Chapter-2

Date
Page

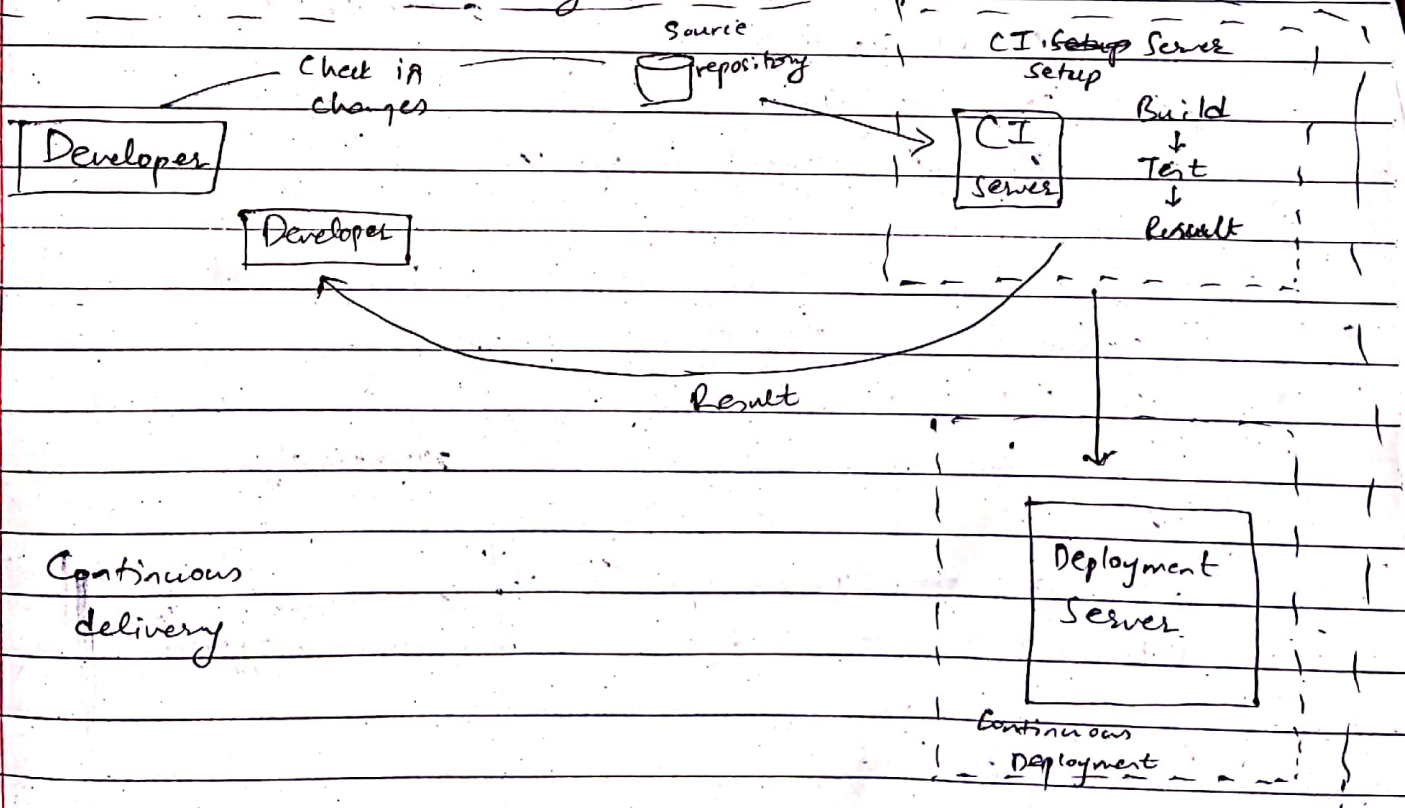
Development Process Management

2.2 # Continuous Integration :

→ Continuous Integration (CI) is a software development practice in which developers merge their change to the main branch many time.

→ Each merge triggers an automated code build and test sequence.

→ A successful CI build may lead to further stages of Continuous Delivery (CD).



Date
Page

CD (Continuous deployment)

→ It is an extension to CI where code changes are automatically prepared in the form of build object ready for deployment / development.

→ In Continuous Integration :

(i) developer
(ii) Source code
(iii) Build and test

automatically

→ In Continuous deployment :

(i) developer
(ii) Source code
(iii) Build and test
(iv) the test environment
(v) production deploy

automatically

→ In continuous delivery

(i) developer
(ii) Source code
(iii) Build and test
(iv) ~~the env test~~ environment
(v) & production deploy

automatically

auto deploy but after manual approval.

→ Benefits of CI :

① Easy and quicker integration.
② Catch issues or error early and solve them easily.
③ Spend less time debugging and more time adding feature.

Date ____
Page ____

(4) Avoids confusion : Stop waiting to find out if the code ~~can~~ works.

(~~For~~ more look note ~~in~~ physical note)

2.1# Source Code Management : (SCM).

→ Source code Management are the tools that helps team collaborate and modify the source code repository (repo) of their projects.

→ What it is ~~NOT~~ :-

1. Multiple people can work in same code.
2. Old versions files can be easily retrived.
3. Keeping logs ~~of~~ about what changed in a file.
4. Integrating changed code.
5. Generating released builds.

→ What it is NOT :-

1. A replacement for communication.
2. A replacement for management.
3. ~~3~~

Version Control System (VCS) :-

→ A VSC is a tool for managing a collection of program code that provides us ~~you~~ with three important capabilities : reversibility, concurrency and annotation (PL comments).

→ It is synonymously also known as Source code Management ~~Code~~ (SCM).

→ { Same terms}

(Work Cycle of VCS)

→ How do we normally work without VCS?

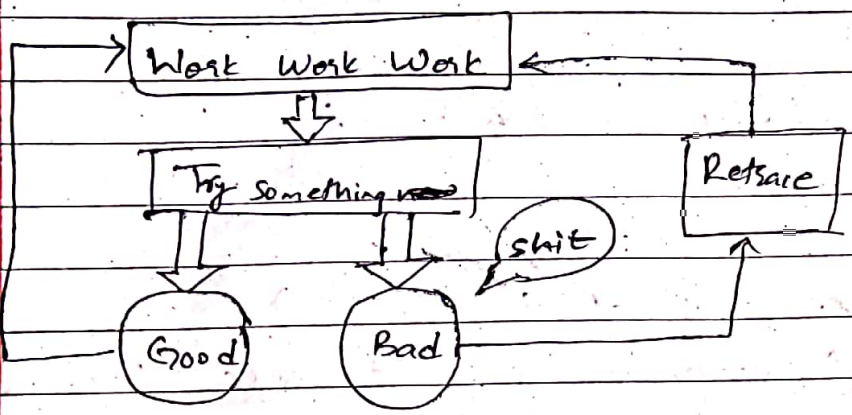


fig:- Work cycle without VCS

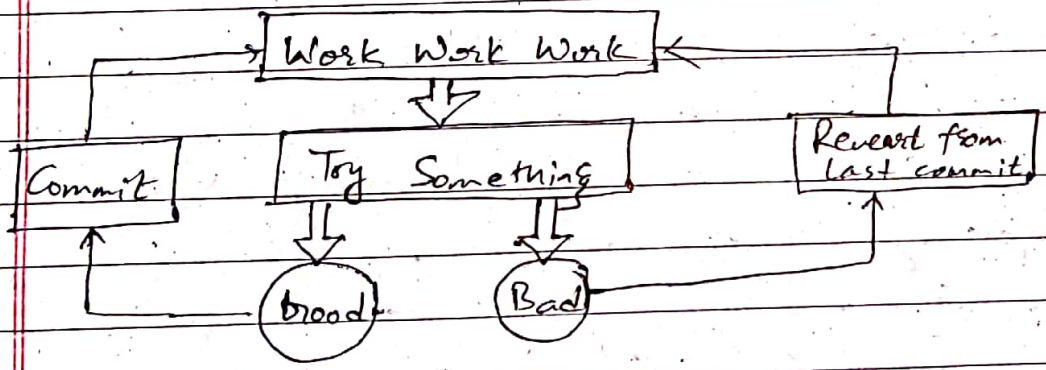


fig:- Work cycle with VCS.

→ Things that don't change should be archived, not versioned.
For eg:- an email once posted, does not change therefore, versioning it wouldn't make sense.

→ Basic terms:-

(1) Repository
(2) Commit
(3) Log messages
(4) Update
(5) Checkout
(6) working copy
(7) branch
(8) merge
(9) conflict
(10) lock

(See in physical note)

Date
Page

→ Version Control tools :-

There are many Version Control tools :-

1. Project Revision Control System (PRCS)
2. Source Code Control System (SCCS)
3. Revision Control System (RCS)
4. Concurrent Version System (CVS)

(1) Project Revision Control System (PRCS) :-

→ PRCS is a front end to a set of tools that (like CVS) provides a way to deal with sets of files and directories as an entity.

→ It is similar to that of SCCS, RCS, CVS but it is much simpler than any of these systems.

(2) Source Code Control System (SCCS) :-

→ SCCS lacks some of the features of RCS but it is generally equivalent systems and has a few capabilities that RCS does not.

(3) Revision Control System (RCS) :-

→ RCS allows people working on the control system for multiple revision of text i.e, revised frequently such as documentation or programs.

(4) Concurrent Version System (CVS)

→ CVS is an open source version control system designed to manage entire software projects.

→ It is an important component of SCM.

{Workflow & build term same as SCM or VCS :}

→ Types of VCS :-

(1) Centralized VCS

(2) Distributed / Decentralized VCS

• Difference :-

Centralized VCS	Distributed / Decentralized VCS
(1) Centralized VCS is a version control in which central repository of the server provides latest codes to the client machines	(1) Decentralized / Distributed VCS is a version control in which complete codebase is mirrored on every developers computer.
(2) There are no local repository	(2) There are local repository.
(3) Works comparatively slower	(3) Works faster.
(4) Always requires internet connectivity	(4) Developers can work with a local repository without any internet connection.
(5) Considers entire columns for compressions	(5) Considers column as well as partial columns
(6) Focuses on synchronizing, tracking and backing up files	(6) Focus on sharing changing.

Date
Page

(f)	A failure in the central Server terminates all the Versions Diagram	7	A failure in the main Server does not affect the development Diagram

To move a change from one branch to another. This includes merging from the main trunk to some other branch, or vice versa. In fact, those are the most common kinds of merges; it is rare to port a change between two non-main branches. "Merge" has a second, related meaning: it is what the version control system does when it sees that two people have changed the same file but in non-overlapping ways. Since the two changes do not interfere with each other, when one of the people updates their copy of the file (already containing their own changes), the other person's changes will be automatically merged in. This is very common, especially on projects where multiple people are hacking on the same code. When two different changes do overlap, the result is a "conflict".

Conflict

What happens when two people try to make different changes to the same place in the code. All version control systems automatically detect conflicts, and notify at least one of the humans involved that their changes conflict with someone else's. It is then up to that human to resolve the conflict, and to communicate that resolution to the version control system.

lock

A way to declare an exclusive intent to change a particular file or directory. Not all version control systems even offer the ability to lock, and of those that do, not all require the locking feature to be used. This is because parallel, simultaneous development is the norm, and locking people out of files is (usually) contrary to this ideal.

Version control systems that require locking to make commits are said to use the lock-modify-unlock model. Those that do not are said to use the copy-modify-merge model.

Types of VCS: Centralized vs Distributed

Currently, the most popular version control system in use is Subversion, which is considered a centralized version control system. The main concept of a centralized system is that it works in a client and server relationship. The repository is located in one place and provides access to many clients. It's very similar to FTP in where you have an FTP client which connects to an FTP server. All changes, users, commits and information must be sent and received from this central repository.

The primary benefits of Subversion are:

- It is easy to understand and get started.
- You have more control over users and access (since it is served from one place).
- More GUI & IDE clients (Subversion has been around longer).

The main drawbacks of Subversion are:

- Dependent on access to the server.
- Hard to manage a server and backups.
- It can be slower because every command connects to the server.
- Branching and merging tools are difficult to use.

Decentralized/Distributed systems are a newer alternative for traditional centralized VCS. In distributed version control, each user has their own copy of the entire repository, not just the files but the history as well. Think of it as a network of individual repositories. The most popular VCS in this category is Git.

The primary benefits are:

- It is fast. More powerful and detailed change tracking, which means less conflicts.
- No server necessary – all actions except sharing repositories are local (commit offline).
- Branching and merging is more reliable, and therefore used more often.

Some drawbacks:

- The distributed model is harder to understand and not much GUI clients (as it is new).
- The revisions are not incremental numbers, which make them harder to reference.
- It can be easier to make mistakes until you are familiar with the model.

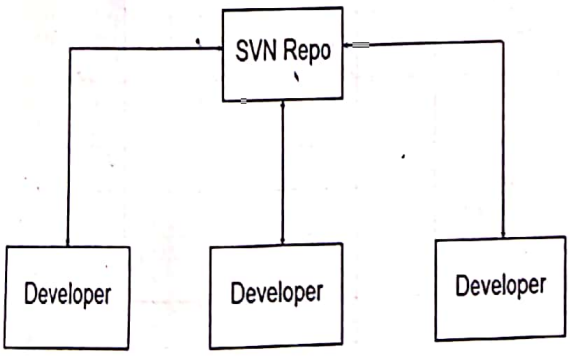


Fig: Centralized VCS

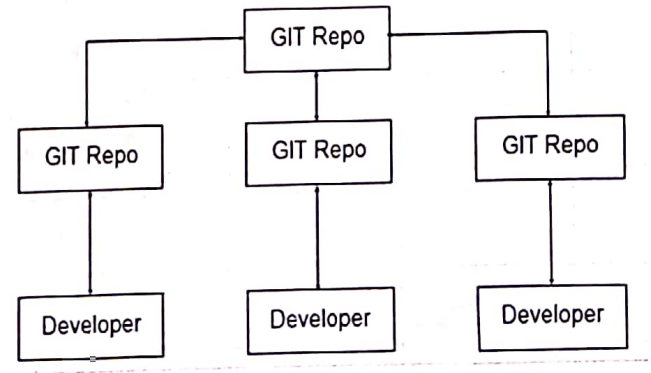


Fig: Decentralized VCS

References:

http://www.slideshare.net/xSawyer/source-code-management-systems

http://zeroturnaround.com/rebellabs/devprod-report-revisited-version-control-systems-in-2013/#!/