Software Engineering

Version Control

Michael L. Collard, Ph.D.

Department of Computer Science, The University of Akron

SCM

Software Configuration Management

- Tracking and controlling changes to files used in software development
- Based on revision control (version control)
- Used for managing builds and releases
- Used for accounting and auditing of process and product

We use version control for ...

- Coordinating source code for a particular release
- Collecting metrics on software productivity
- Studying the process of development
- Informing non-developers of the current state of the source code
- Bringing new and absent developers up to date

diff and *patch*

- Distribute changes efficiently

- Simplistic form of handling versions

- *diff* utility creates a *patch* file

- The *patch* utility applies the *patch* file to the starting code to create the updated file

Ex: Create a Patch

Ex: Apply a Patch

Version Control (Revision Control)

After an editor and a compiler, the version control is the most crucial tool for software development

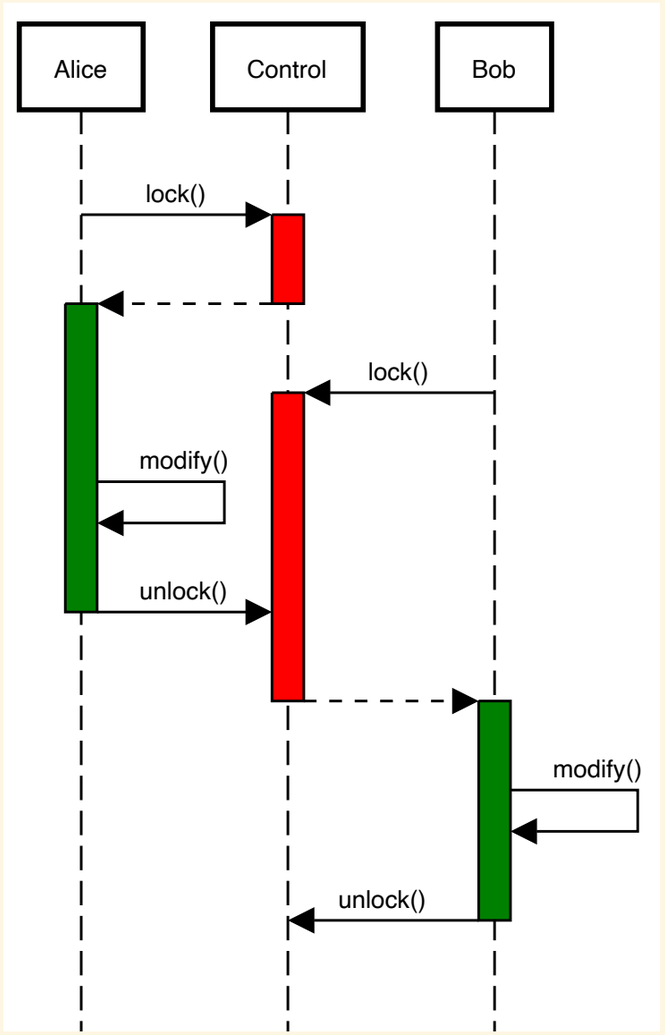
- **Essential** to the coordination of changes among collaborating developers
- **Essential** to a solitary developer working on anything non-trivial
- Maintains a history of changes
- Management of branches and product families
- Defines workflow
- All parts of development revolve around version control

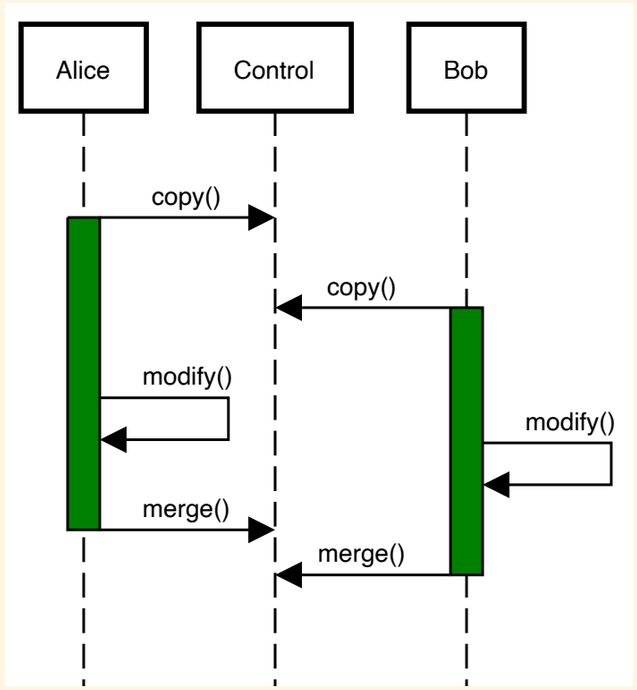
Common Features

```
#include <iostream>
#include <iomanip>
#include <vector>

int main() {

    // input hourly rainfall data
    std::vector<double> rainfall;
    double n;
    while (std::cin >> n) {
        rainfall.push_back(n);
    }
    if (!rainfall.size())
        return 1;

    // calculate average rainfall
    auto total = rainfall[0];
    for (int i = 1; i < rainfall.size(); ++i) {
        total += rainfall[i];
    }

    // calculate heaviest rainfall
    auto max = rainfall[0];
    for (int i = 1; i < rainfall.size(); ++i) {
        if (rainfall[i] > max)
            max = rainfall[i];
    }

    // output the rainfall report
    std::cout << "Average Hourly Rainfall: " << std::fixed << std::setprecision(2) << (total / rainfall.size()) << " hundreds of inches" << '\n';
    std::cout << "Heaviest Hourly Rainfall: " << max << " hundreds of inches" << '\n';

    return 0;
}
```

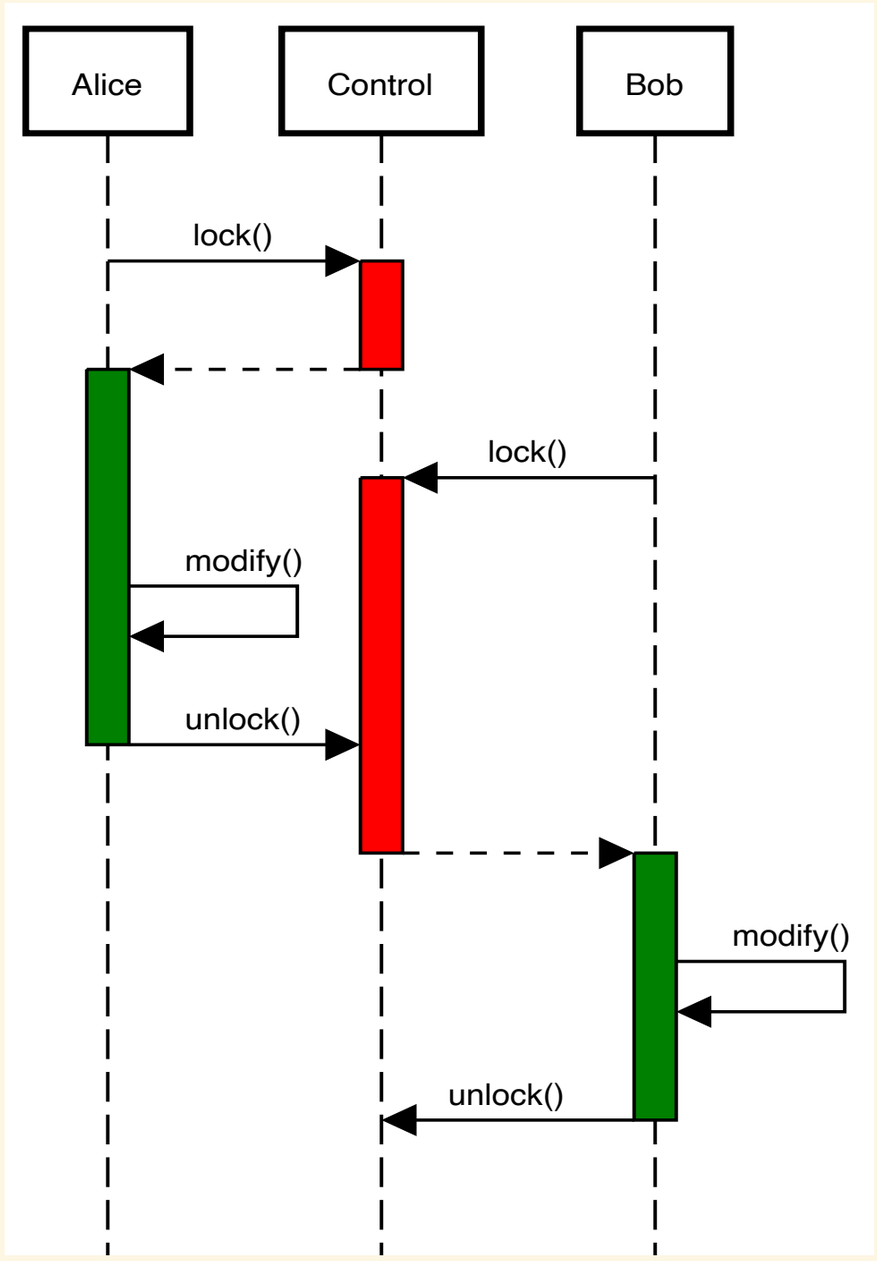
- Versioning down to file level
- **Text Files**: Only understands the *lexical* level (i.e., a source-code file is a file of characters)
- No understanding of the syntactic structure of code
- Does **not** know what a `while` statement is
- **Binary Files**: Stay at the file level

Management Models



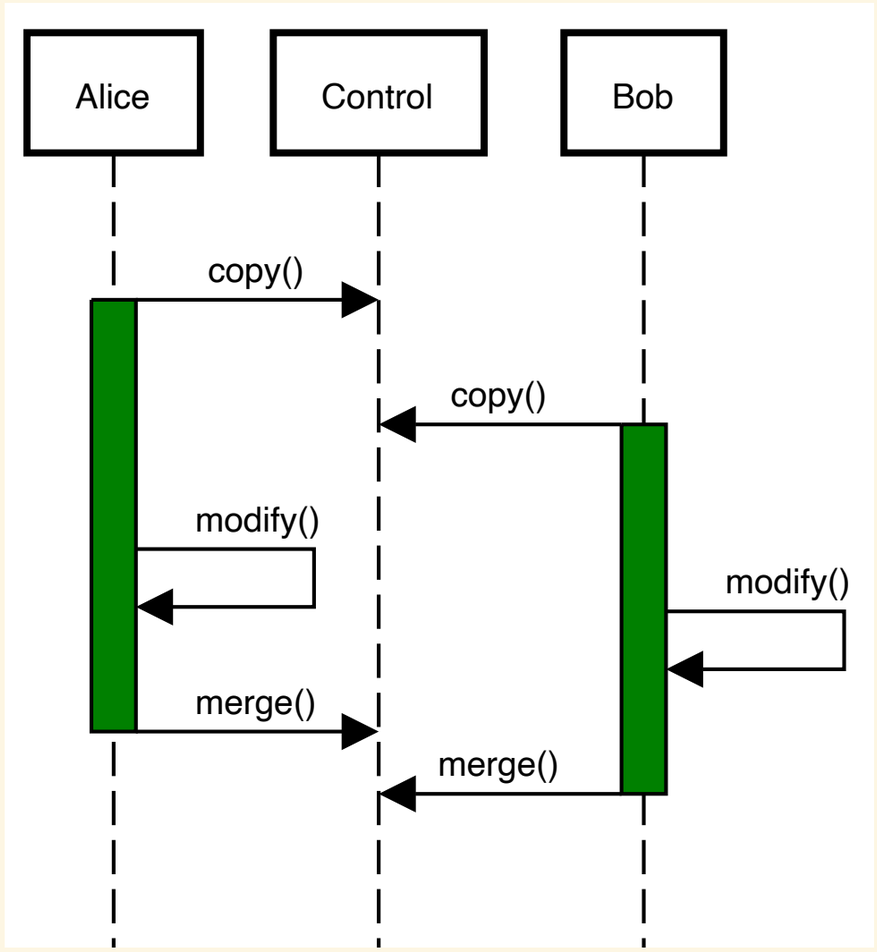


Management Model: File Locking



- Only one developer at a time has access to a file/resource
- *Lock-Modify-Unlock*
- One developer at a time has the "token"; other developers have to wait
- *Library model*
- Advantage: No merging problems
- Disadvantage: Prevents other developers from working
- Disadvantage: Impractical in distributed development due to time/space differences

Management Model: Version Merging



- No restrictions on access
- Developers can work simultaneously
- *Copy-Modify-Merge*
- Advantage: No restrictions on working
- Disadvantage: Merge issues

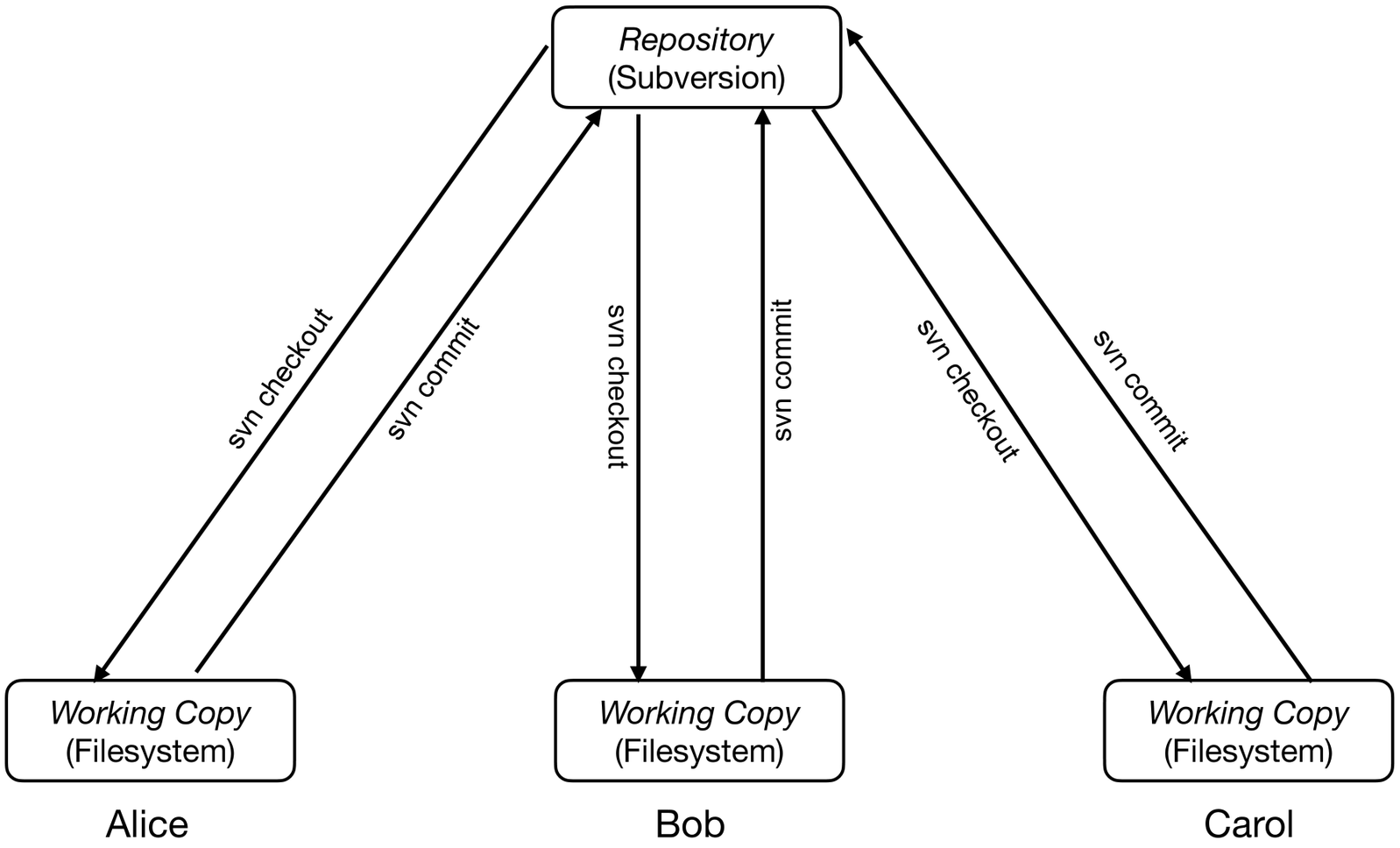
Current Practice

- A large majority of the usage of version control is *Version Merging*

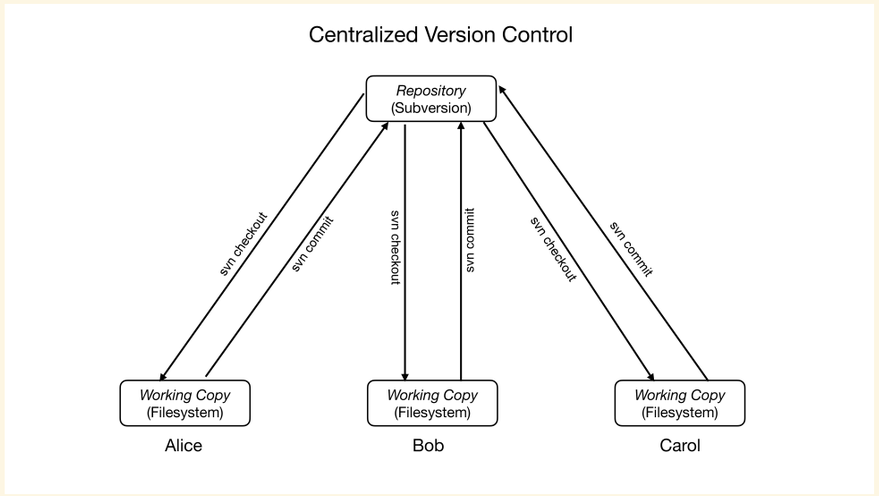
- *File Locking* is typically only used for binary files (e.g., MS Word files)

- May find old projects (and developers) that use File Locking

Centralized Version Control



Centralized Version Control



- e.g., ***Subversion (SVN)***, *ClearCase*, *Vault*

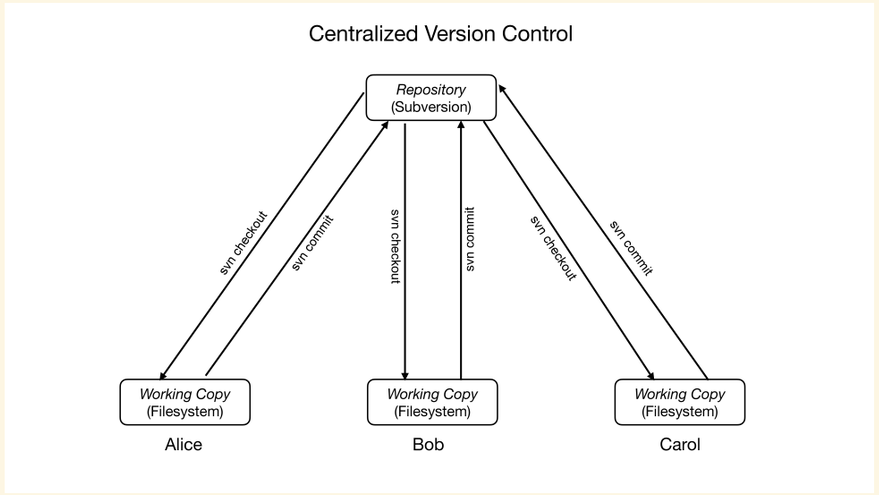
- A single central repository, local working copies

- Access controlled by the server

- One sequence of version numbers

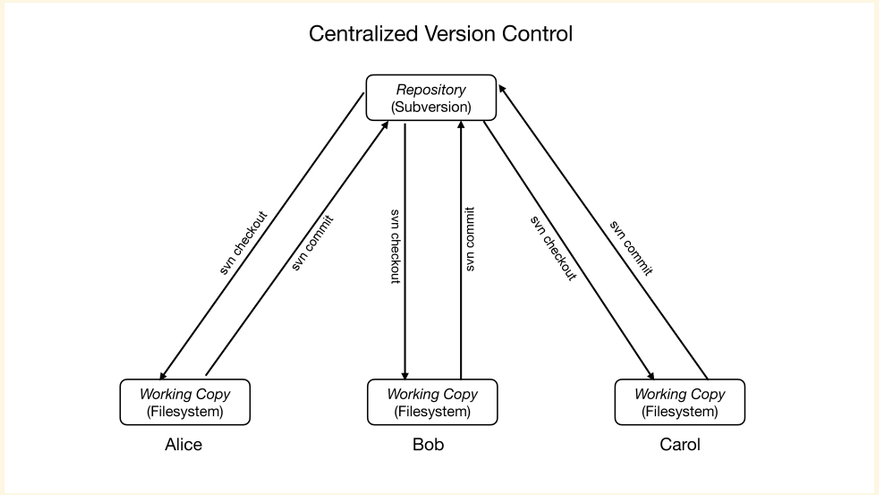
- Traditional approach

SVN View



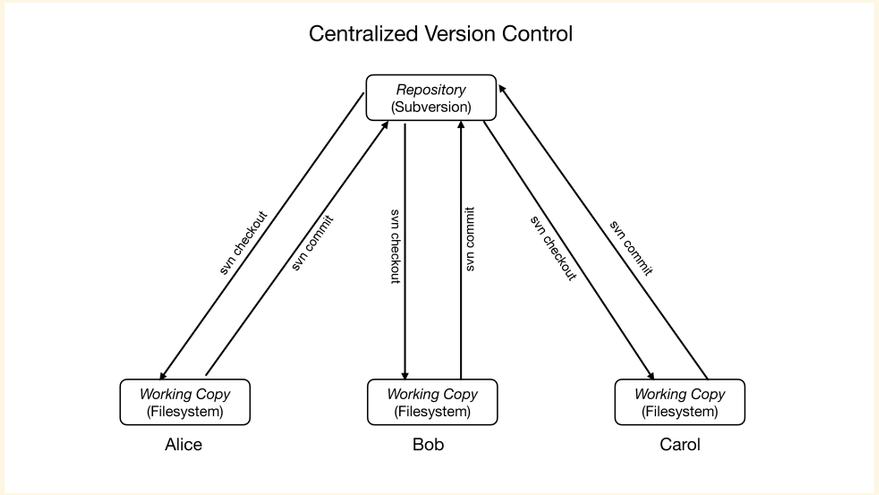
- ***(remote) repository:***
- The single, central SVN repository typically running on a remote machine
- ***working copy:***
  - Sometimes referred to as a *local repository*, but **it is not**
  - The code you checked out into your filesystem
  - Where you modify your files

SVN



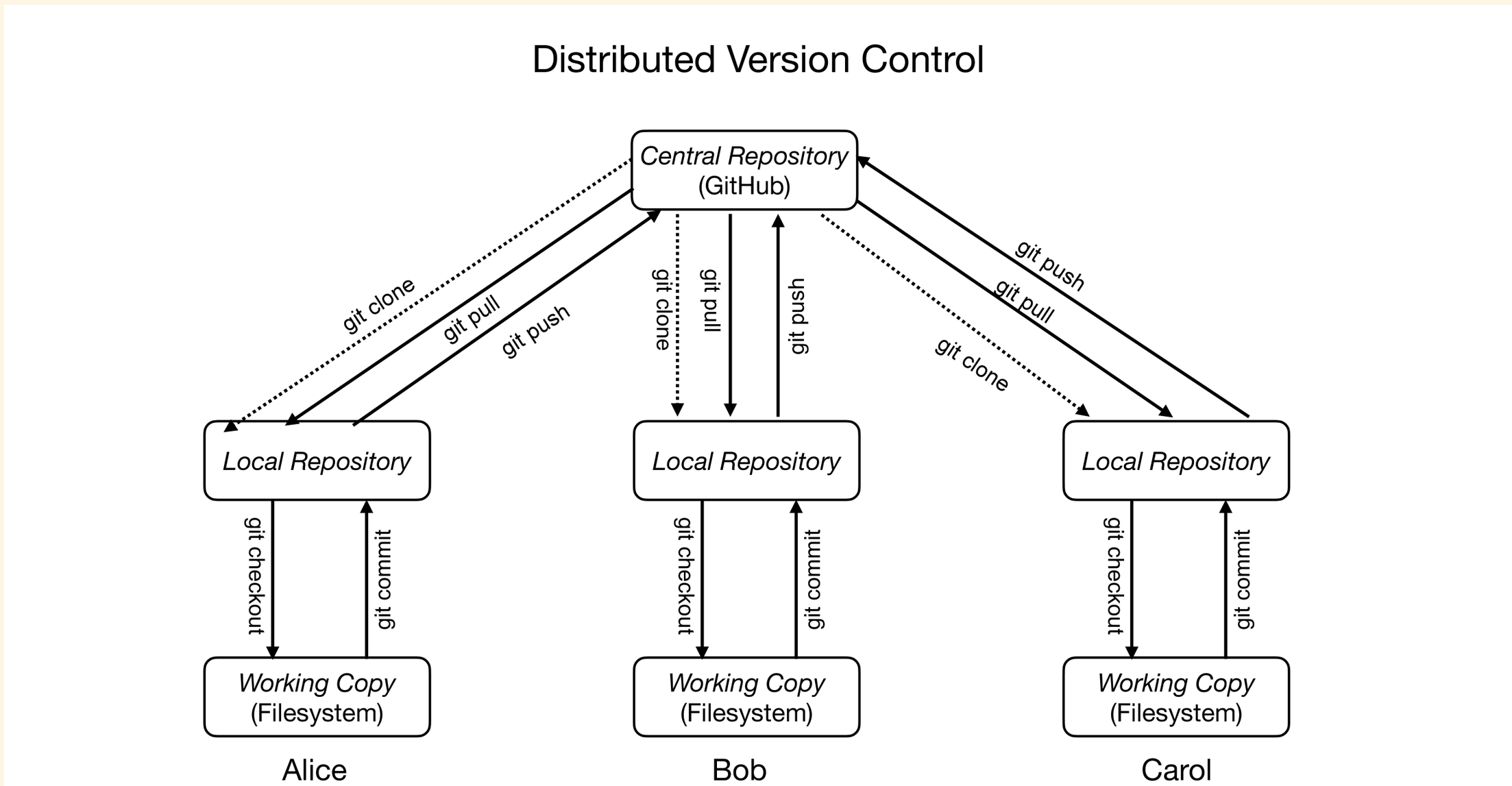
- Versions identified by monotonically increasing numbers
- URLs identify both the location of a central repository and directories/files in the central repository
- Each commit has an *author*
- Support for per-directory permissions, with some limitations

Common SVN Issues

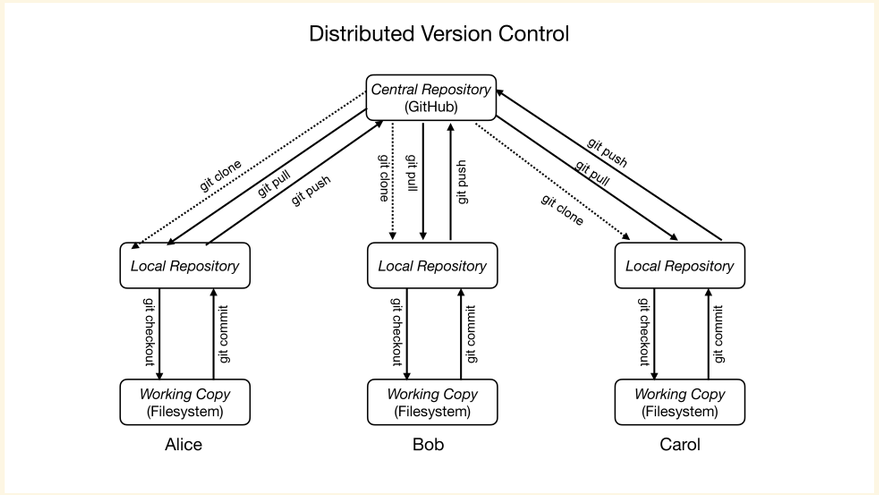


- Need access to a server to create a shared repository
- No distinction between private and public changes
- Merging is difficult
- Branching creates problems

Distributed Version Control



Distributed Version Control



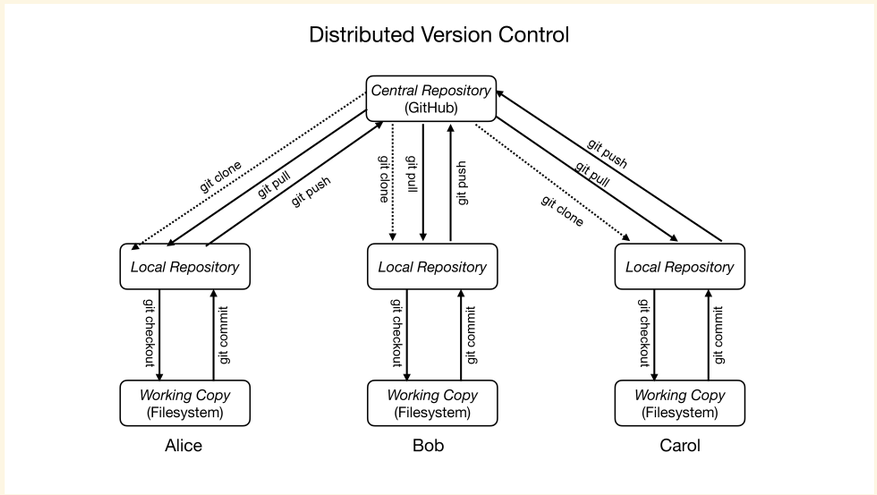
- e.g., ***Git***, *Bazaar*, *Darcs*, *Mercurial*, *Monotone*, *SVK*

- Peer-to-peer, no central repository; all are repository copies

- No one sequence of version "numbers" (Why?)

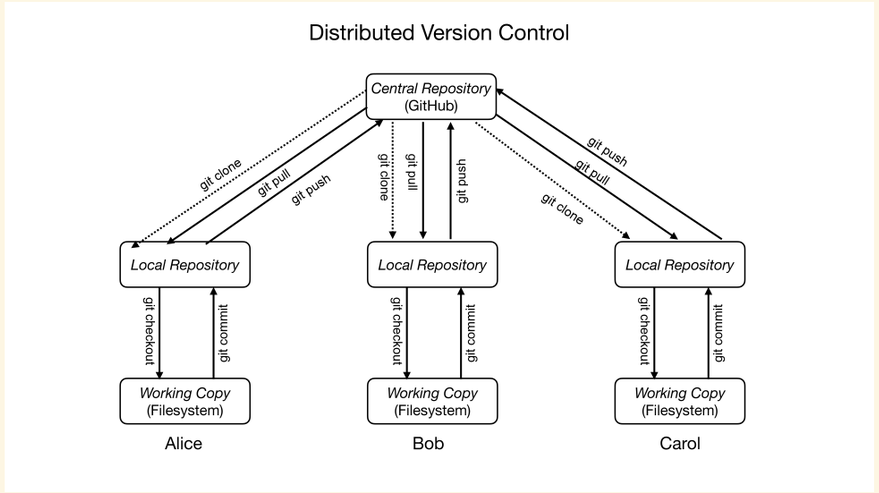
- Access controlled by the server

Git



- Distributed revision control and SCM (Source Code Management) system
- Created in 2005 by Linus Torvalds for Linux kernel development
- Used by major companies, e.g., Microsoft, Apple
- Built-in to many IDEs
- Fluency in Git is a requirement for anybody in software engineering

Git View



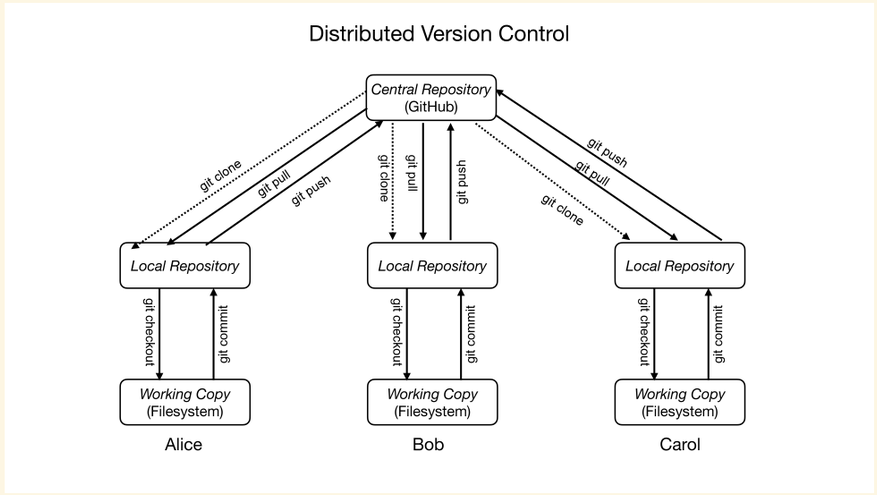
- ***repository***
- Stored in the (hidden) directory *.git*
- What you *clone* from another repository
- ***working copy***
- The code you *checkout* into your filesystem
- The files that you see
- Where you modify the file

Git Characteristics

```
$ git cat-file -p 447f217479def651ac4eb47c4474f89e348372ca
tree 0b8b79464df8d5a86c0a9bf5750f72bfebfe9227
parent 1e69d284b336b3f8edff56ae022154c9a1ce5a41
author Michael L. Collard <collard@uakron.edu> 1727211879 -0400
committer Michael L. Collard <collard@uakron.edu> 1727211879 -0400
gpgsig -----BEGIN SSH SIGNATURE-----
 U1NIUOlHAAAAAQAAADMAAAALc3NoLWVkMjU1MTkAAAAgRH24VmVE7dPEFgUWnY8Y9q9P6v
 t15VPNspgdl4LA2o0AAAADZ2l0AAAAAAAAAAZzaGE1MTIAAABTAAAAC3NzaC1lZDI1NTE5
 AAAAQNTHQL+HCc0gG8kHLj8Afy225jjdpE6i/5LLO6tvisFLSSIo7LgPJsNNYyK6cvgCfI
 dwEkJjDJfMSpLY0nn3oQI=
 -----END SSH SIGNATURE-----

Add JavaScript for test case generation
```

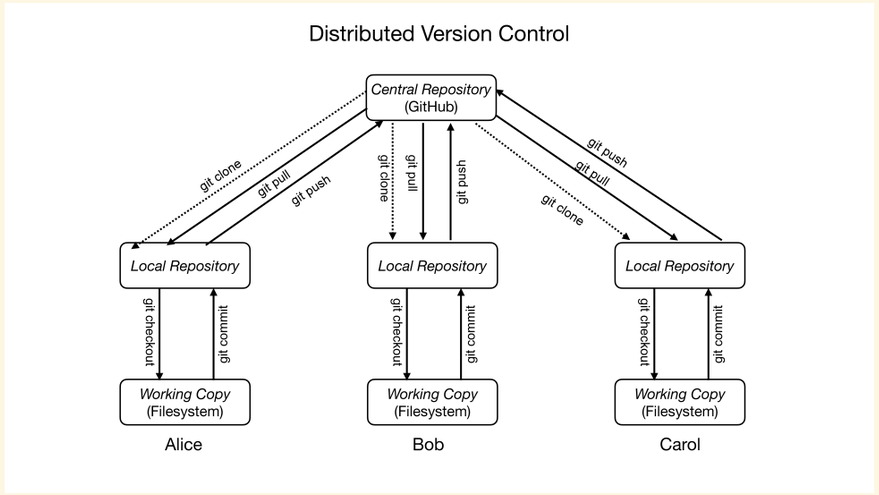
- Each commit has a *hash*, currently a SHA1 id (160-bit numbers in hexadecimal)
- Each commit has an *author* and a *committer*
- peer-to-peer
  - The URL only identifies the repository's location. The repository should always have branches and tags, and the default branch is the "main" (previously "master").
  - Each copy is a full-fledged repository and can be worked on locally without access to a central server
- Each
  chan

Git Benefits



- Records complete new version
- Handles local and remote repositories
- Tracks merged data
- Staging changes

Git Comparison



- Advantages: fast, flexible, powerful, multiuser
- Disadvantages: complex, challenging to learn, GUI tools less developed than SVN tools
- Despite disadvantages, Git is a standard tool for software engineering and software development in general
- Also used as a data format for applications

GitHub

- One issue with Git is that to collaborate with others, your repository must be public
- GitHub is a hosting service for software development projects using Git
- Web-based hosting site for Git repositories
- About 500 million repositories with 100 million public repos, about 120 million users
- Founded in 2008, Microsoft purchased in 2018
- An account at GitHub is necessary for software development and software engineering (not just this class)

Software Ecosystems

- World of Code
- Software Heritage

Git Influence

- **Microsoft:**
- Visual Studio Online 2015
- Visual Studio Blog: Git
- Azure Repos

- **Apple:**
- Current: XCode 15 and macOS Sonoma 14
- XCode - Git/SVN support through XCode 8
- XCode 9: Dropped SVN support
- XCode 11: SVN deprecated
- macOS Catalina 10.15: No longer installed on the command line

Git Tools

- Command-line `git`

- GUI Tools

- IDEs

Recommendation: command-line Git

- Command-line git is the proper Git; everything else is an approximation
- Most answers to Git questions show the command line answer
- As you use the command line, you start to remember commands (not the case in GUIs)
- Often, the GUI "easy" solutions are not the only way to fix a problem and are often not the best
- GUIs overlay proper git and try to make it simpler, but they fail
- Can script/automate command-line solutions

Platform

- Linux - Command-line git

- macOS - Command-line git

- Windows - ~~*Git Bash* in Git for Windows~~ Command-line git in WSL

SVN/Git Command Comparison

SVN	Git
svn checkout *url*	git clone *url* (git checkout *branch*)
svn update	git pull
svn commit -m "Add feature"	git commit -am "Add feature"; git push
svn status	git status
svn revert *path*	git reset –hard *path*
svn add *file*; svn rm *file*; svn mv *file*	git add *file*; git rm *file*; git mv *file*