. Use planning poker.

HFSWD 3:

—Prioritize requirements when time needs to be cut down. **Customer** does this!

—Define iterations. Keep them short, balanced

—Velocity: how much of time working is actually productive time?

—Days of work / Velocity = days required to get work done

—Burn down chart: work left vs. days left.

—Development plan to execute and monitor

Evernote (DONE)

DONE

Old Papers

Devin

Parsing XML (DONE)

Devin