**CS262 Notes**

*Week 1*

* Software engineering = engineering principles and their applications to the development of software systems
* Software characteristics: efficient, understandable, usable, modifiable, correctness
* Software engineer skills: teamwork, communication, planning
* Postman 5 idea: technology change is trade-off, adv/disadv not distributed evenly among population, powerful idea in technology, ecological not additive, media tend to be mythic
* Relative effort phase
  + Maintenance- 67%
  + Implementation- 13%

*Week 2*

* Software development process
  + Life cycle model: category of methodology and software development process refer to specific process chosen by specific organization
* Waterfall: analysis, design, implementation, verification, maintenance
  + Long cycle and high ceremony
* Prototyping: creating prototype; less planning more adaptive process
* Spiral: waterfall and prototyping
* Kanban: manage work
* React vs react native: web vs mobile
* Function vs class component:
* Core component
* native component: backed by same views as android and ios
* Components: independent reusable codes, like JS function but isolation and return html
* Jsx: write element inside javascript
* Html: elements
* Props vs state
* hooks

*Week 3*

* brook’s law= adding people to an already late software development project makes it later
* lean management
  + pull rather than push -> if there is a demand
  + limited work in progress
  + optimize the process
  + visualizing the workflow
* agile software dev
  + discovery and solution improvement through self organizing and cross functional team with user, planning, development, deliver, improvement, flexible response, understand problem to be solve
  + target complex system, product development with dynamic/non-deterministic
    - plans are hard to get/low
    - leap of faith
  + adaptive
    - rolling wave = planning in waves as project proceed and detail become clear
    - changing realities
  + extreme programming = to improve software quality and responsiveness to changing user requirement
    - releases in short dev cycle
  + scrum = teams of 10 or lower and break work to goal to be completed within timed iteration (sprints-2 week to 1 month)/sprint
  + Kanban = schedule system for lean management
  + Pair programming=two people work on 1 thing; coding/reviewing and switch
  + Programming s-b-s= two people work side by side
  + predictive
    - analyze and plan future in detail; entire length of dev process
    - early phase analysis
* What does a version control system do?
  + Track history of changes
* Compare and contrast distributed vs. centralized version control
  + Distributed= all have complete copy of project and history
  + Centralized= one person does pull/push/merge
* Understand the following terms: clone, commit, push, pull
* Manifesto for agile
  + Individuals and interactions
  + Working software
  + Customer collaboration
  + Responding to change

*Quiz 1*

* Software engineering = engineering principles and their applications to the development of software systems
* Brook’s law = adding people to an already late project makes it even later
* Postman’s 5 idea = technology change is trade-off, adv/disadv not distributed evenly among population, powerful idea in technology, ecological not additive, media tend to be mythic
* Waterfall = analysis, design, implementation, verification, maintenance -> long cycle/high ceremony
* Agile software = discovery and solution improvement through self organizing and cross functional team with user, planning, development, deliver, improvement, flexible response, understand problem to be solve + adaptive
* Agile manifesto:
  + Individuals and interactions
  + Working software
  + Customer collaboration
  + Responding to change
* XP = to improve software quality and responsiveness to changing user requirement + short development cycle
* Scrum = teams breaking work to goal to be completed within a timed iteration called sprint (2 week-1 month)
* Kanban = lean method to manage and improve work
* Prototyping = creating prototype; less planning more adaptive process
* lean management:
  + pull rather than push -> if there is a demand
  + limited work in progress
  + optimize the process
  + visualizing the workflow
* Life cycle = category of methodology and software development process refer to specific process chosen by specific organization

*Week 4*

* Functional =system that describe behaviour between input and output
* non-functional = judge the operation of system
  + URPS= usability, reliability, performance, supportability
* unified modeling language= general purpose developmental modeling language
  + structure diagram = static aspect of system; things present in system
  + behaviour diagram = dynamic aspect of system; what happen in system
* use cases (uml)= usage scenario for piece of software + potential scenario where system receive external request and responds to it
  + stick figure (user) -> diagram (use case)
  + fowler use cases
    - title, main success scenario, extension
* user stories= description of feature of software system + pov of end user
* requirements: stakeholder centered, precise, consistent, complete, realistic

*Week 5*

* Software design
  + Booch’s 4 basic principle= abstraction, encapsulation, modularization, hierarchy
  + Information hiding: information contained in a module is inaccessible to other module that have no need for information
* Design pattern= reusable solution to common occurring problem within context
  + Original= Christopher alexander
  + Factory method= interface for creating single object
  + Singleton= class only have 1 instance
  + Adapter= convert class interface to interface client expect
  + Façade= higher level interface that make subsystem easier to use
  + Iterator= way to access element w/o exposing underlying representation
  + Observer= one to many dependency between object where state change in one object result in all dependent notified/updated automatically
* No silver bullet
  + Essential:
    - Complexity, conformity, changeability, invisibility,
  + Accidental:
    - High level language, time-sharing, unified programming environments
  + Thesis: concepts that are the most familiar/used is the most horrible
* Class diagram
* Goal of architecture
* Utilitas= utility
* Venustas= how it looks
* Firmitas= built it right/firm; don’t crash

*Week 6*

* Great minds in development
* Discount usability: simplified user testing, narrowed down prototype, heuristic evaluation
  + 5 people min for testing
* Plan usability test
  + Format/setting of study: lab/field, moderate/unmoderated, person/remote
  + Proxy user not really valuable
  + Type of task: exploratory, specific,
  + Ethical user research: avoid harming ppl, respect ppl dignity, act w/ integrity, maintain confidentiality
  + Difficult scenario:
    - team insist participant continue studying -> take a break
    - team wants you to lie to user -> transparent/truth
    - team wants to rescind payment -> pay participant entire incentive
    - participant boss wants to observe session -> policy to not allow
    - team becomes defensive -> rules for observer etiquette
    - team is distracting during test session -> simple room and rules for participating
    - team jokes about user -> remind member to be professional
* HCI principles
  + Learnable, control, robust
* 4 chunks of information
* Fitt’s law -> big button on corner
* 5 easiest point on desktop to acquire -> 4 corners, 4 edges, button click

*Quiz 2*

* Functional
* Non-functional
* UML= structure and behaviour + use cases,
* User stories
* Booch 4 basic principle + information hiding
* Design pattern
* No silver bullet – essential, accidental, thesis
* 3 Goals of architecture
* Class diagram – domain model and use cases
* Discount usability
* Plan usability test
* HCI principle
* Fitt’s law

*Week 7*

* Database management system: the software that interact with end user, application, database
  + Mysql
  + 4 functions/facilities
    - Data definition: creation, modification, removal of definition that define organization of data
    - Update: insert, modification, delete actual data
    - Retrieval: provide information in a form directly usable by other application
    - Administration: register and monitor user
* Database: organized collection of data stored and access electronically
* E.F codd= outline new approach to database; search facility
* Database before= hierarchical model uses pointer for physical disk address
* External level= how each end user group sees organization of data in database
* Conceptual= unifies external view to compatible global view
* Internal= internal organization of data inside DBMS
* Semantic= meaning model
* Logical data model
* Information system
* Relational model
  + Developed by e.f codd
  + Declarative= specify data -> user state what information database have and what they want. Let database take care of describing and retrieval
  + Relation= table structure with data
  + Tuple= unordered row
  + Column= attribute
  + Field=
  + Null= unknown/missing
  + Integrity= maintenance and assurance of data accuracy and consistency over life cycle
  + Primary=specify a row in table
  + Foreign key= uses key from another table
  + Relational key=
  + Traditional pointer= old database that uses physical disk address

*Week 8*

* SELECT \* FROM Part, PartJob, Job WHERE (Part.ID = partID) AND (jobID = Job.ID) AND (Job.address LIKE ‘%Calvin%’)

*Week 9*

* Object relational mapping= convert data between type systems using object oriented programming language = virtual object database
* Information system domain
  + Impedance mismatch= difficulty when how to match object system to relational database
  + Alternative
    - Native procedural language provided with every major database
* Object oriented= Data and methods, encapsulation, inheritance, polymorphism
  + things
* REST= representation state transfer is software architecture that describe uniform inheritance between physically separated components
  + 4 interface constraint:
    - identification of resource, manipulation of resource, self descriptive message, hypermedia as the engine of application state
* Rest principles
  + Statelessness
    - Communication protocol, no session information retained by server/receiver
  + Cache ability
    - Response cacheable or non-cacheable to prevent client from providing stale/inappropriate data in response to further request
    - Use http protocol
  + Uses standardized URL
  + Uses JSON, XML to transfer data
  + Idempotency= applying method (get, pull, delete) multiple times to a resource result in same state change of the resource as applying them once, though response might differ
  + Client server
    - Separation of concern: user interface concern from data storage concern
* Html, http, url = website
* Data operations:
  + Create-post, read-get, update-put, delete-del
  + Idempotency: get, put, del
  + Nullipotency: same answer regardless of command= get

*Quiz 3*

* Postman 3rd idea= everything has a numerical value
* Database management system
  + 4 functions
* Database
* Codd
* Database before
* External, conceptual, internal
* Relational model
* Declarative, relation tuple, column, field, null, integrity, primary, foreign, relational key, traditional pointer
* Database schema
* Object relational mapping
* Information system domain: impedance mismatch, alternative
* Object oriented
* REST: 4 interface, principle
* Data operation: create, read, update, delete, idempotency, nullipotence
* Polyglot = build, run, scale application in similar manner across all language
* Buildpack= take application, dependencies, runtime, and produce slug
* Slug- bundle of source, runtime, output
* Dyno= isolated container that provide environment required to run application

*Week 10*

* 5 key to nonverbal communication
  + Eye contact, voice fluctuation, position in room, facial expression, hand gesture,
* Powerpoint is evil
* Defense of ppt

*Week 11*

* Software quality
  + Correctness, efficiency, usability, understandability
  + Software quality
    - Functional: building the right product = validation
    - Structural: building the product right = verification
  + Garvin’s 5 quality= transcendental perspective (metaphysical aspect of quality), user perspective (appropriateness of product), manufacturing perspective (quality to requirement), product perspective (quality measured by characteristic of product), value based (different perspective have different importance to stakeholders)
  + Demarco’s definition of software quality= product quality is a function of how much it changes the world for the better
  + Quality and testing relation
* Software testing
  + V & V= verification and validation
  + Purpose of testing= detect software failures so may be discovered and corrected
  + Concept: software fault, programmer error, defect, failure
    - Defect = bug/fault
    - Failure = may or may not from defect
  + Methods: static vs dynamic test, white vs black-box testing
    - Static = review/walkthrough
    - Dynamic = execute code with test case
    - White = verify internal structure of program; api, static, fault injection
    - Black = examine functionality without knowing internal implementation
  + Levels: unit, integration, system, acceptance testing
    - Unit = individual units of source code are tested to determine whether they are fit for use
    - Integration = individual software module is combined and tested as a group
    - System = testing completed system to evaluate system with specified requirement
    - Acceptance = test to determine if requirement of specification are met
  + Types: regression vs acceptance testing, alpha vs beta testing, functional vs non-functional testing, A/B testing
    - Regression = find defect after major code change; seek lost features/bugs that may come back
    - Acceptance = smoke test for build acceptance prior to further testing. User acceptance testing is customer perform test on own device
    - Alpha = simulated/actual operational by potential user/test team
    - Beta =
    - Functional
    - Nonfunctional
    - A/B
  + Mcconnel
    - 10x more

*Week 12*

* CMMI= capability maturity model integration = process level improvement training and appraisal program
* 5 maturity level= initial, managed, defined, quantitatively managed, optimizing
* Process areas
  + Configuration management
    - Establish/maintain integrity of work using configuration identification/control/status/audits
  + Project planning
    - Establish/maintain plan that define project activities
  + Requirement management
    - Manage requirement of project product and product component and ensure alignment between requirement and project plan and work product
  + Risk management
    - Identify potential problem before occur so risk handling activity can e planned and invoked as needed across life of project to mitigate impact/objective
  + Technical solution
    - Select design and implement solution to requirement
* Software metrics
  + Objective vs subjective
  + Direct vs indirect
  + Public vs private
  + Sizing
    - Estimate size of software to be able to implement other software project management activities
  + Software product
    - Defects
    - Mean time between defect
  + Software process
    - runtime
  + Software quality
    - Functional and structural
  + Deming: Most important thing cannot be measured
    - Can’t measure, cant manage
  + Metric can do more harm than good
* Continuous integration
  + Software development practice where team integrate work frequently
  + Integration is long and unpredictable process

*Week 13*

* Technical writing
  + Technical communication
    - Form of communication: about specialized topics, using technology, provide instructions about how to do something
    - Phone/device
  + General communication
    - Communicate through actions/words
  + Audience analysis
  + Accuracy
  + Document layout
* Technical writing style
  + Active vs passive voice
    - Should be = passive voice
  + Present, past, future tense
  + First, second, third person
  + Articles
    - Grammar: the, an, silent h
  + Sentence length
  + Semantic ambiguity
  + Application consistency
  + Action verbs
* Won’t hire people with poor grammar
  + Avoid wiens pet peeves
  + Sentence ending preposition
  + Split infinitive
  + Double negative
* Nielson
  + 3 basic criteria: legibility, readability, comprehension
  + Flesch-kincaid reading score: fry score
    - Length of word and sentence.
    - Longer word = less common
    - Longer sentence = harder to parse
  + Inverted-pyramid writing style
    - Start with conclusion/overview
    - Main points -> supporting info

*Week 14*

* Software evolution
  + Continual development of software after its release to address changing stakeholder/market
* Software maintenance
  + Corrective maintenance= reactive modification of software product after delivery to correct discovered problem
  + Adaptive maintenance= modification of software product after deliver to keep product usable in changed environment
  + Perfective maintenance= modification after deliver to improve performance
  + Preventive maintenance= modification after deliver to detect and correct faults
* Lehman
  + Continuing change
  + Conservation of familiarity
  + Continuing growth
  + Declining quality
* UML- deployment diagrams
  + Structural diagram
  + Nodes= hardware components; database server
  + Arc/artifacts= software components that run on nodes; database
* Fred brooks
  + Never uncertain, always open
  + Open source development
    - early large-scale testing
    - multiple version of components gets built, votes
      * compatibility issue

*Quiz 4*