+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

SDE

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

1. Design Principle Correctness,

Robustness – deal exceptional conditions,

Efficiency, Flexibility, reusability

1. Maintainability -(major work lifetime) correct adapt enhance(re-engineering)

Corrective- remove bugs or defects

Adaptive- due to changes in environment

Regressive – changes after long period

Perfective – Extends its original functionality

1. SDLC – planning-Req analysis(system analyst), Defining, Design, Coding, Testing, Deploy, Maintain
2. Incremental – large project with clear and priority delivers in series of versions

Waterfall – odd, small project, no risk handling, clarity of req at start, Linear or classical life cycle (separate and discrete phase)

Spiral – Each loop (phase-Boehm), radius cost till then, risk management (no project documentation)

Prototype throwaway prototyping, evolutionary – experimental software to understand requirements, risk reduction, requirement elicitation, user interface design

1. RAD – rapid app dev – largest manpower (highly trained) Water Fall Model

Features

Well understood constrained modularized

Construction use 4GL

Multiple teams each has separate function

1. DFD – Data source, flow, Stores, users, decision table trees tools for req analysis, modelling, design(not a logical models)

(Balancing of dfd conservation of inputs and outputs)

Context Diagram- Data source, flow, users, major input outputs, external entities(people, hardware, software, databases), not show functionality

Data Flow Model – (data processing at different levels) store->Process, entity->process, process->process, not in store-> entity

Logical model and good guide to system not representation of system

Object Model

Semantic Model – logical structure of data imported and exported by (ERD)

Service usage Model

1. ER-Diagram – object-relationships, cardinality, modality
2. SRS characteristics (consistent, verified, unambiguous) No redundant
3. Coupling

Normal – without exchange of data or control of info

Stamp

Control

Common – global areas

1. Cohesion

Sequential

Coincidental

Temporal

Communicational

1. Comments –use problem domain, code at crucial places only, document changes to code
2. Testing
   1. White Box – internal programming logic(statement, Path, Condition)
   2. Black Box – Derived from SRS, Functional Approach
   3. Unit testing- along with coding
3. Boundary Value Analysis – approach to designing black box test case

Complementary to class partitioning

Test cases based on boundaries of equivalence classes

1. Control flow graph – Cyclomatic complexity (indicator of structural, max no of independent path, calculated from no of edges and nodes)

Flow chart – separates modules from the softwares

Structure chart-software architectures only one module at top, most one control arrow between two modules, sequence not represented

1. SPMP document –Configuration Management plan, Quality, Risk
2. Float time of activity – difference between the latest finish time and earliest finish time
3. Software risk components – Performance, Cost, Schedule(Identification, Analysis, Prioritization)
4. Risk of Unrealistic estimates and schedules can overcome by – objective estimation, software reuse, multisource estimations
5. SCM
6. Cleanroom Software Development – formal specification, static verification, statistical verification
7. Modular program structure – Architectural, high level, system
8. Graphical tools – data flow, structure, decision
9. Transform analysis – Afferent, Efferent, Central
10. Validation – Are we building right product

Verification – Are we building product right?

1. Stub is dummy version of subordinate module
2. Driver is dummy version of superordinate module
3. Stress testing – beyond maximum design load

Thread testing

Back to back testing – same tests to different versions of system and compare outputs

1. RMMM - project plan
2. Update window() paints , invalid data rect() - WM\_Paint,
3. COCOMO – heuristic estimation technique

COCOMO embedded – software strongly coupled to complex regulations is categorised as

COCOMO complete – large projects as sum of estimates diff subsystems by diff in complexities

Effort estimation – Person-Months

1. PERT/CPM – (dependencies while making projects) scheduling projects, monitoring, optimising no Quality Control
2. Configuration Control – procedure make changes

Configuration Management - One time activity

1. CASE – Computer Aided Software Engineering
2. Cohesion Intra-Module
3. Class diagram – conceptual design
4. Collaboration diagram – organization of Objects
5. OOD abstractions – class
6. Umbrella Activity – QA, Configurations, project monitoring
7. Sequence Diagram – Messages on time scale
8. Quality Control – inspection, testing, removal defects
9. Control abstraction – indicating desired effects without actual mechanism
10. Transform analysis – Structure chart from DFD identifies Afferent, Efferent, Central

CLOUD

1. PAN personal Area Network Bluetooth

LAN local School

MAN Metro Area Net cable

WAN wide Area Net ATM

1. Mesh Wireless
2. App layer hosts
   1. HTTP 80
   2. HTTPS 443
   3. FTP 20,21 File transfer protocol
   4. SFTP 115 simple ftp
   5. DNS 53 Domain Name Service
   6. POP3 Post office protocol
   7. SSH 22 Secure shell
   8. SMTP 25 simple mail transfer
   9. LDAP 389 lightweight
3. Presentation layer data translator

Translation, compression/decompression, encoding/decoding, encrypt/decrypt

1. Session layer
2. Transport layer Data -> Segment (sequence no, checksum, data)

TCP WWW, HTTP connection oriented protocol, slower, guaranteed data, Flow Control error checking

UDP streaming, Connectionless, no Flow Control data delivery, Faster

1. Network layer IP, EGP: exterior Gateway Protocol segment -> Packet (source ip, dest ip)
2. Data Link layer packets -> frame (source MAC, dest MAC)
   1. Logical - uppermost Error control flow control
   2. Media access – frame sync LAN switching
3. MAC – NIC 6bytes(48 bits) 3byte – manufacture 3byte NIC address
4. IPv4 4bytes(32 bit)
5. IPv6 128 bit
6. Virtualization Network, storage, data, desktop, application, hardware – TYPE I bare metal does not need underlying OS

TYPE II need existing OS hosted

1. SAAS – Soft. As a Service (apps, Facebook, YouTube) host apps

PAAS – Plat (S3, app engine) platform to develop, run, manage

IAAS – (EC2, VM) provide resources could use any service want

1. EC2 resizable compute capacity create instance(images)

AMI (amazon machine image) – specific region, template for volume (instance, EBS)

Instance type – General Purpose – A, T, M

Compute – C

Memory – R, X, Z

Storage – I, D, H

Storage – Block (instance, EBS), Object(S3), File Sharing(EFS)

SSD

EBS general Purpose (gp2) 100-3000IOPS, 32000IOPS, 1GiB – 16 Gibs

Provisional IOPS (io1) 4GiB – 16 GiB

HDD

Throughput Optimized (st1) 500MiB/s

Cold HDD (sc1)

Security Groups – virtual firewall for control inbound and outbound traffic (instance level not subnet level)

KEYPAIR .pem format (privacy enhanced email)

DEVOPS

1. Managing and tracking code – SCM tools (git)

Incremental test, deploy – Jenkins (CD/CI)

Manual testing – Selenium

Scalability – Puppet

Dependencies – Docker

1. Never –ending process of continuous improvement

Reasons – Predictability, reproducibility, maintainability, time to market, greater quality, reduced risk, resiliency (stable, secure, auditable)

1. Plan

Code – IDES, SCM,

Build Maven

Test

Release

Deploy

Operate, Monitor

Git

1. Version Control System
2. Git – One of the distributed version control systems every commit save
3. Commands

git config --global user.name

git config –global user.email

git init

git add .

git commit –m “message”

git diff

git mv source dest

git rm <item>

git checkout file –rm unwanted changes

git branch - create branch

git checkout <name>

git merge <name>

git branch –d <name>

git remote add <name> <url>

git clone <url>

git push <name> <branch>

git pull

1. Repository .git
2. Object key-value
3. Blobs (binary Large Object no metadata) sha1
4. Clone create instance of repository, checkout and mirror repo
5. Pull copies changes to local sync
6. Push local to remote
7. Head committed snapshot .git/refs/heads/
8. Commit
9. Branches another line of development
10. Hosting service for version control using git

Docker

1. Containers – share OS among apps like object of class
2. Containerization – lightweight alternative to VM
3. Packaging software code and all its dependencies so that it will run on any platform process-level isolation
4. Images OS file systems and apps like class of object
5. Commands

Docker images ls

Docker image pull <name>

Docker image inspect <name>

Docker image rm <name>

Docker image push <name>

Docker image tag <name> <tag>

Docker image build <dockerFile>

Docker container ls list all running

Docker container ls –a list all including stopped

Docker container run <name>

Docker container start <name>

Docker container stop <name>

Docker container rm <name>

Docker container stats <name>

Docker container exec <name> execute

Docker container commit <name> create image of current state

Docker container top <name> running process

Docker-compose up - run

Docker-compose stop

Docker-compose down

1. Compose – tool for defining and running multi-container

docker-compose.yaml (data serialization lang manage data and Unicode char)

One direction processing case sensitive .yaml or .yml no tabs comments with #

1. Mapping - Key: value, -reading

Sequence – list of values separate line using dash and space

1. Docker swarm is container orchestration running on hosts let them use together

Node is instance of docker engine participating submit service to manager node – dispatches work called tasks, also perform orchestatation

1. Worker node – receive and execute task dispatched from manager node

Service – tasks to execute on the manager central system and primary root to user

Task – container and the commands to run inside container atomic scheduling unit of swarm run on the assigned node

1. Docker swarm init –advertise-addr <ip>

Docker info

Docker node ls

Docker swarm join-token worker

Docker swarm join –token <token>

Jenkins

1. Continuous Integrations – automating building and testing code
2. Continuous Delivery
3. Continuous Deployment
4. Jenkins CI. CD. DSL (domain specific language )
5. Node mean any system that can run jobs might be container

Master primary controlling system complete access default locations

Agent Jenkins slave non master

Executor run a job on a node can have zero or more executors

1. Jenkins Pipeline – engine which supports a number of auto pattern

Adds a powerful set of tools use cases that span

Support implementation and integration getting software from version control to user and customers

Code, Durable, Pausable, Versatile, Extensible

Pipeline entire process include stages for building application

Kubernates

1. For managing workloads and services facilitates configuration and automation
2. Service Discovery and load balancing - load balance and distribute network traffic

Storage orch

Automated rollouts and rollbacks – desired state for your container

Automated bin packing – how much CPU Ram needs, best use of resources

Self – healing – container that fails replaces automatically

Secret and config management –

1. Cluster – has one worker and one master
2. Master components

Kube – apiserver – expose kubernates API

Etcd – consistent and highly-avaliable key value store used as backing store for all cluster

Kube-scheduler master that newly created pods that have no code

Kube-Controller-manger – runs controller each separate process but reduce complexity run in signal process

Node = noticing and responding when nodes go down

Replication = maintaining correct pods

EndPoints = poplulates endpoints

Service = create default account and APIS

1. Node Components

Kubelnet – runs on each node running on pod

Kube-proxy – maintains network rules pods from session inside or outside cluster

Container runtime – running containers