1, Software configuration management

configuration management refers to the process by which all artifacts relevant to your project and relationships between them, are stored, retrieved, uniquely identified, and modified using Version control, managing dependencies managing software configuration

Managing software configuration

1, If we are in an application lifecycle it makes sense to inject a particular piece of configuration at the point of deployment or installation time, at start up time, or at run time

2, keep the available configuration options for your application in the same repository as it's source code configuration settings have a lifecycle completely different from that of code while sensitive information should not be checked into Version control at all

3 use clear naming for your config options should be easly understandable by everyone

4 ensure that your configuration information is modular and encapsulated so that changes in one place don't have knock on effects for other apparently unrelated, pieces of configuration

5 be minimalist - keep configuration as simple as possible

6- ensure that you have tests for your configuration that are run at deployment or installation time check the services ~~factors~~ in your s/w for availability

~~Version control systems, also known as Source~~ Contd

2, Distributed version control system

in Github model contributions are made by first forking the repository of the project you wish to contribute to making your changes and then asking for the owners of original repository to pull your changes

3 Test Doubles

- replacing parts of a system at run time with a simulated version
- A test double is an object that can stand in for a real object in a test
- the most common types of test doubles are stubs mocks and fakes
- meszaros coined the general term test doubles and distinguished various types of test doubles

4 Deploying pipeline

A deployment pipeline is an automated management of process for getting software from version control into the hands of users
this process involves building the software followed by the progress of these builds through multiple stages of testing and deployment

5 principles of software delevery

, Create a repeatable, reliable process of releasing software
Releasing software should be easy. If should be as simple as pressing a button
the reliability & repeatability derive from two principles

• Automate almost everything

Automation is a prerequisite for the deployment pipeline because it is only through automation that we can guarantee that people will get what they need at push of a button

• Keep everything in version control

Everything you need to build deploy test & release your application should be kept in some format versioned storage

2, if it hurts do it more frequently & bring the pain forward

if releasing software is painful, aim to release it every time somebody checks in a change that passes all automated tests

if you can't release it to real users upon every change, release it to production like environment upon every check in

o Build quality in

The earlier you catch defects the cheaper you can fix it

3, Done means released

A functionality is "done" once it has been successfully demonstrated to and tried by representatives of the user community, from a production like environment

5 everybody is responsible for the delivery process

i) Have a system where everyone can see at a glance the status of the application, it's health, it's various builds on tests they have passed

Scanned with CamScanner

6, continuous improvement

The first release of an application is just the first stage in it's life

all applications evolve and more releases will follow It is important that your delivery process also evolves with it

6, commit stage

The commit stage begins with a commit to the version control system

it ends with either a report failure or at successful

The commit stage is also the point at which you should begin the construction of your deployment pipeline

the commit stage tests the following

covering features that can be deployed into the

performing any analysis necessary to calculate

validate the code

creating any other artifacts that will be used in any other stage of a deployment pipeline