Processes and Tools

CSC 326

© NC State Software Engineering Faculty

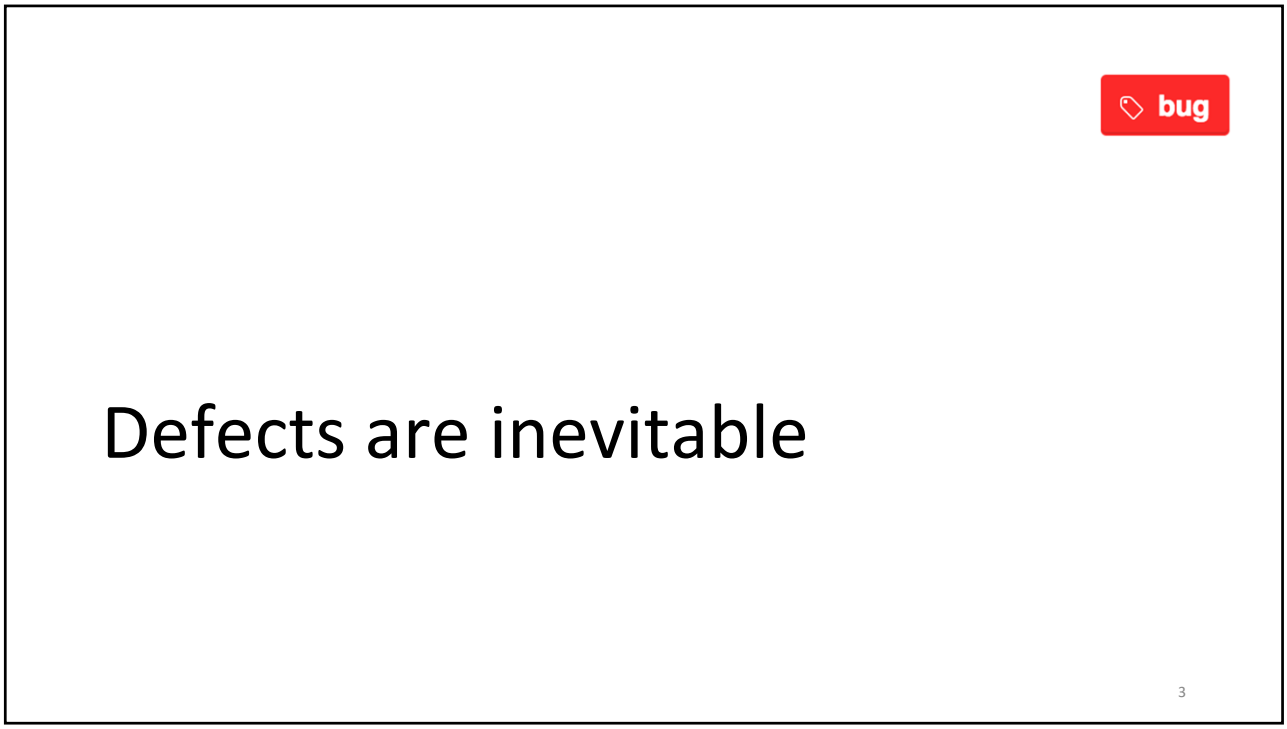
1

Overview

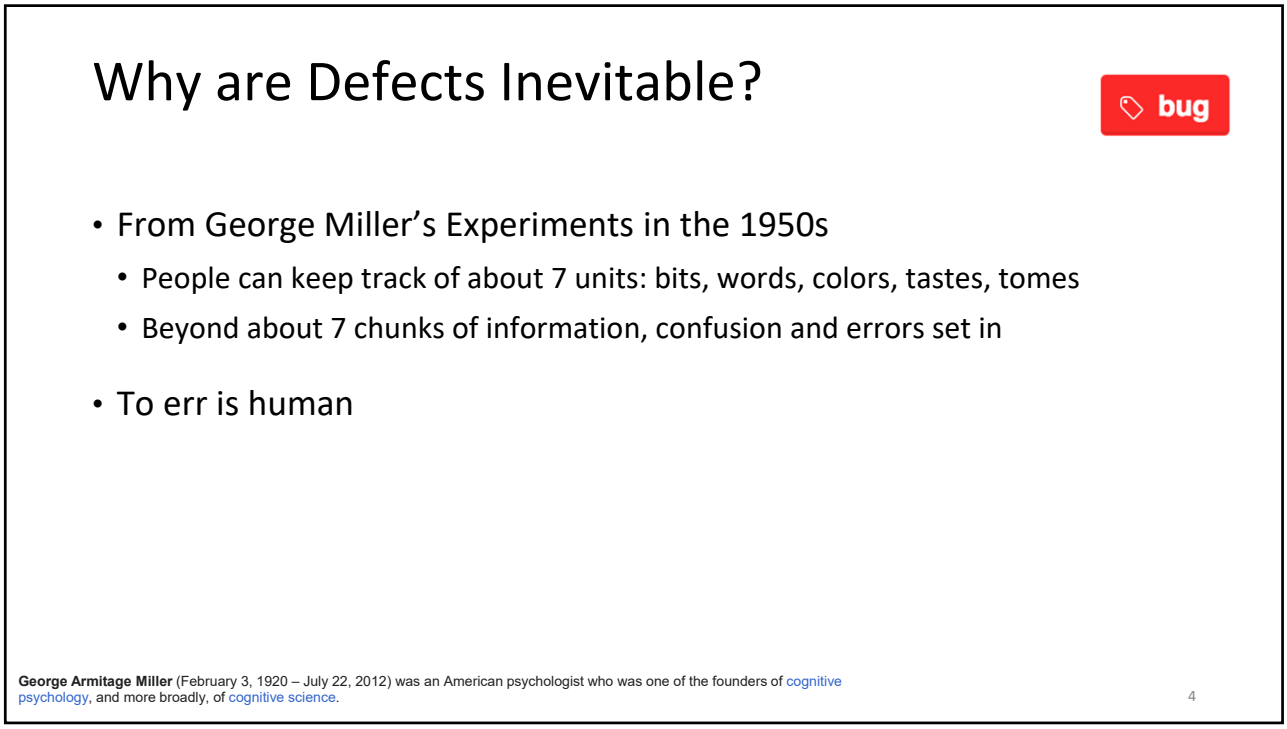
- Bugs, Faults, & Failures
- Git, GitHub, & Issues
- Git Processes in CSC 326
- Onboarding Project (Start in lab on Thursday)

© NC State Software Engineering Faculty

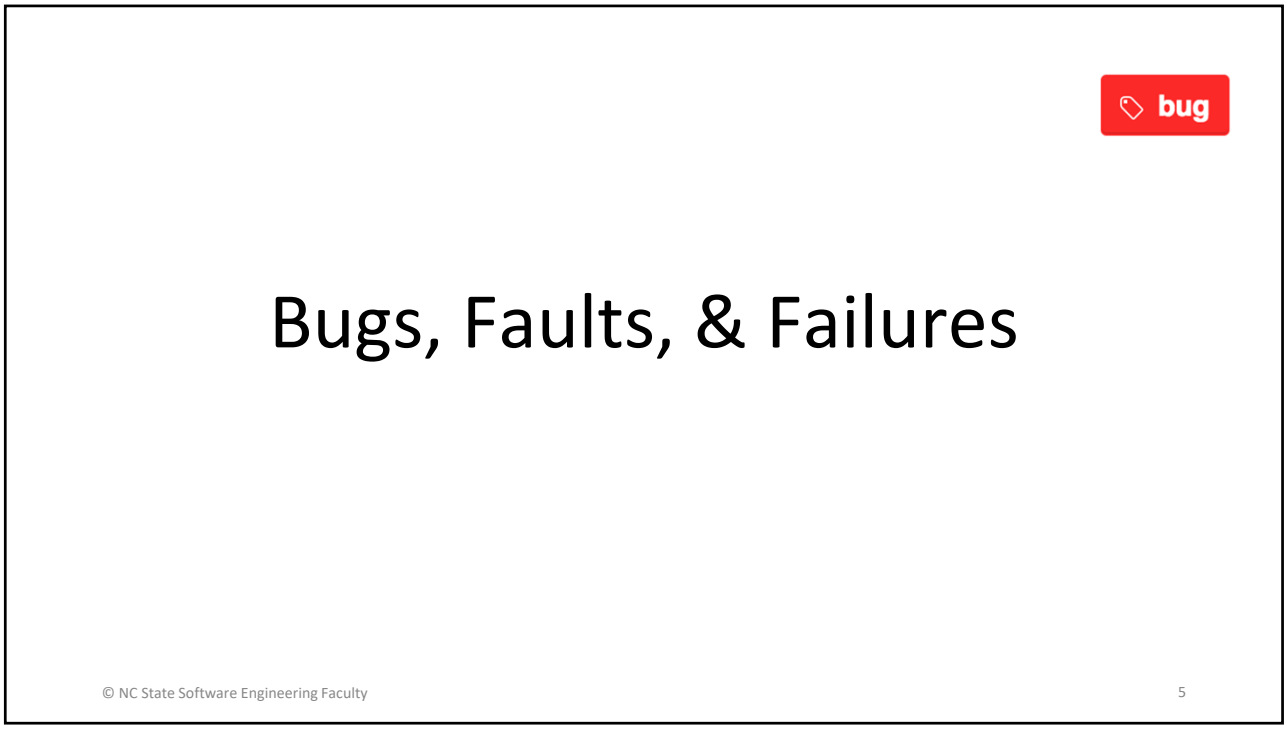
2


bug
Defects are inevitable
3

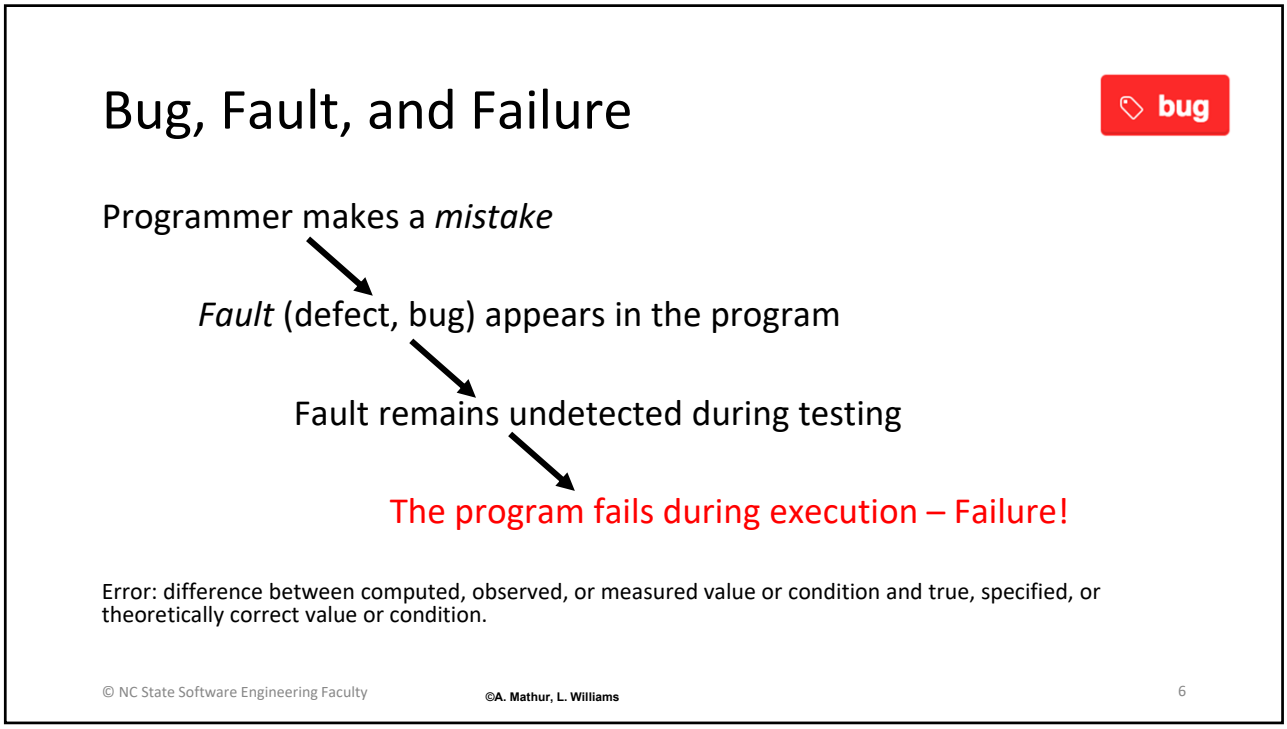
3


Why are Defects Inevitable?
bug
• From George Miller's Experiments in the 1950s
• People can keep track of about 7 units: bits, words, colors, tastes, tomes
• Beyond about 7 chunks of information, confusion and errors set in
• To err is human
George Armitage Miller (February 3, 1920 – July 22, 2012) was an American psychologist who was one of the founders of cognitive psychology, and more broadly, of cognitive science.
4

4


bug
Bugs, Faults, & Failures
© NC State Software Engineering Faculty
5

5


Bug, Fault, and Failure
bug
Programmer makes a *mistake*
Fault (defect, bug) appears in the program
Fault remains undetected during testing
The program fails during execution – Failure!
Error: difference between computed, observed, or measured value or condition and true, specified, or theoretically correct value or condition.
© NC State Software Engineering Faculty
©A. Mathur, L. Williams
6

6

bug



FAILURE
Observable incorrect behavior



FAULT (AKA BUG)
Incorrect code

© NC State Software Engineering Faculty

7

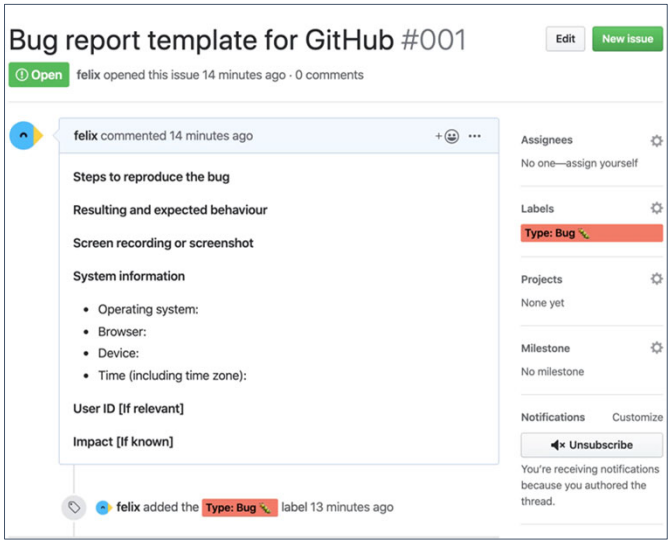
Example

bug

- When an app crashes some of the time
  - There must be a fault or faults in the app's code
  - But, failures occur only some of the time; e.g., when there is a crash

8

Bug Reports



Bug Reports

bug

- Record of system failure – when a system ***fails*** during ***execution***
- Technical Document
  - Describes the failure (output response) of a **System Under Test (SUT)**
  - Describes the test inputs
  - Describes the execution environment
- Communicates failures to developers
- Shows specific quality problems

© NC State Software Engineering Faculty

Key Elements in Bug Reporting



1. **Reproduce**: test it again
1. **Isolate**: test it differently – narrow the reproduction steps to get to the bug the easiest way possible
1. **Generalize**: test it elsewhere, differently

© NC State Software Engineering Faculty
Text © Rex Black

11

Reproduce

REPRODUCE
ISOLATE
GENERALIZE

- **Check it again:** Always check reproducibility of the failure as part of writing bug report
  - Three times tends to work well
- **Write down the steps:** Document a crisp sequence of actions that will reproduce the failure
  - You should be able to give the sequence to someone else and they should be able to follow the instructions to observe the failure

© NC State Software Engineering Faculty
Text © Rex Black

12

Reproduce – Bad Example

REPRODUCE
ISOLATE
GENERALIZE

Nasty bug trashed contents of new file that I created by formatting some text in Arial font, wasting my time.

Why is this a bad bug report?

Think-Pair-Share

© NC State Software Engineering Faculty

Text © Rex Black

13

Reproduce – Better Example

REPRODUCE
ISOLATE
GENERALIZE

Steps to Reproduce

1. I started the *AmazingSpeedyWriter* editor, then I created a new file.
2. I copy-pasted an email into the file.
3. I then typed in four lines of text, repeating "The quick fox jumps over the lazy brown dog" each time
4. I set the font effect to be different on each line: italic, strikethrough, and underline.
5. I highlighted the text, then pulled down the font menu, and selected Arial.
6. The text becomes undecipherable symbols, like "fiIfl›IAˇfl‡fi˝◊∞ ¢ ∞™∞∂∞"
7. I saved the file.
8. I was able to reproduce this problem three out of three tries.

Why is this a better bug report? *Think-Pair-Share*

© NC State Software Engineering Faculty

Text © Rex Black

14

Isolate

REPRODUCE
ISOLATE
GENERALIZE

- Narrow down the reproduction conditions
  - Remove a step, does the bug remain?
  - Simplify a step, does the bug remain?
  - Make changes one by one
- Good isolation shows due diligence and gives developers head start on debugging

© NC State Software Engineering Faculty

Text © Rex Black

15

Isolate – Better Example

REPRODUCE
ISOLATE
GENERALIZE

Steps to Reproduce

1. I started the *AmazingSpeedyWriter* editor, then I created a new file.
2. ~~I copy-pasted an email into the file.~~
3. I then typed in ~~four lines of text, repeating~~ "The quick fox jumps over the lazy brown dog" ~~each time~~
4. ~~I set the font effect to be different on each line: italic, strikethrough, and underline.~~
5. I highlighted the text, then pulled down the font menu, and selected Arial.
6. The text becomes undecipherable symbols, like "fiIfl›IA˘fl‡fi˝◊∞ ¢ ∞™∞∂∞"
7. I saved the file.
8. I was able to reproduce this problem three out of three tries.

© NC State Software Engineering Faculty

Text © Rex Black

16

Generalize

REPRODUCE
ISOLATE
GENERALIZE

- Start changing the steps:
  - Does the same failure occur in other modules or locations?
  - Are there more severe occurrences of the same fault?
- Start moving the steps: how else can you get to the same fault?
- Can be extensive
  - Match amount of effort to the severity of problem
- Generalizing reduces duplicate bug reports and refines understanding of failure

© NC State Software Engineering Faculty

Text © Rex Black

17

Generalize – Better Example

REPRODUCE
ISOLATE
GENERALIZE

Steps to Reproduce

1. I started the *AmazingSpeedyWriter* editor, then I created a new file.
2. I then typed in "The quick fox jumps over the lazy brown dog". I tried typing other text, "Hello, please do not turn to garbage" and the bug remained.
3. I highlighted the text, then pulled down the font menu, and selected Arial.
4. The text becomes undecipherable symbols, like "fiIfl›IA˘fl‡fi˝◊∞ ¢ ∞™∞∂∞"
5. I saved the file.
6. I was able to reproduce this problem three out of three tries.

Generalization

- If you save the file before Arializing the contents, the bug does not occur (moving Step 5 to Step 3)
- On the vague suspicion that this was just a formatting problem, after step 5, I closed AmazingSpeedyWriter and reopened the file. The garbage remained. (adding this step identified another condition for the bug)
- The bug does not occur with existing files. (Tried to see if the bug exists when generalizing on "new file" in step 1)
- Also happens with Helvetica and Droid fonts (generalization on font in step 3).

© NC State Software Engineering Faculty

Text © Rex Black

18

Think-pair-share (3 minutes)

- Why do we make bug reports?
- How is *isolate* different than *generalize*?
- Submit to Exercise 02.01 (link on website)

© NC State Software Engineering Faculty

Extra food - read for fun!

What Makes a Good Bug Report?

Nicolas Bettenburg*
nicbet@st.cs.uni-sb.de

Sascha Just*
just@st.cs.uni-sb.de

Adrian Schröter¶
schadr@uvic.ca

Cathrin Weiss‡
weiss@ifi.uzh.ch

Rahul Premraj*§
premraj@cs.uni-sb.de

Thomas Zimmermann+§
tz@acm.org

* Saarland University, Germany
‡ University of Zurich, Switzerland
¶ University of Victoria, BC, Canada
+ University of Calgary, Alberta, Canada

https://thomas-zimmermann.com/publications/files/bettenburg-fse-2008.pdf

Git, GitHub, & Issues

(Tracking and fixing bugs and changes)

© NC State Software Engineering Faculty

21

GitHub

Web-based Git Service

Hosting & Service Platform



© NC State Software Engineering Faculty

22

Video: *GitHub Process*

- Words to watch for:
  - "Issue"
  - "Branch"
  - "Pull Request"

© NC State Software Engineering Faculty

23



24

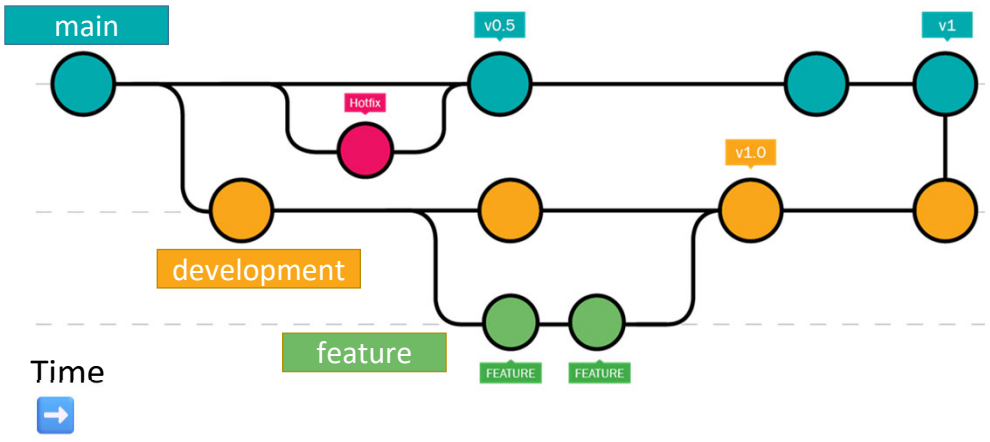
GitHub – for Version Control

- Provides a central, remote repository to support collaborative development of (sometimes very large) software systems.
- Repositories are **cloned** locally for work
  - Can also initialize repositories and later connect to a remote repository
- Changes are committed locally and **pushed** to the remote

- **Submission of work in this class means that you're pushing your local changes to the remote NC State controlled repository.**

© NC State Software Engineering Faculty

GitHub

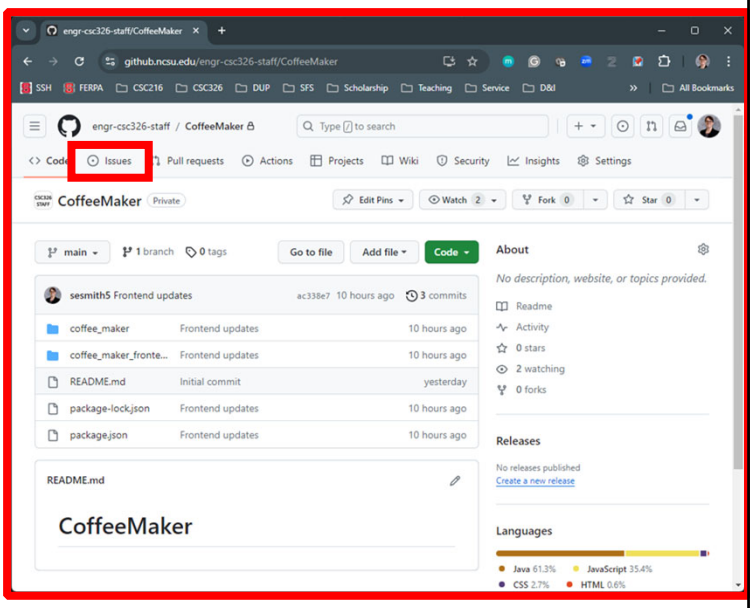


© NC State Software Engineering Faculty

Image: https://medium.com/@sam.rizae/git-workflow-strategy-37bd5c242133

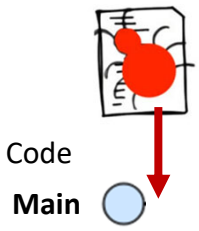
Issues

- Used to track *tasks* and *bugs* for a project
  - Label the issue – bug, feature, etc.
  - Associate the issue with a milestone by naming major portions of your project
  - Assign the issue to a team member
- Provides a mechanism for conversations around a project
- Interact with issues through the command line
- Tie issues to feature branches



© NC State Software Engineering Faculty

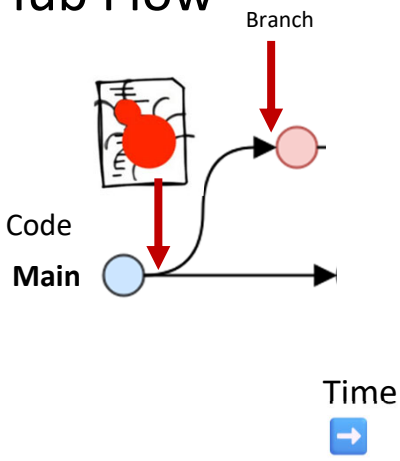
The GitHub Flow



Time

© NC State Software Engineering Faculty
https://guides.github.com/pdfs/githubflow-online.pdf

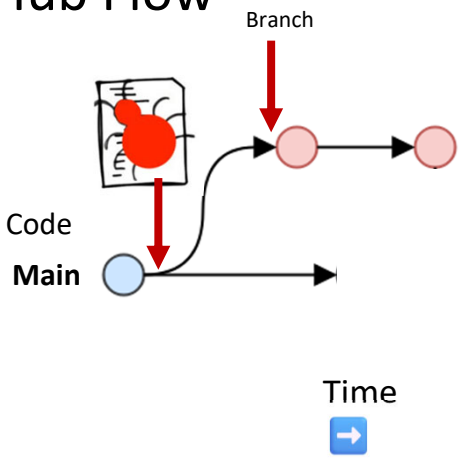
The GitHub Flow



© NC State Software Engineering Faculty
https://guides.github.com/pdfs/githubflow-online.pdf

29

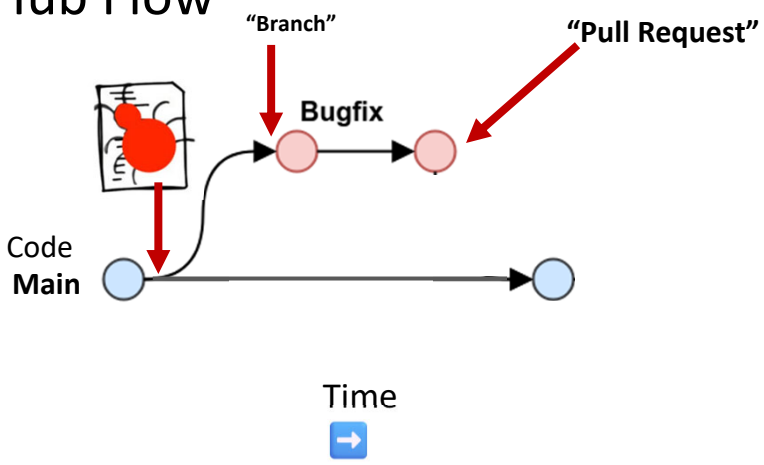
The GitHub Flow



© NC State Software Engineering Faculty
https://guides.github.com/pdfs/githubflow-online.pdf

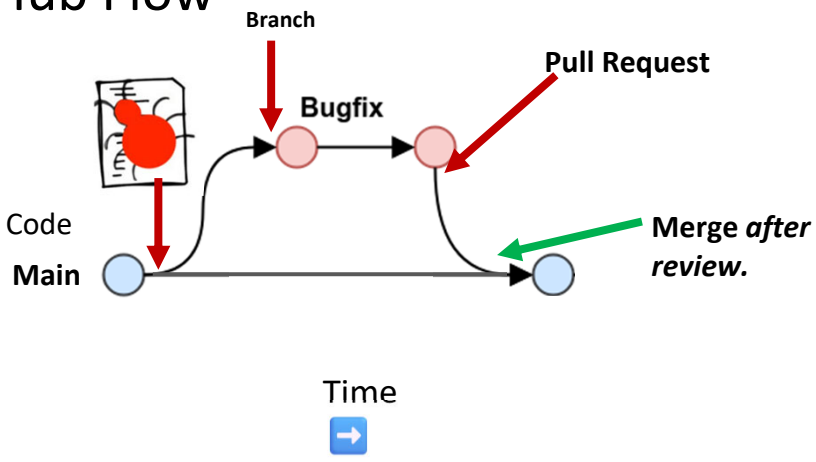
30

The GitHub Flow



31

The GitHub Flow



32

Pull Requests: What are they?

- **Pull requests** let you tell others about changes you've pushed to a GitHub repository. Once a pull request is sent, interested parties can review the set of changes, discuss potential modifications, and even push follow-up commits if necessary.
- Pull Requests are **living conversations** that streamline the process of discussing, reviewing, and managing changes to code.

https://yangsu.github.io/pull-request-tutorial/
https://github.com/features/projects/issues

© NC State Software Engineering Faculty

Why **Pull Requests?**

- Continuous integration!
  - New features should be introduced on **feature** branches of **development**
  - **main** should always be clean and build to a working product
  - Review is ***before*** merge, not after
  - … basically, you don't want to automatically override **main**

© NC State Software Engineering Faculty

Why Code Review?

Code review is a quality control process (Linus's law is: *given enough eyeballs, all bugs are shallow* 👀).

Code review promotes knowledge sharing across teammates, and reduces information hoarding.

Code review also promotes culture sharing across an organization by communicating norms.

Code review is especially important given generative AI use in software engineering.

Who is Linus? Linus Torvalds, the creator of the Linux kernel. The law was formulated by Eric S. Raymond in his essay and book The Cathedral and the Bazaar (1999).

What should I write in a review?

- If it's code, make sure you understand:
  - *why* the code was changed
    - **Commit messages** are helpful here
  - *how* the code changes the program behavior
    - **Code comments** are helpful here
    - **Test cases** are also helpful here
- If there's something you don't understand, ask!
- If they could do something better, suggest changes!
- If something is wrong, doesn't meet the requirements, hacky, hard-coded, redundant, inconsistent, say so (constructively!)

© NC State Software Engineering Faculty

Think-pair-share (3 minutes)

- Why do we fix bugs on a *branch* of the code?
- Why is *Pull Request review* useful?
- Complete Exercise 02.02

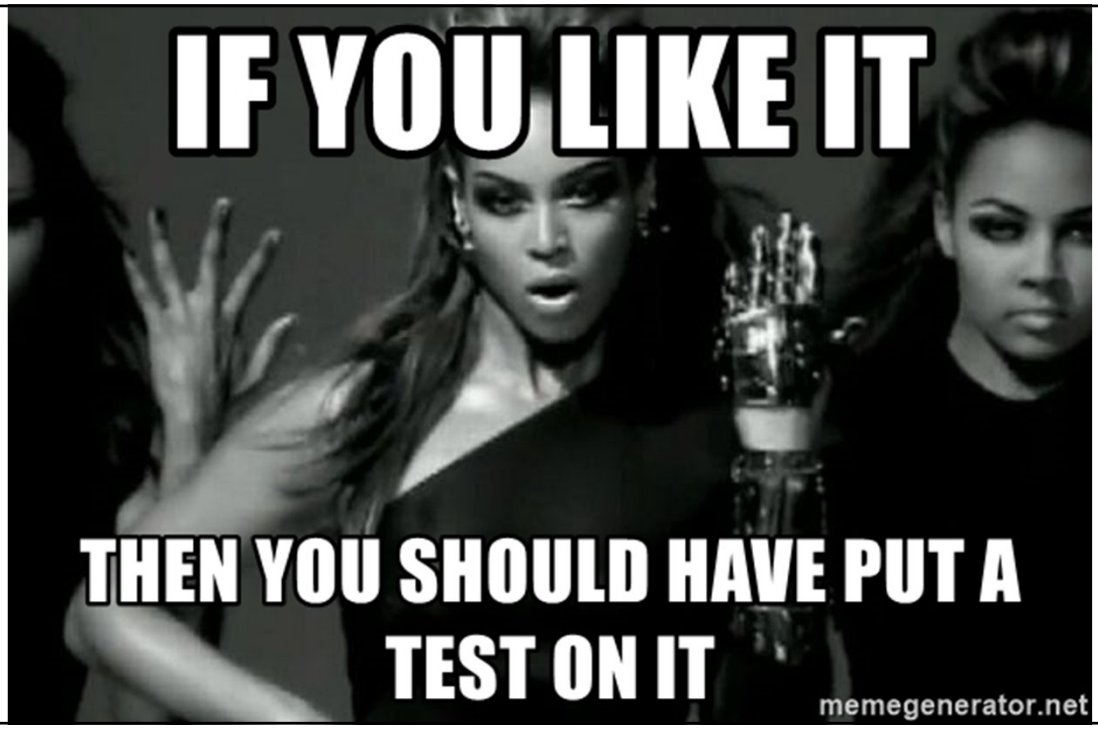
© NC State Software Engineering Faculty

Pro Tips

- **Pull in changes** from 'upstream' often so that you stay up to date so that when you submit your pull request, merge conflicts will be less likely.
- **It's best to test**. Run your changes against any existing tests if they exist and create new ones when needed. Whether tests exist or not, make sure your changes don't break the existing project.
- **Wait for GitHub Actions to run before completing the merge**. We're using GitHub Action to build (and eventually test) our systems. The build should finish before merging. But if you're doing a good review of pull requests this shouldn't be an issue.

© NC State Software Engineering Faculty

https://guides.github.com/activities/contributing-to-open-source/



39

39

Resources

- GitHub Pull Requests Documentation
- Pull Request Tutorial
- Understanding the GitHub Flow
- Command line video of pull request and conflict resolution

© NC State Software Engineering Faculty

40

40

Git Processes in 326

© NC State Software Engineering Faculty

Git Processes in 326

GitHub Issues

- Document all enhancements and bugs as issues
  - Bugs should provide a full bug report
- Assign issues to team members
- Commits must reference the issue addressed (e.g., “fixes #52”)

Branching & Pull Requests

- development branch to integrate features before pushing “live.”
- **No direct pushes to main (and development)**
  - All commits and pushes must be on feature branches
  - All changes to development and main come from pull requests
- You may not approve your own feature (unless an individual repo). A team member must review and the results of the review must be in the feature documentation.
- All features must be merged into main to be graded.
- Branch naming: “unityid/branchname”

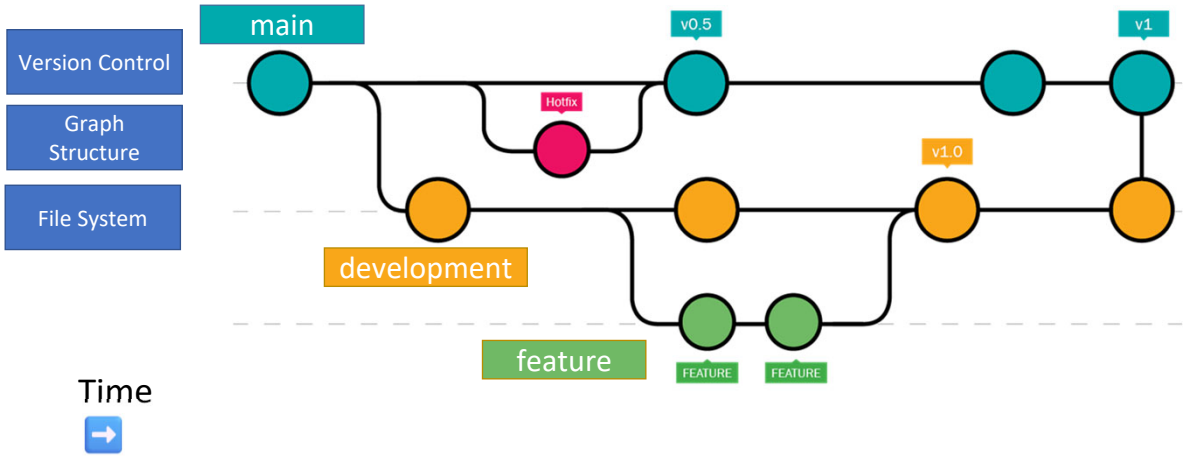
© NC State Software Engineering Faculty

GitHub

Version Control

Graph Structure

File System



© NC State Software Engineering Faculty

Image: https://medium.com/@sam.rizae/git-workflow-strategy-37bd5c242133

Maven

Yiddish: **"an expert on something"**

Problem: *ARG!!! I can't find where to download version 3.1.4 of* ***Super_Useful_Library.jar***

Solution: ***Use a package manager. Like Maven or npm.***

© NC State Software Engineering Faculty

Maven

- Project management tool
- Executable – `mvn`
  - Will execute targets to build, test, deploy a project given the project's POM file
- Project – POM Files
  - Information about how to build, test, deploy a project

© NC State Software Engineering Faculty

Project Information

```
<?xml version="1.0" encoding="UTF-8"?>
   <project xmlns="http://maven.apache.org/POM/4.0.0"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-
   4.0.0.xsd">
       <modelVersion>4.0.0</modelVersion>
       <parent>
          <groupId>org.springframework.boot</groupId>
          <artifactId>spring-boot-starter-parent</artifactId>
          <version>3.5.16</version>
          <relativePath/> <!-- lookup parent from repository -->
       </parent>
       <groupId>edu.ncsu.csc326</groupId>
       <artifactId>coffee_maker</artifactId>
       <version>0.0.1-SNAPSHOT</version>
       <name>coffee_maker</name>
       <description>CoffeeMaker project for CSC 326</description>


</project>
```

Maven Info

Parent Info (Spring Boot)

Project Info

© NC State Software Engineering Faculty

Dependencies

- Libraries needed for your project. Maven will find them and download them into your project
  - No more lib/ directory!
- Search for dependencies in locations like mvnrepository.com, even if pulling from another repository

```
<dependencies>
    <dependency>
       <groupId>org.springframework.boot</groupId>
       <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
       <groupId>org.springframework.boot</groupId>
       <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

</dependencies>
```

© NC State Software Engineering Faculty

Properties

- Describe information about project characteristics

```
<properties>
    <java.version>21</java.version>
    <maven-checkstyle-plugin.version>3.6.0</maven-checkstyle-plugin.version>
    <checkstyle.version>13.10.0</checkstyle.version>
</properties>
```

© NC State Software Engineering Faculty

Build Targets

- Describes how to build, test, and deploy the project
  - Defaults for common behavior
  - This will be used by GitHub Actions!

```
<build>
   <plugins>
…
      <plugin>
         <groupId>org.apache.maven.plugins</groupId>
         <artifactId>maven-checkstyle-plugin</artifactId>
         <version>${maven-checkstyle-plugin.version}</version>
         <dependencies>
            <dependency>
               <groupId>com.puppycrawl.tools</groupId>
               <artifactId>checkstyle</artifactId>
               <version>${checkstyle.version}</version>
            </dependency>
         </dependencies>
         <configuration>
            <configLocation>config/checkstyle/csc_checkstyle.xml</configLocation>
            <includeTestSourceDirectory>true</includeTestSourceDirectory>
            <consoleOutput>true</consoleOutput>
            <!-- The CSC config reports everything at severity "warning". -->
            <violationSeverity>warning</violationSeverity>
            <!-- Report style problems without breaking the build. -->
            <failOnViolation>false</failOnViolation>
            <failsOnError>false</failsOnError>
         </configuration>
         <executions>
            <execution>
               <id>checkstyle-validate</id>
               <phase>validate</phase>
               <goals>
                  <goal>check</goal>
               </goals>
            </execution>
```


Repositories (Optional)

- Places to look for dependencies

```
<repositories>
   <repository>
      <id>spring-releases</id>
      <url>https://repo.spring.io/libs-release</url>
   </repository>
</repositories>
<pluginRepositories>
   <pluginRepository>
      <id>spring-releases</id>
      <url>https://repo.spring.io/libs-release</url>
   </pluginRepository>
</pluginRepositories>
```

© NC State Software Engineering Faculty

Onboarding Project

© NC State Software Engineering Faculty

Onboarding Project

- CoffeeMaker project requirements (very high level)
  - Add, edit, delete recipes
  - Add inventory
  - Make coffee
  - Some new ones will be added next week
- Introduces you to course technologies!
- General Tasks for Milestone 0
  - Setup development environment + AI tooling (record demo)
  - Get familiar with the CoffeeMaker requirements
  - Follow good process

© NC State Software Engineering Faculty

Evaluation

- Technical Deliverables – includes code, test, documentation, reflections, etc.
  - Team Technical Deliverables
  - Individual Technical Deliverables
- Technical Process
- Team Collaboration
- Project Management
- Retrospectives

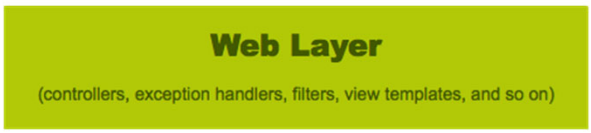
- Possible deductions for poor participation & collaboration beyond the points available in the Team Collaboration section
- Deduction for not having a working & demoed environment by the deadline. If you have any trouble, please be sure to attend TA office hours.

© NC State Software Engineering Faculty

CoffeeMaker Architecture





























© NC State Software Engineering Faculty

Milestone 0 – OBP Setup + Debugging

- Before lab – set up your development + AI environment!
- Set up CoffeeMaker (You'll complete Milestone 0 in your individual OBP repository) – try to do this before lab too!
  - Demo video by August 26th at 11:45pm
- Workshop – bug fixes + process + AI
- Individual Work – find and document bugs
- Team Collaboration – no points this week
- Practice Process
- Practice Project Management

Individual Deliverables: Due Wednesday

You are expected to work with others in your lab on the workshops during lab!

© NC State Software Engineering Faculty