Release Management A Problem You Cannot Afford To Ignore (5 Steps to an Automated Release Flow)

Jeffrey Wren

Paradigm Works

PARADIGM™ WORKS

Introduction

- Release Management Defined
  - Release management is the set of steps taken to guarantee that one's source code, schematic, layout, etc, is ready for distribution to the customer.
  - The customer is defined as another team member, another division within the company, or a customer in the traditional sense.

Overview

- The 3 problems of release management
- The 5 steps to a good release flow
  1. Create/Update Workspace
  2. Modify and Commit Changes
  3. Submit Changes
  4. Integrate Submits (Automate)
  5. Publish (Automate)
- Case Study
- Conclusions

Jeffrey Wren

Release Management Problems

- The User Problem
- The Release Manager Problem
- The Reproducibility Problem

Jeffrey Wren

User Defined

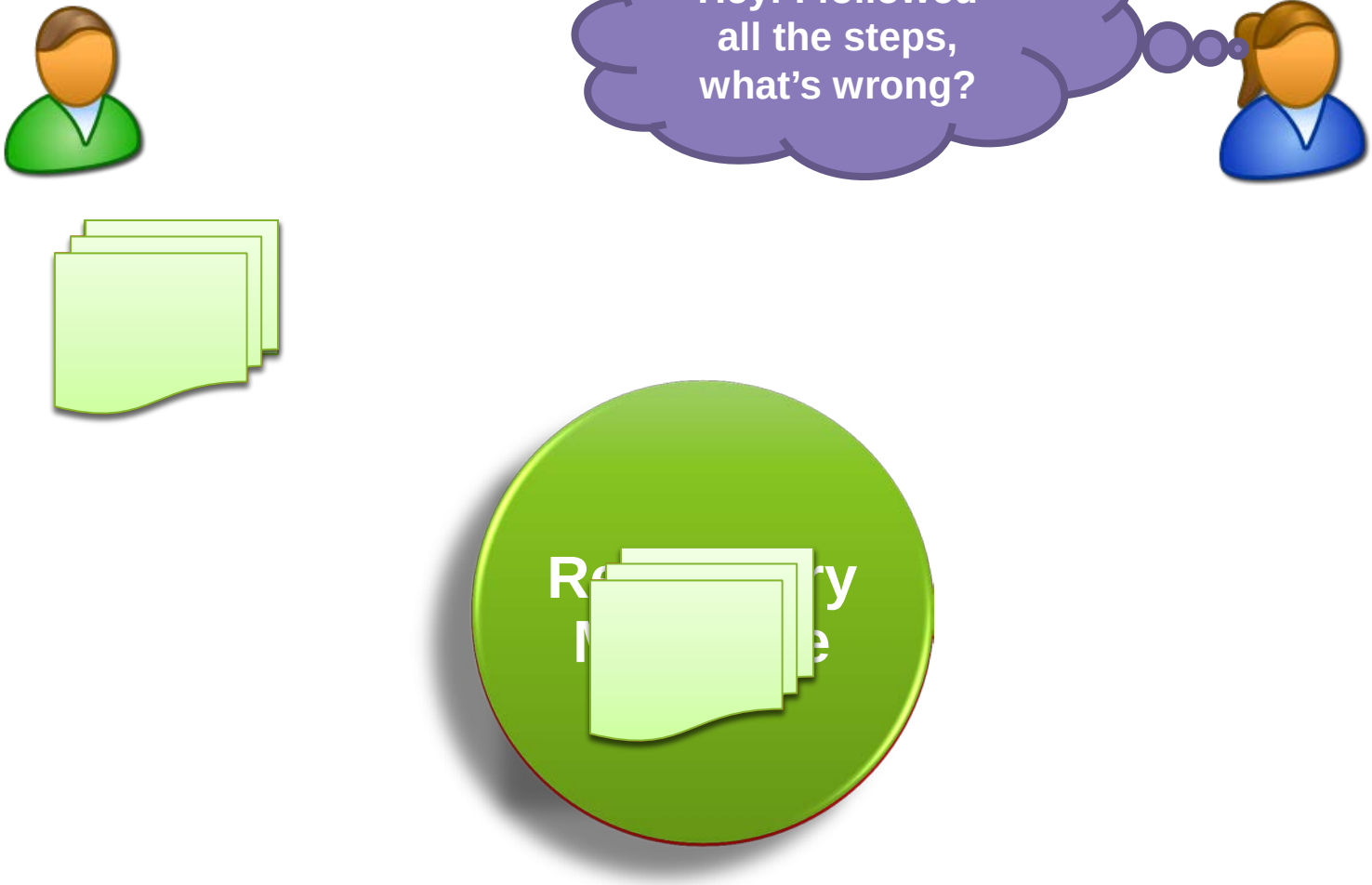
- Anyone who modifies the source files of the repository that make up the project. This can be a designer, a verification engineer, schematic entry tech, etc.

The Typical User Workflow

- Typical List of Workflow Steps For User
  1. Creates workspace from Source Control.
  2. Makes modifications
  3. Runs local tests or regression list to verify changes
  4. Update to latest changes on trunk
  5. Re-run tests to make sure everything still works
  6. Commits changes to source control.

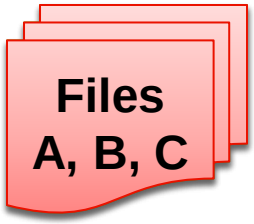
Jeffrey Wren

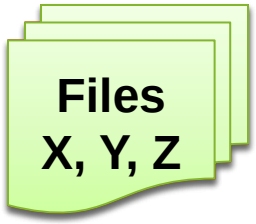
User Mistakes Break Repository

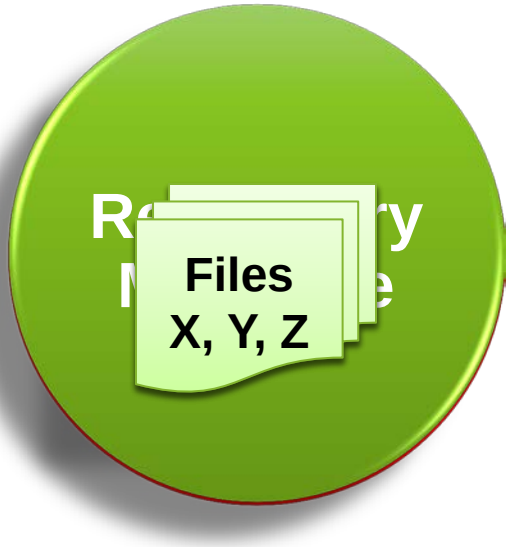


Jeffrey Wren

Race Condition Between Users

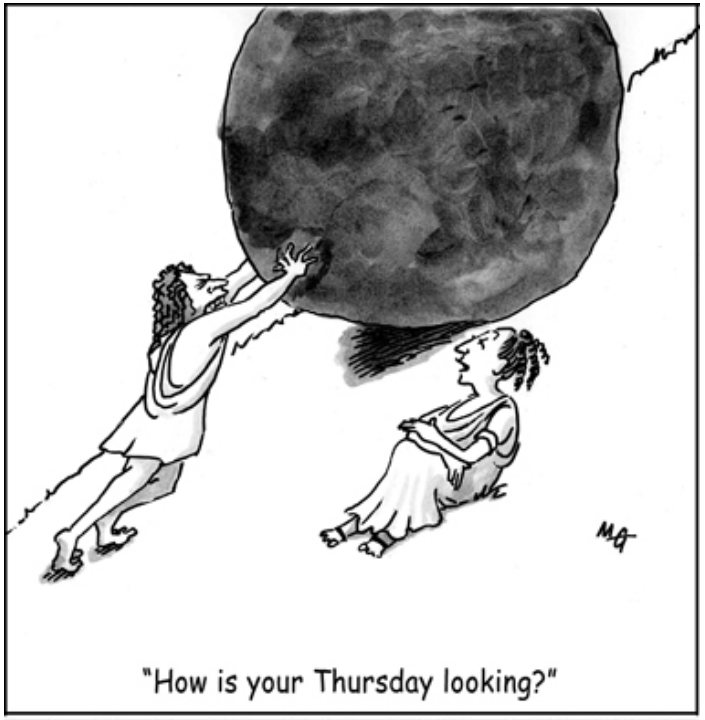






Release Manager Problem

- Focus is on the entire project
- Typical Workflow Steps for Release Manager
  1. Determine latest good code.
  2. Compile a workspace
  3. Run regression suit of tests
  4. Interpret results and resolve issues
  5. Label the files
  6. Notify project team
  7. Repeat

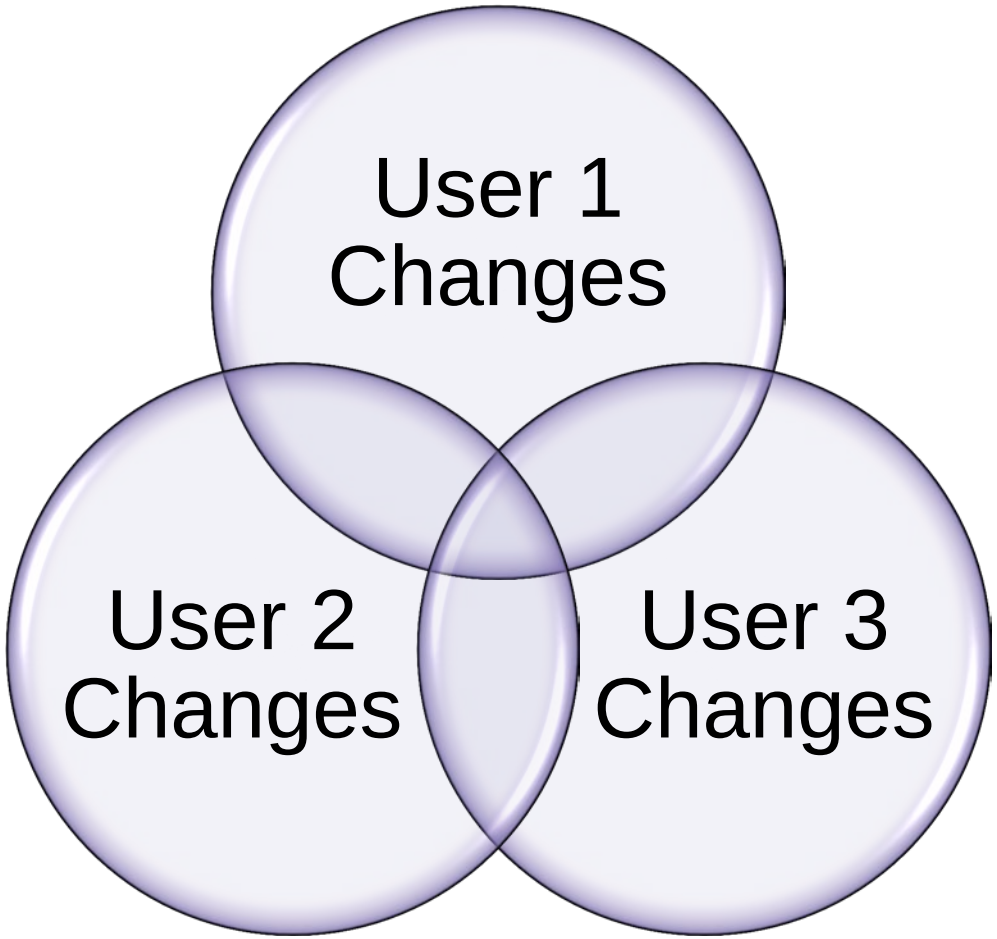


Reprinted from Funny Times / PO Box 18530 / Cleveland Hts. OH 44118
phone: 216.371.8600 / email: ft@funnytimes.com

Interpreting Results Issues

- What set of changes are causing compile/test errors
- Who is responsible for the fix?
  - Are they available (Vacation, out sick)?
  - Different time zone communication and resolution delays
- Once problem is found, should it be fixed?
  - Is fix required before moving on?
  - Should changes be backed out?
  - Does fix causes other areas of code to break?
- Resolution can be a moving target

Resolution Can Be Moving Target



Jeffrey Wren

Reproducibility Problem

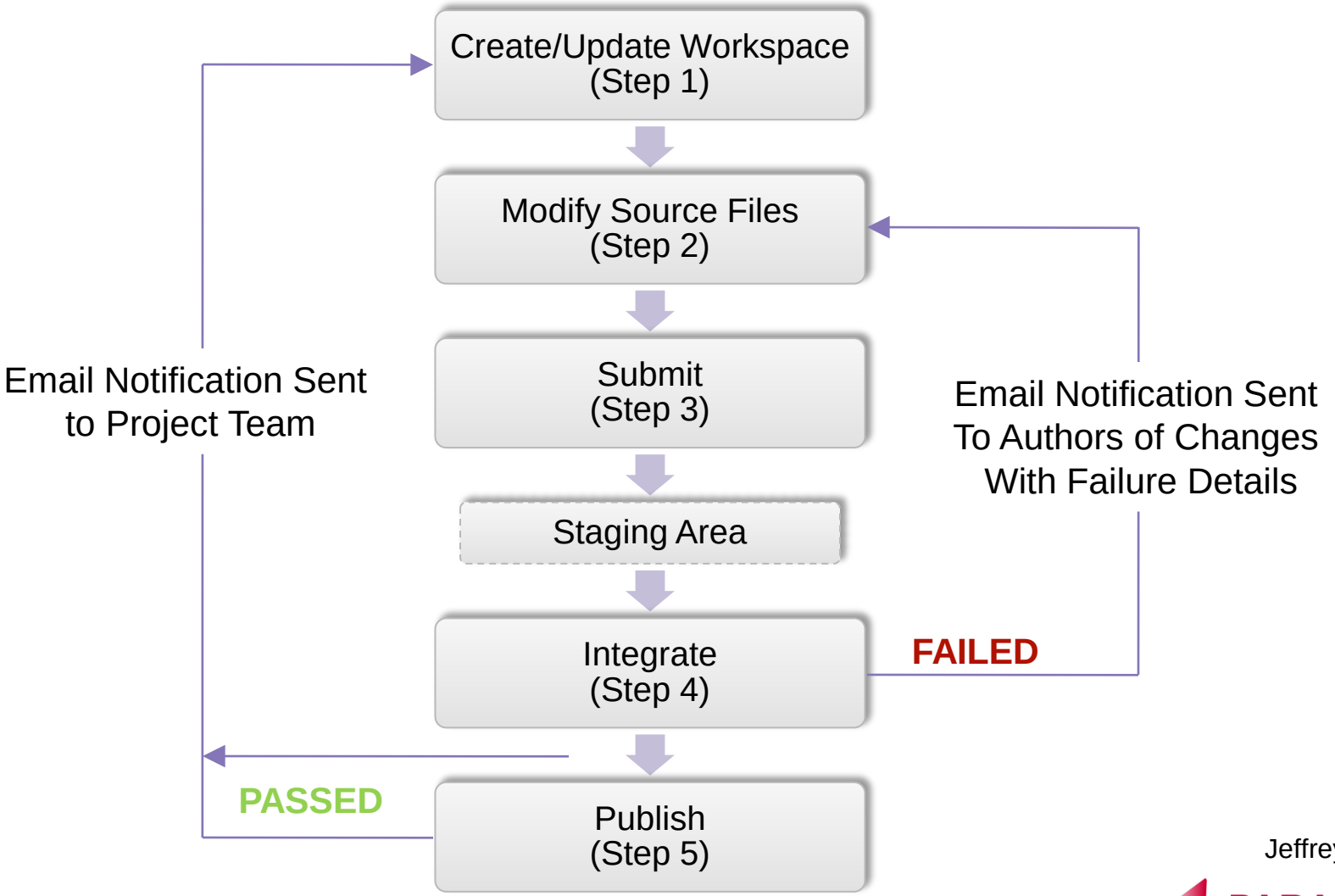
- Random verification environments based on seed
- Changes to the source can effect the random behavior
- Burden on user to create label that represents problem
  - Can be time consuming
  - Everything must be committed before labeling

What Makes Up a Good Release Flow

- Takes burden of acceptance testing out of the hand of the user
- Makes the release manager's job easier so that they can work on other project related tasks
- Makes the development area reproducible at any incremental stage of the design with minimal effort

Jeffrey Wren

Five Steps For A Good Release Flow



Jeffrey Wren

Create Workspace

- Command line tool or script that creates a workspace to a known working label

```
% release_tool create_sandbox sandbox_name
```

Where:

• release_tool	The command line executable
• create_sandbox	The subcommand call to the release_tool that performs the create
• sandbox_name	A user argument that defines what the name of the sandbox should be

Create Workspace Example Cont.

- Option to reproduce previous release

```
% release_tool create_sandbox \
  -label PROJ-REL_1_2_3 sandbox_name
```

Update Workspace Example

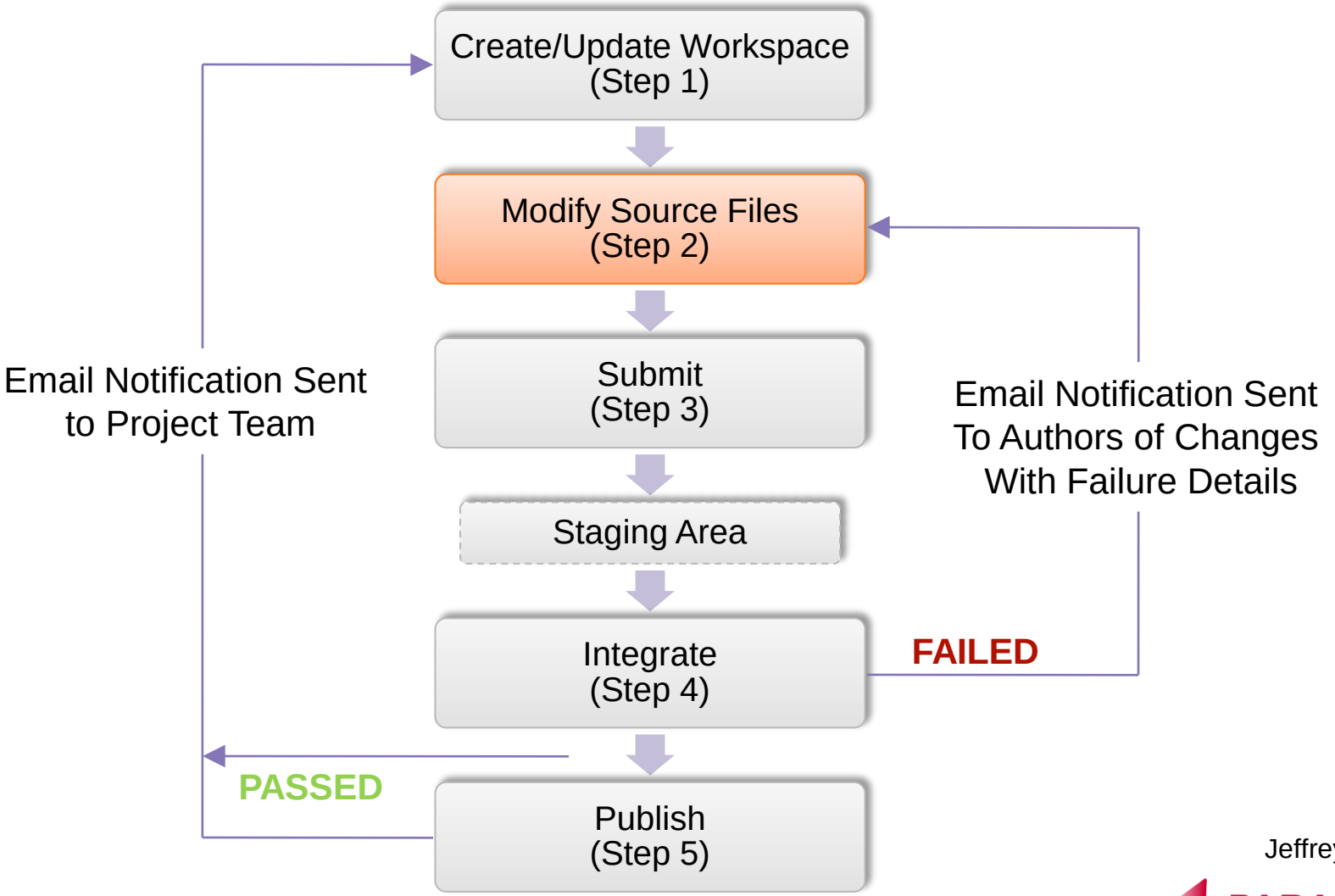
- Ability to update an existing workspace to the latest Published release

```
% release_tool update_sandbox
```

- Option to update to the latest Integrated release

```
% release_tool update_sandbox -latest_accepted
```

Five Steps For A Good Release Flow



Jeffrey Wren

Step 2: Modify Source Code

- User makes modifications to source code
- Commits changes to source control tool
- Two methodologies are available
  - Mainline/Trunk Approach
  - User Branch Approach

Mainline/Trunk Approach

- **Pros:**
  - Files can be locked to prevent concurrent development
  - Commits cannot be performed until changes are rectified

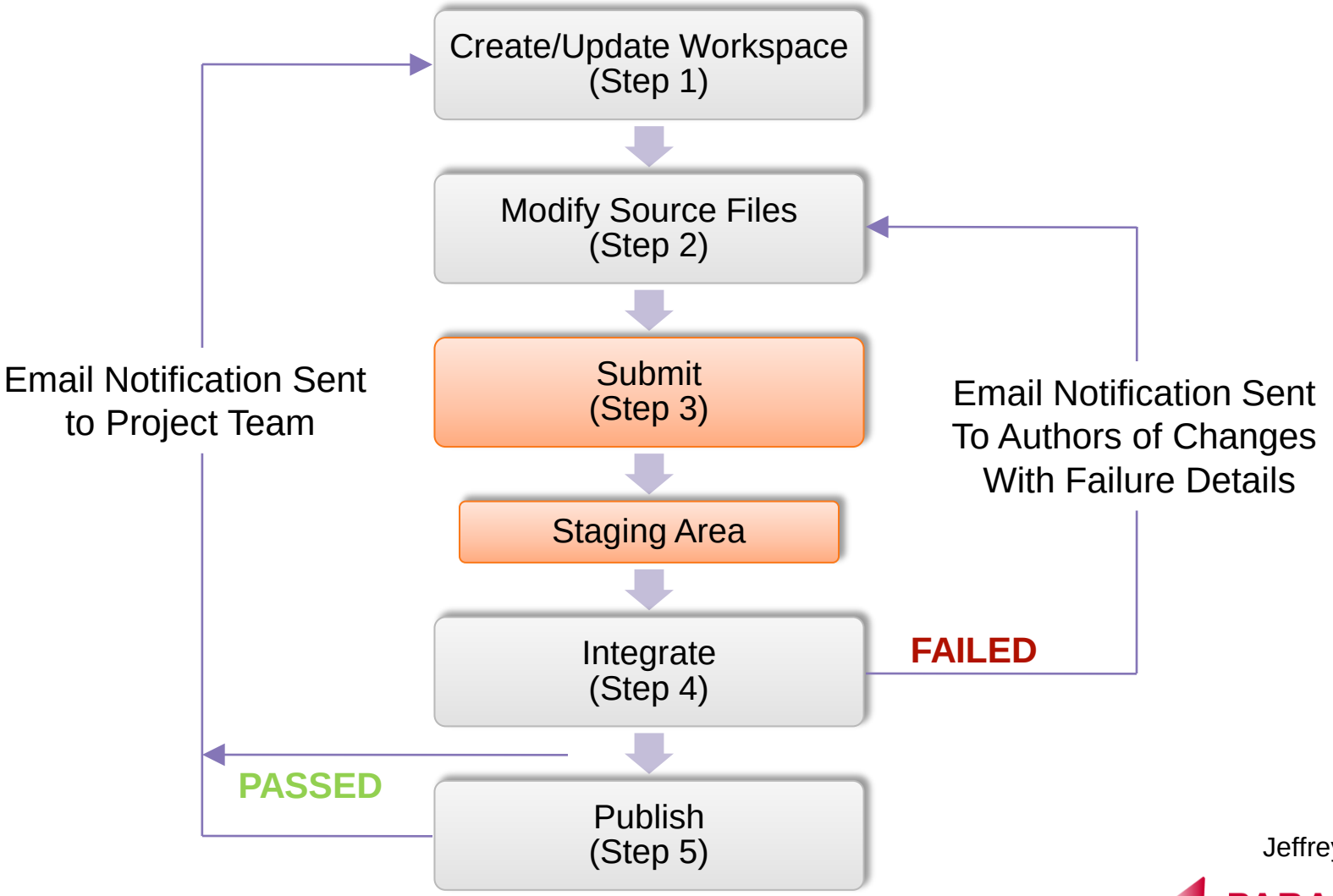
- **Cons:**
  - Two users cannot work on the same file at the same time. More of an issue with distributed work teams.
  - Blocking head problem

User Branch Approach

- **Pros:**
  - Commits to the user branch do not affect anyone else.
  - Commits can be done frequently.
  - Solves blocking head problem

- **Cons:**
  - Files cannot be locked. This can be an issue for binary files.
  - More merge operations are performed

Five Steps For A Good Release Flow



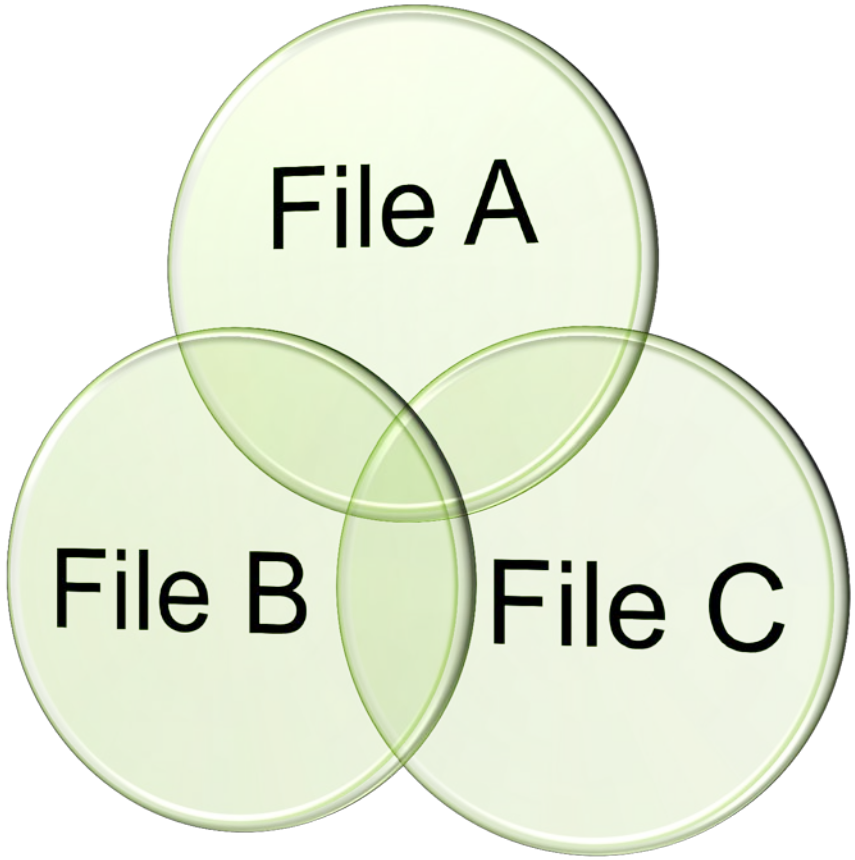
Jeffrey Wren

Step 3: Submit Changes

- A Submit is made up of 1 or more files
- Contains all file dependencies
  - Functional Change
  - Bug Fix

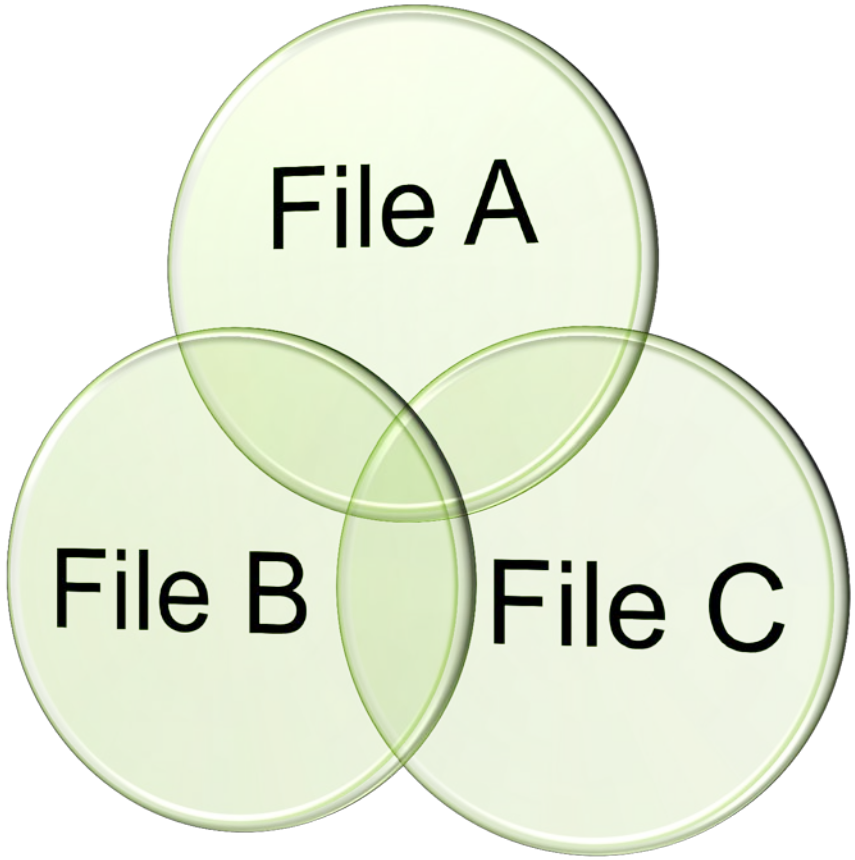
Jeffrey Wren

Submit Files As Change Group



Staging Area

Submit Files As Change Group

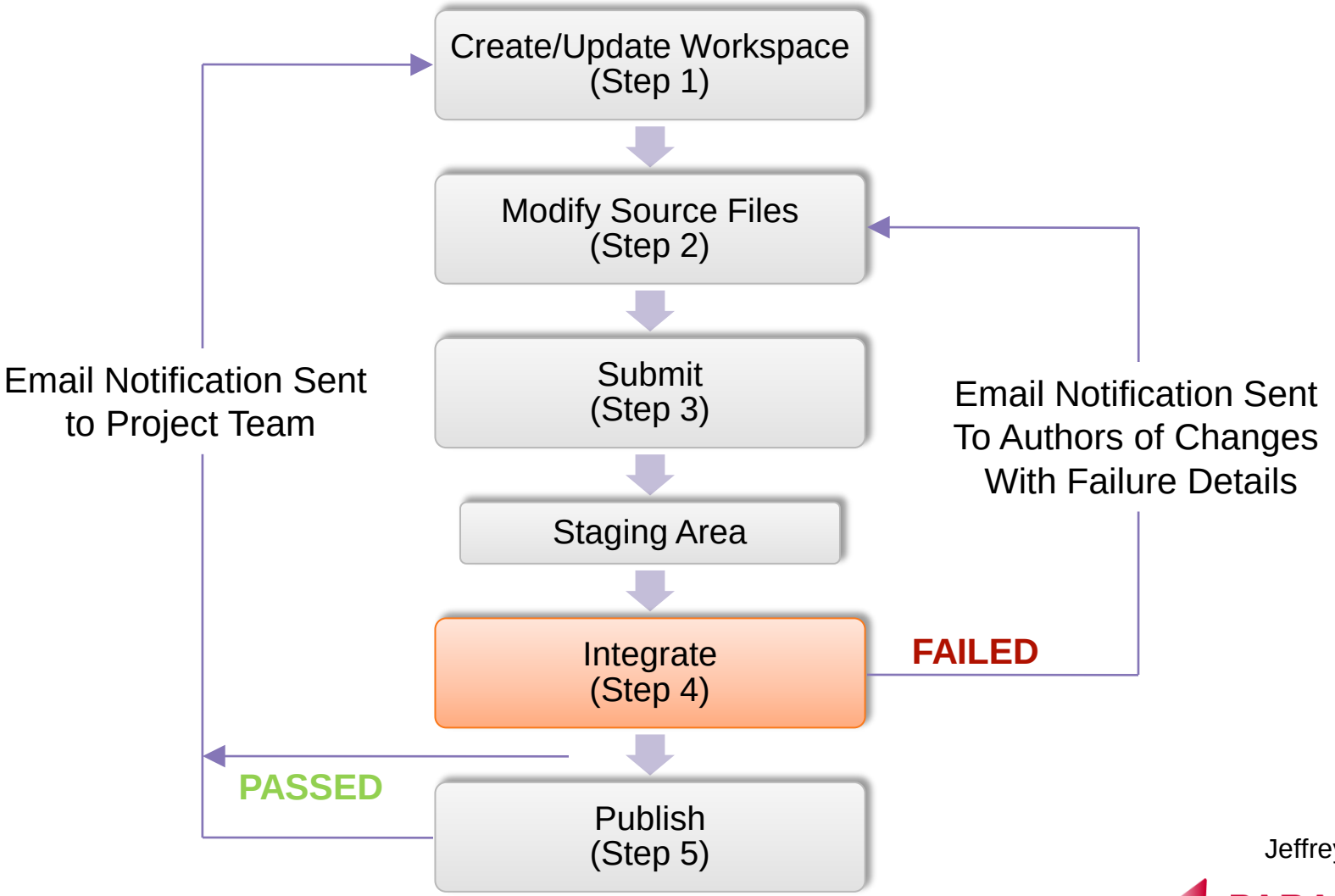


Staging Area

Submit Changes Cont…

- Depending on source control tool the Submit information can be:
  - Listing of each file along with appropriate source control version
  - Label that marks the file versions that have changed
- Information is stored in a staging area (A directory, or file)

Five Steps For A Good Release Flow



Jeffrey Wren

Step 4: Integrate the Submit Groups

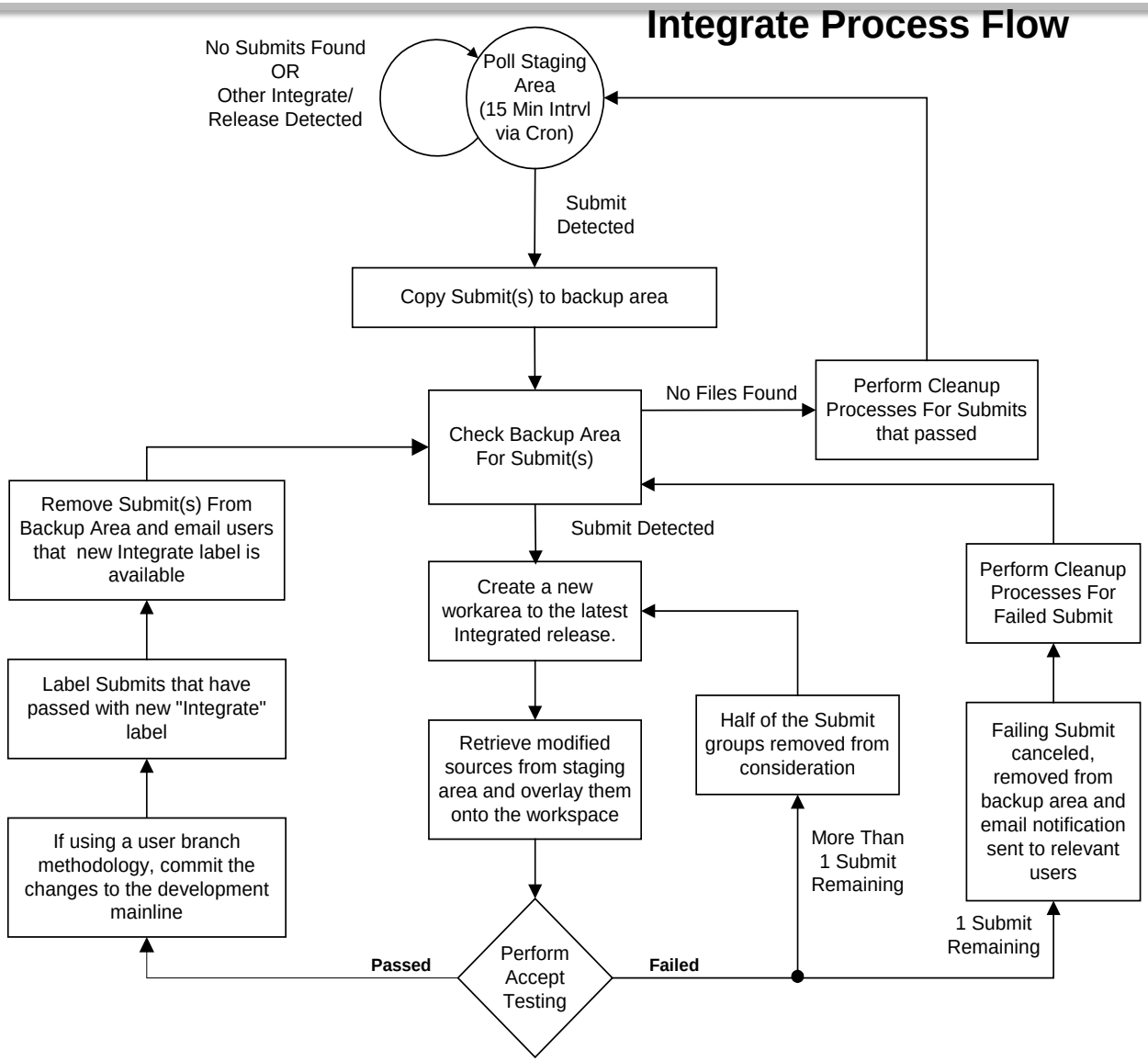
- Heart of Release Flow
- Keeps code in a known working state
- Made up of acceptance testing
  - Compile of each relevant testbench
  - Simple test from each testbench
- Everything must pass in order to be integrated
- Can be automated with binary search algorithm

Step 4: Integrate the Submit Groups Cont…

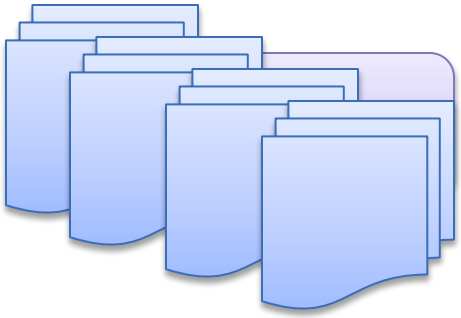
- Email can be sent based on results
  - Team announcement on passing integrate
  - Owner of submit contacted on failure

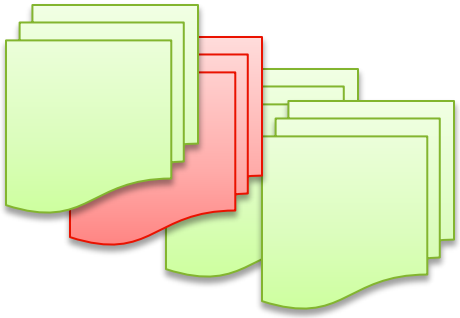
- Solves reproducibility problem:
  1. Create workspace to the Integrate label at time of failure
  2. Overlay changes detailed in Submit group

Integrate—Step by Step (Automate)



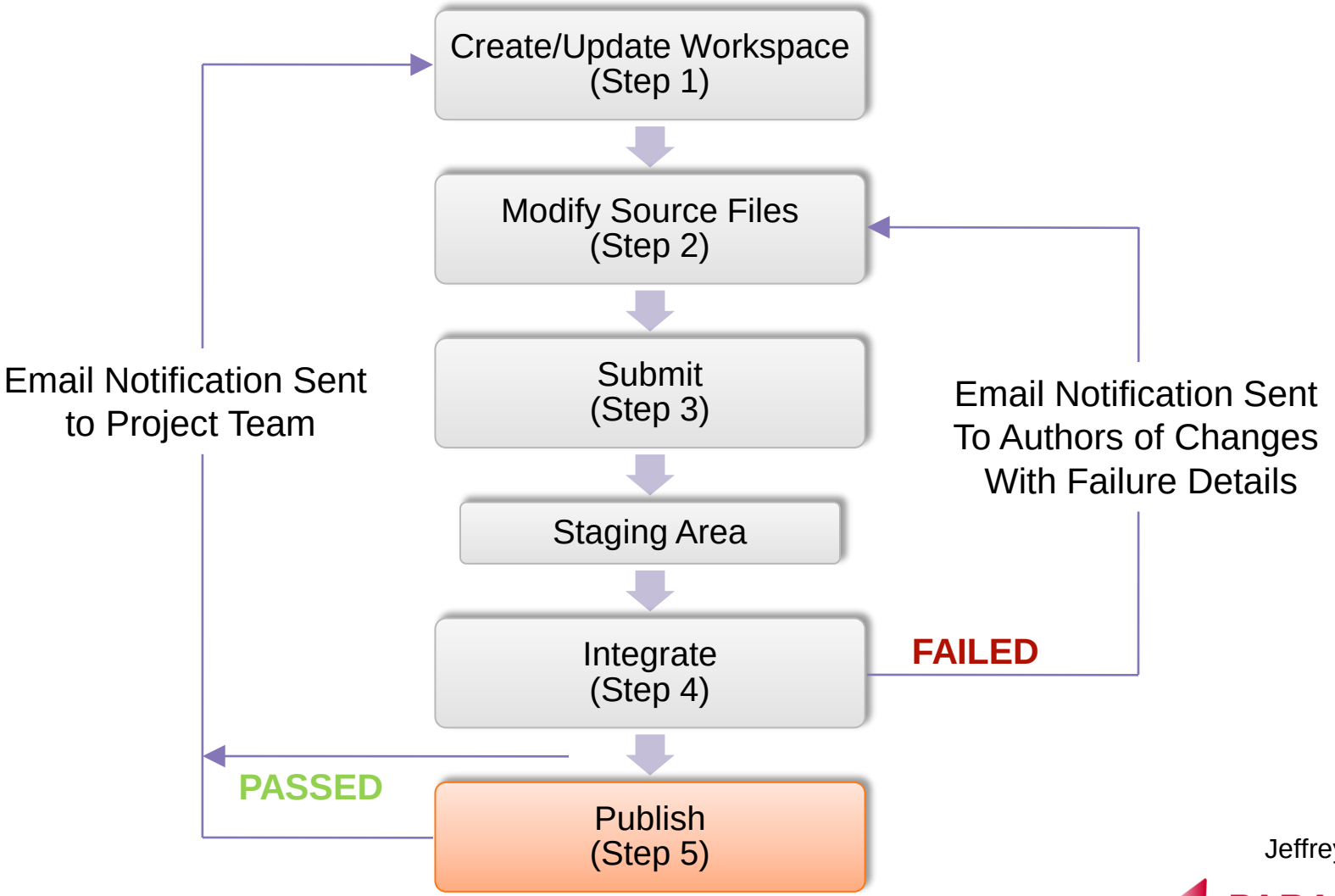
Jeffrey Wren






Clean
Workspace

Five Steps For A Good Release Flow



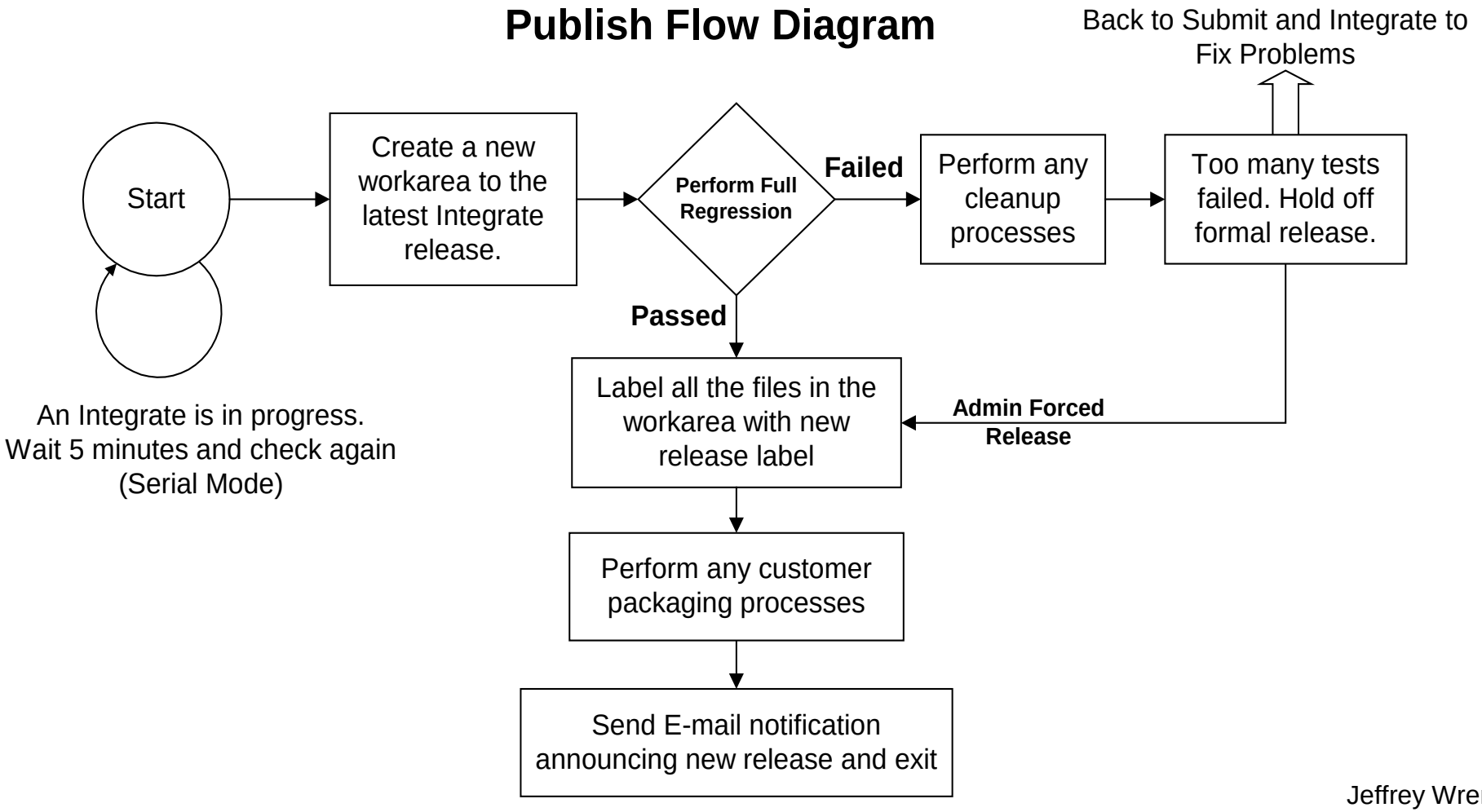
Jeffrey Wren

Step 5: Publish

- The full suite of testing is performed
  - Nightly Regression
  - Weekend Regression
  - Full Regression
- Blessing of the label applied during Integrate
- New label is applied only on passing regression
- Concurrent Integrates and Publishes for large projects needed

Publish—Step by Step (Automate)

Publish Flow Diagram



Jeffrey Wren

Case Study

- Company X utilized in-house solution called the Release Process Tool (RPT)
- Source control tool was ClearCase by IBM Rational
- RTP tool acted as wrapper for commits
- It solved the problem of reproducibility
- Still had problems regarding user and release manager

Company X's RPT Tool vs. 5 Step Flow

- Acceptance Testing
  - **RPT**
    - Burden placed on user, but not enforced.
    - Release manager responsible for debug if something broke
  - **5 Step Flow**
    - Burden taken away from the user.
    - Changes only integrated into release if acceptance testing passed.
    - Release manager no longer needed to debug inter-dependencies between multiple user changes
  - **Potential Savings**
    - 1 man hr per commit

Company X's RPT Tool vs. 5 Step Flow

- Feedback to user if changes successfully integrated into release
  - **RPT**
    - Dependent on release manager: Typical 1 day, Worst case 1 week
  - **5 Step Flow**
    - Dependent on time to perform acceptance testing and binary search for failing Submit.
    - For Company X this was a minimum of 2 hrs, worst case 1 day.
  - **Potential Savings**
    - 2 man hrs
    - Based on fact user gets quick feedback, saving time on issues not propogating to other areas of the project.

Company X's RPT Tool vs. 5 Step Flow

- Release manager time to perform release
  - **RPT**
    - Typical: 0.5 day
    - Worst Case: 3 days
  - **5 Step Flow**
    - Automated
    - Release manager is free to perform other tasks
  - **Potential Savings**
    - 4 man hrs per release

Company X's RPT Tool vs. 5 Step Flow

- Nightly Regressions
  - **RPT**
    - Often would crash or not be run
      - `User didn't properly submit needed changes`
      - `Incompatibility between user changes`
    - A loss of feedback as to the state of the project of 1 day
  - **5 Step Flow**
    - Automated acceptance testing executed throughout the day
    - Nightly regressions executed cleanly
    - Consistent daily feedback as to the state of the project
  - **Potential Savings**
    - 1 day of project productivity per week.
    - Based on estimate that nightly regressions would fail at least once a week

Company X's RPT Tool vs. 5 Step Flow

- Time between Publishes
  - **RPT**
    - Dependent on release manager. Best case, 1 a day.
  - **5 Step Flow (Automated)**
    - Consistent incremental Submit releases based on acceptance testing
    - Daily release based on nightly regression
    - Weekly release based on full weekend regression.
  - **Potential Savings**
    - Invaluable.
    - Helps keep project on schedule.

Conclusion

- The 3 major problems for release management to solve were presented
  - The user problem
  - The release manager problem
  - The reproducibility problem
- A robust 5 step solution was detailed that solves the problems and which can be automated
- Case study was provided showing how Company X made significant performance increases by moving to an automated 5 step flow
- The larger the team, the greater the cost reward
- Improvements could very well be $Priceless if time to market is reduced allowing a company to beat their competition to market

Resources

- Free Open Source Software Tool ReleaseWorks®
  - Utilized by Company X to implement 5 step flow
  - Available on SourceForge
    - http://releaseworks.sourceforge.net
  - Currently supports
    - ClearCase
    - CVS
    - SVN (comming soon)