Objectives

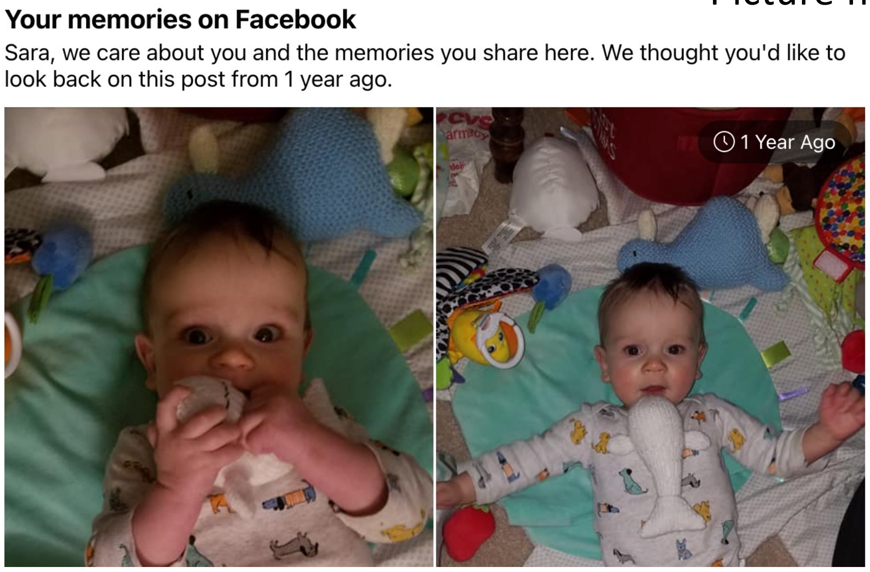
- Version Control Systems

Mar 2, 2022 Sprenkle - CSCI397

1

Facebook Bug

Memory shown 2022, Picture from 2020


Your memories on Facebook

Sara, we care about you and the memories you share here. We thought you'd like to look back on this post from 1 year ago.

Mar 2, 2022 Sprenkle - CSCI111 2

2

Review: Version Control Systems

- What is a Version Control Systems?
  - What are their features?
  - What are their components?
  - What are their benefits?
- What are the differences between a *centralized* and a *distributed* VCS?
- True or False: Git != GitHub

Review: VCS Features

- Collaborate on code with a team
- Roll back/restore older version of code
  - Granularity: Individual files or collection of files
- Store ownership of files/changes and when occurred
- Record reasons for changes
- Track progress
- Each developer has own sandbox of code
- Maintaining multiple branches

Review: Centralized vs Distributed VCS

Centralized

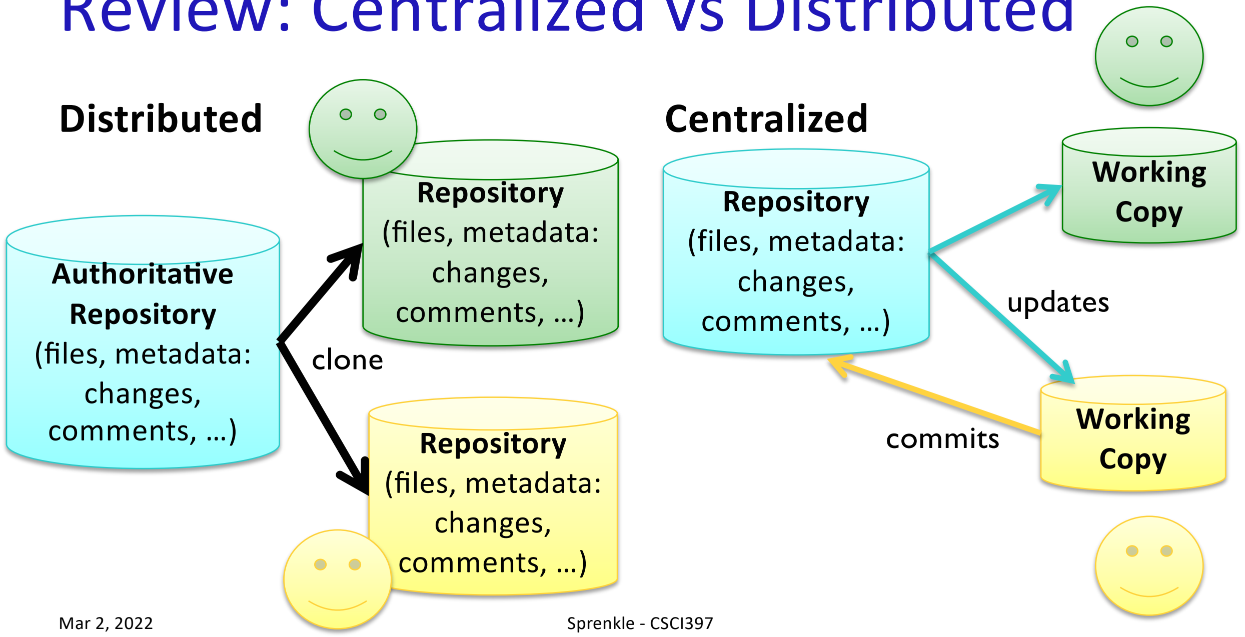
- One central repository: the gold standard
- All updates made against central repo
- No access to repo? No updates
- Must sync with central repo before adding updates
- Examples: CVS, Subversion

Decentralized

- Multiple copies/clones/forks of repositories
- You can always have a local repo
- You can optionally have a central repo
- More distributed sharing options
- Examples: Git, Mercurial, Bazaar

5

Review: Centralized vs Distributed


Distributed
Authoritative Repository (files, metadata: changes, comments, …)
clone
Repository (files, metadata: changes, comments, …)
Repository (files, metadata: changes, comments, …)
Centralized
Repository (files, metadata: changes, comments, …)
Working Copy
updates
Working Copy
commits

6

Review: Centralized vs Distributed Repo Tradeoffs

- CVCS: Mostly remote operations
  - Requires network connectivity for updates, commits
    - More expensive operations
  - Less space for each client
- DVCS: Mostly local operations (faster)
  - Does not require network connectivity
  - Whole copy of the repository
  - More space for each "client"

VCS Design Decision: Who can make changes?

Version Control Systems

- Another TLA for VCS is SCM
  - SCM: Source Code Management
  - Older: Software Configuration Management

Repository Organization

- In CVS and Subversion, typically organize top level something like

```
trunk/        (main)

branches/

tags/
```

9

What Should Be Under Version Control?

- What should not?
- Put another way: What files should you add to your Git repository?
  - But, not a git-specific question

10

What Should Be Under Version Control?

Yes:

- Text-based things made by humans
- Source code
- Scripts
- Files that aren't going to change

Nope:

- Automatically built things
  - executables, object files, jar files
- Temporary files
- Sensitive data: passwords, private ssh keys
- Settings, log files

Most VCSs have ways to ignore these

"Coven: Brewing Better Collaboration through Software Configuration Management"

By Mark Chu-Carroll and Sara Sprenkle, Foundations of Software Engineering, 2000

Abstract: Our work focuses on building tools to support collaborative software development. We are building a new programming environment with integrated software configuration management which provides a variety of features to help programming teams coordinate their work. In this paper, we detail a hierarchy-based software configuration management system called Coven, which acts as a collaborative medium for allowing teams of programmers to cooperate. By providing a family of inter-related mechanisms, our system provides powerful support for cooperation and coordination in a manner which matches the structure of development teams.

https://dl.acm.org/doi/10.1145/355045.355058

What is Coven?

- COllaborative Versioning ENvironment
- goal: wide-area collaboration among many users
- central coordination space for collaborative prog. env't
- primary researcher: Mark Chu-Carroll, IBM

13

The World Then

- Emacs and vi(m) existed but Eclipse didn't
  - But Eclipse's predecessor at IBM did
- CVS existed but Subversion (2000) and Git (2010) didn't
  - Subversion: now an Apache Software Foundation Project
  - Git's predessor BitKeeper did exist

"File based systems like RCS and CVS provide a mechanism called *tagging*, which identifies a version of the project."

14

Version Control System

- distinguishing features
  - mediation model
  - artifact granularity
  - consistency model

SPIDER: Coven January 19, 2000

15

Design Issues

- version-control systems
  - file-based
    - traditional source code organization
    - granularity - too big
  - repository-based
    - granularity - right
    - tied to programming environment

SPIDER: Coven January 19, 2000

16

Design Issues

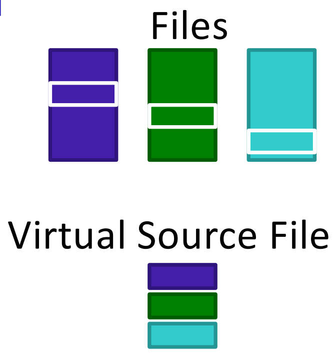
- source management
  - lock-based
  - optimistic (lockless)
- project consistency
  - grouping together program pieces belonging to the same version

SPIDER: Coven January 19, 2000

17

Idea: Fragments + Virtual Source Files

- Problem/Motivation
  - Source code files can be large
  - Developers may only want to edit ***fragments*** of files
  - Developers may want to work on fragments of multiple files (horizontal cut)
- Break source code into fragments (PL-dependent)
- Query source code for fragments to create ***virtual source files***



Mar 2, 2022 Sprenkle - CSCI397

18

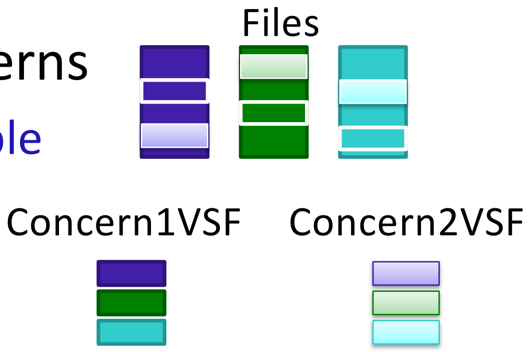
Example Java Fragments

```
package test;

import java.io.*;
import java.util.*;

public class Foo extends Bar implements IBar

protected int _index;

protected String _name;

public static void main( String args[]) {
        …
}
```

19

Idea: Fragments + Virtual Source Files

- Different developers can look at different parts of files
- Principle: Separation of Concerns
  - A file can be made up of multiple concerns



20

Summary

- Coven innovations - better supports collaborative development
  - flexible, adaptable
  - prevents programmer collaboration problems

SPIDER: Coven   January 19, 2000

21

VCS Wrap Up

- Design choices
  - Repository management
  - Artifacts granularity
  - Consistency/collaboration models
- Git is not the last version control system

Mar 2, 2022   Sprenkle - CSCI397

22

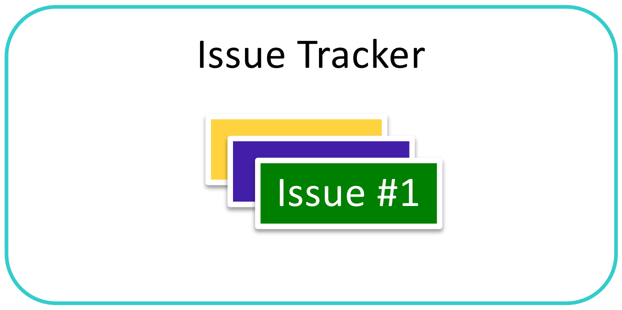
Putting the Pieces Together

- We talked about issue tracking, Scrum boards, and version control, disjointly
- How do these tools fit together?

Mar 2, 2022 Sprenkle - CSCI397

23

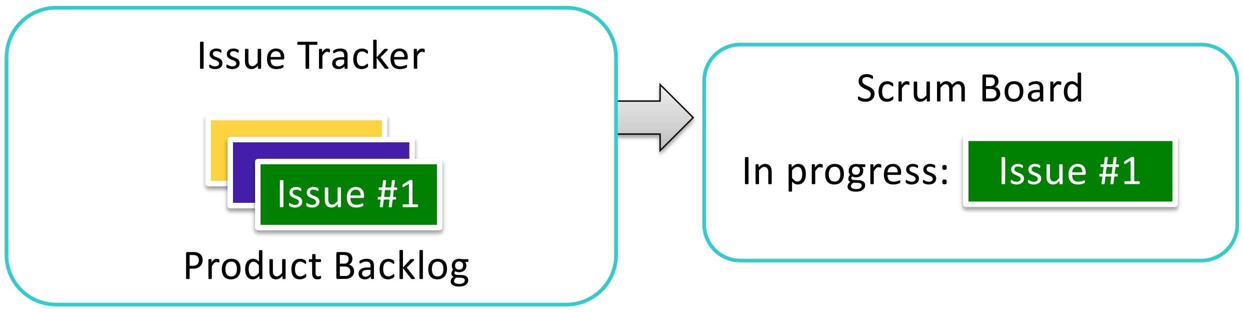
Putting the Pieces Together



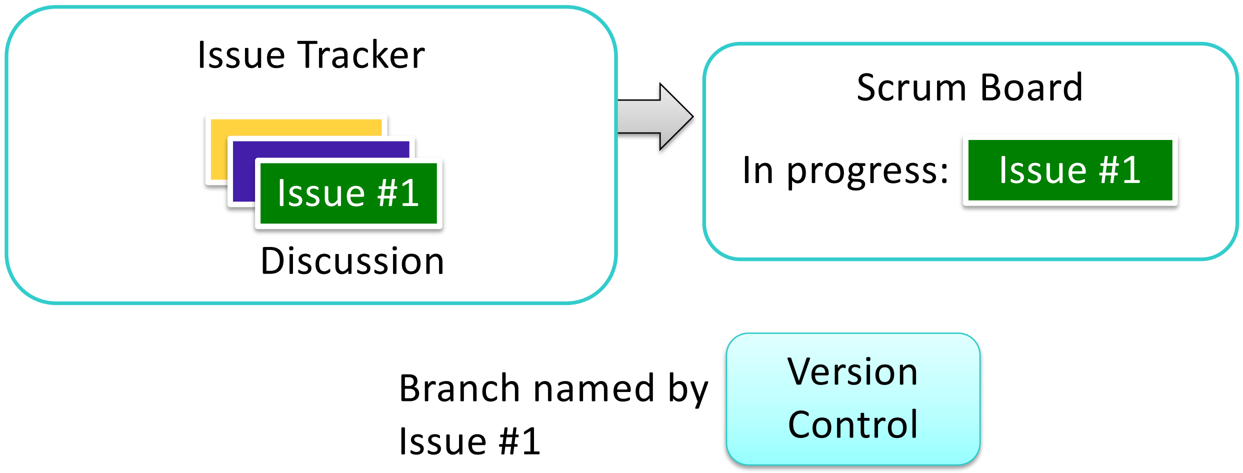
Mar 2, 2022 Sprenkle - CSCI397

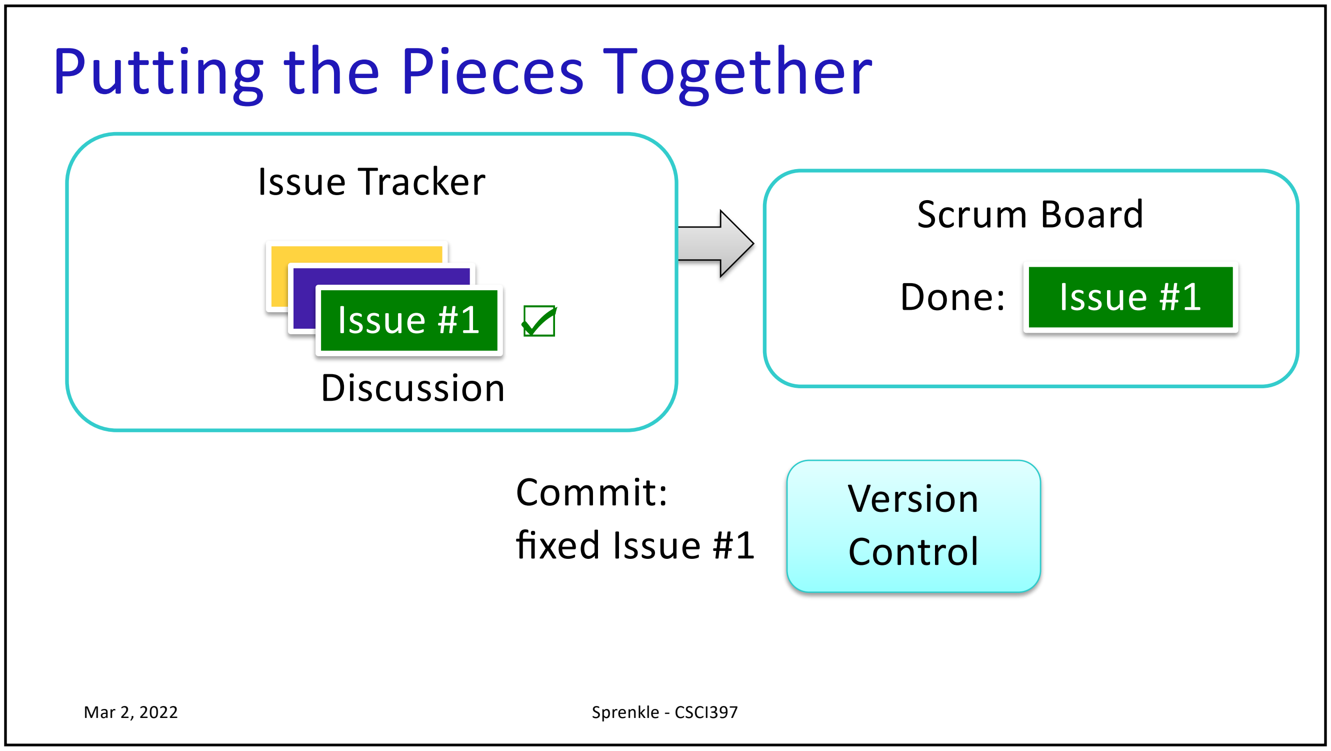
24

Putting the Pieces Together

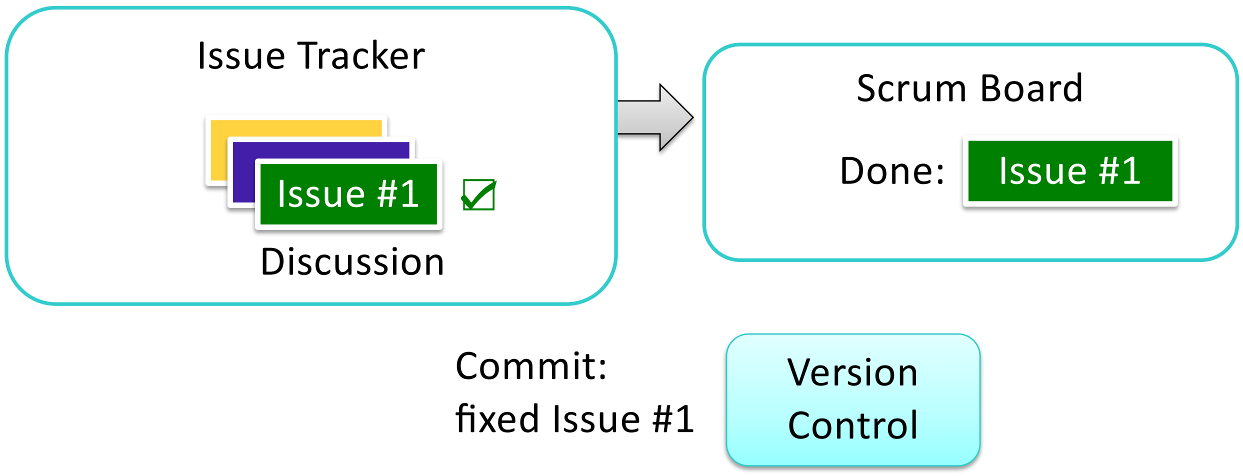


Putting the Pieces Together



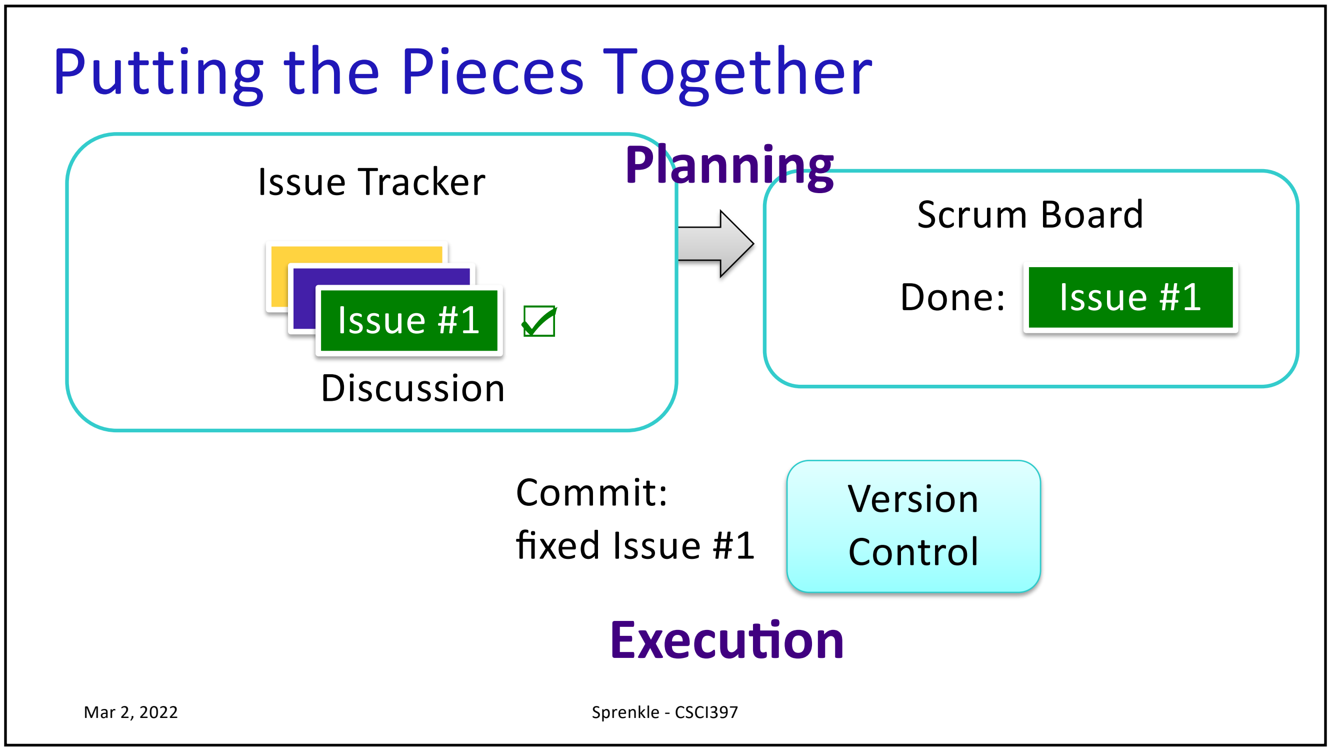


Putting the Pieces Together

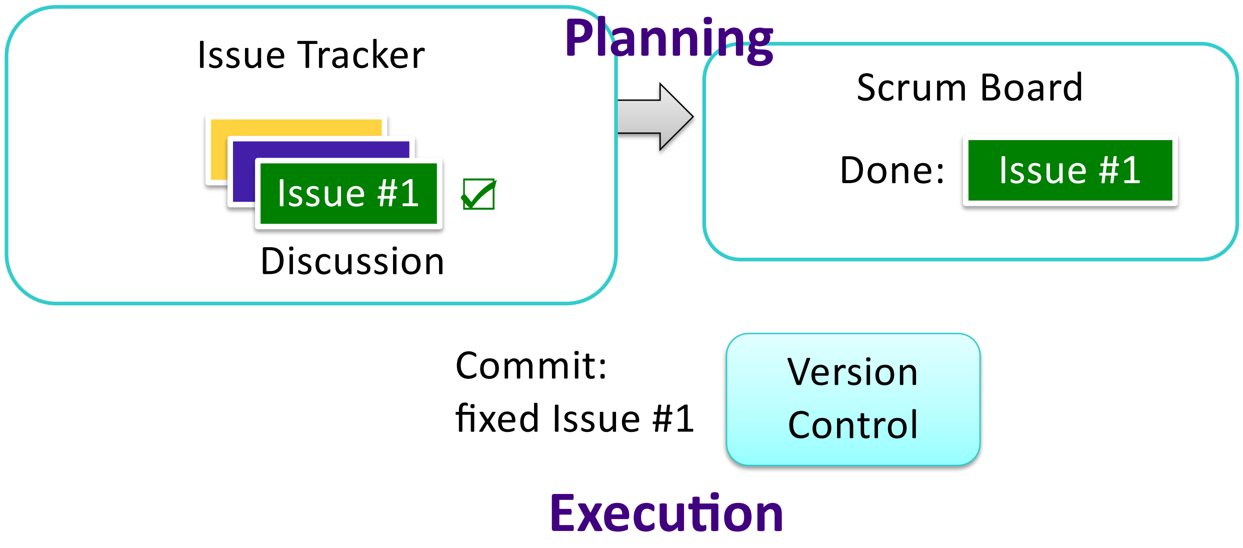


Mar 2, 2022 Sprenkle - CSCI397

27



Putting the Pieces Together



Mar 2, 2022 Sprenkle - CSCI397

28

Pluggable Pieces and Blurred Lines

- I attempt to define each of these components rigidly
  - But the edges aren't always clear
- Various possibilities for combining components and hybrids
  - Examples:
    - Jira's issue tracking with Scrum or Kanban boards and various version tracking systems
    - GitHub's Marketplace has a variety of tools available

Analogy: Cookie Brownies

- Peanut Butter Cookie Brownies
  - Pretty good
  - But definitely better peanut butter cookies, better brownies
- Which is better?
  - PB Cookie Brownies?
  - PB Cookies + Brownies?

How is this related to software tools?

Tradeoffs

All-in-One Solutions

Integrating Solutions

Mar 2, 2022 Sprenkle - CSCI397

31

Tradeoffs

All-in-One Solutions

- Convenient!
  - Updates won't break individual pieces
- Its components might not (individually) be the best
- Do you need all of these

Integrating Solutions

- Separation of Concerns/Single Responsibility Principle
  - Each component is really good
  - Can easily* switch between components
- Integrations can be tricky
- Dependence on multiple things
  - Will upgrades break integrations?

Mar 2, 2022 Sprenkle - CSCI397

32

Other Analogies

- Swiss army knife vs individual tools
  - How does a swiss army knife compare to a phone and its apps?
- Would you rather see a generalist or specialist?

How Should You Choose?

- Consider costs:
  - Purchasing software/service
  - Hosting software/service
  - Maintenance (number of people needed, upgrading, integrations)
  - Ease of use (in general, integrations)
- Do you need *all* those components in the all-in-one solution?
  - Can I add and remove components in the all-in-one?

Looking Ahead

- Watch Software Engineering at Google talk
  - Answer questions, submit on Canvas
  - Before class on Friday

Mar 2, 2022 Sprenkle - CSCI397

35