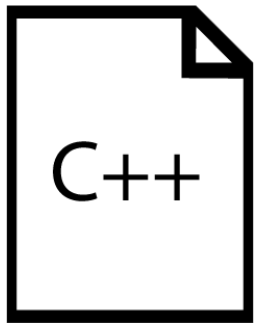
# Speeding Up Cross Platform Testing – A Case Study

Jeffrey.Weston@Microsoft.com



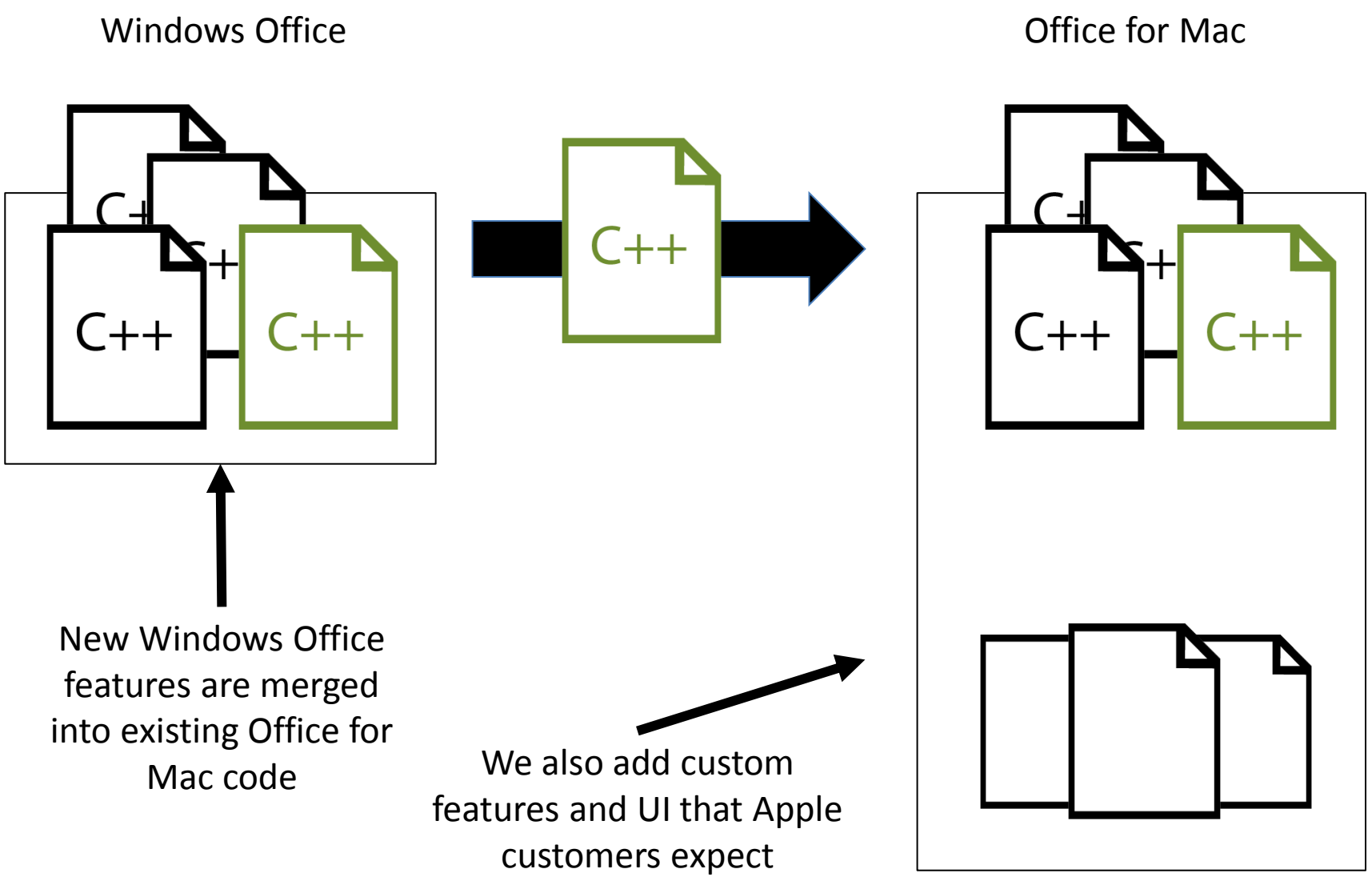
# About me

- Jeffrey Weston
- Senior Software Engineer in Test @ Microsoft
- 9 Years in the Office for Mac Group

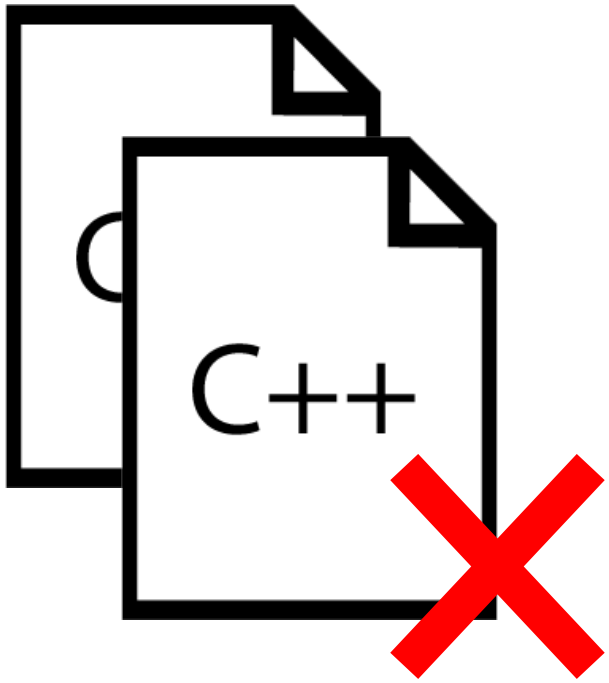
# Overview

1. Making **Office for Mac** with Ported Code
2. How Porting code lead to **“random”** bugs
3. Using **Air Crash investigation techniques** to understand where bugs come from
4. The **bug patterns** we found
5. Recommendations to improve our **Testing Efficiency**

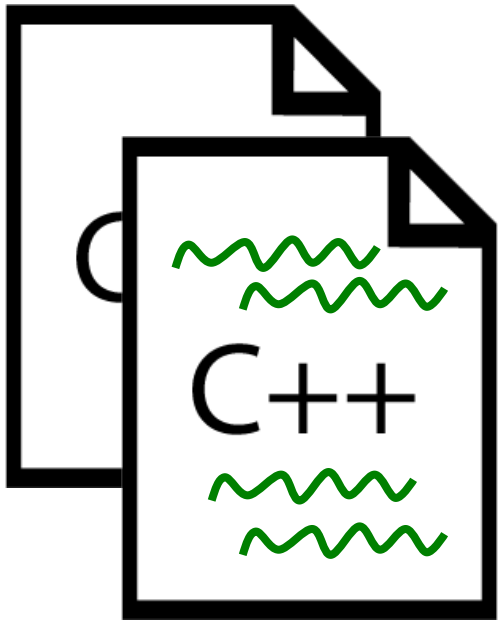
# How we make Office for Mac



But adding new files or just replacing existing ones can **break** the build and cause **compile errors**.




Developers must **manually fix up** the code to properly integrate it into **Office for Mac**.

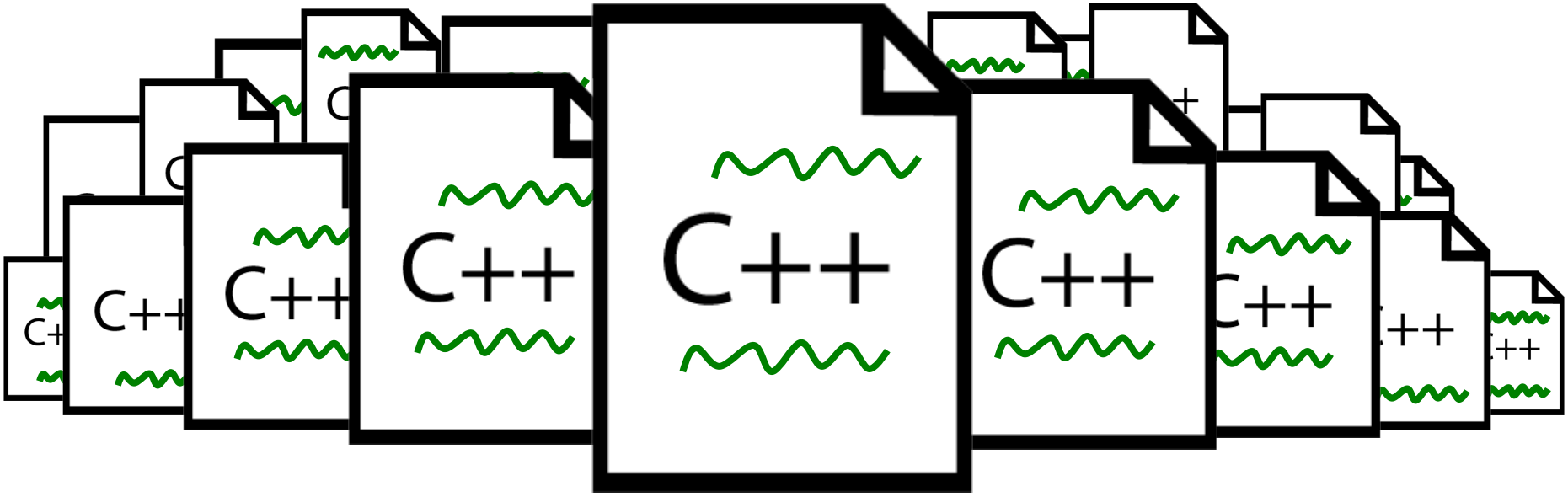


This updating of code happens to lot of files…

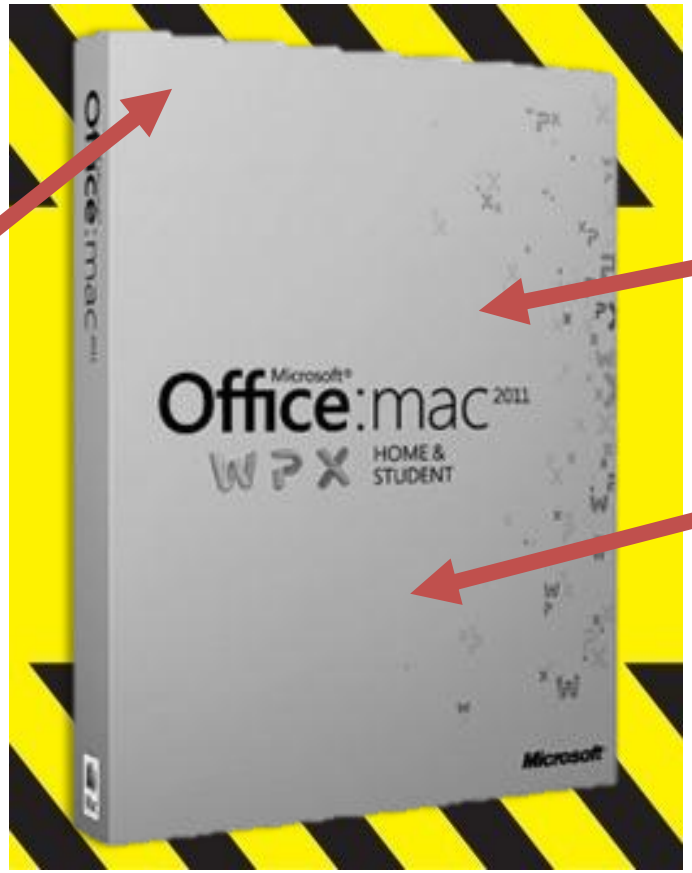
it can take **weeks** to **months** to complete…

all the while the build does **not compile**

and is ***untestable!***



Finally a build is produced…




**Crash Here**

**Memory Leak**

**Error Here**

The bugs are seemingly in

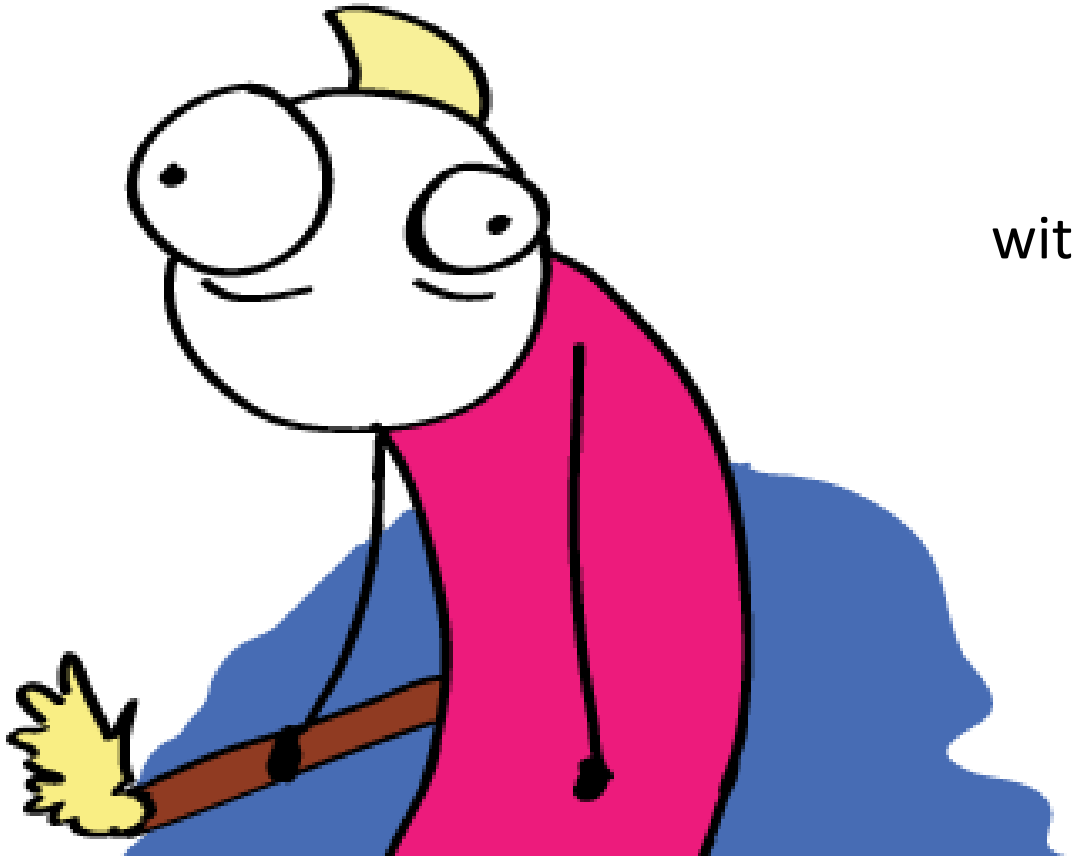
**random** places across the product!

# So the only real option is to…

# But that’s expensive…

time consuming…

boring…

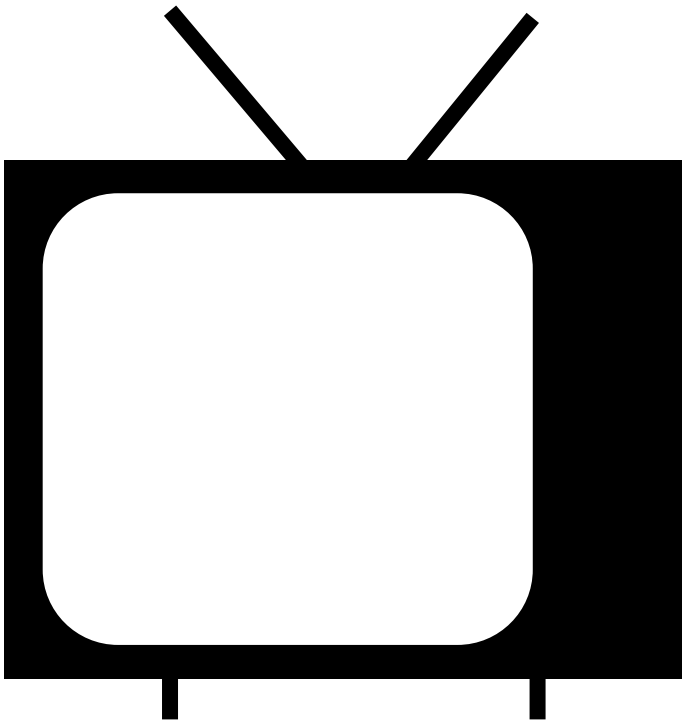


with indeterminate results…

What to do?

# **TV** has the answer***!***

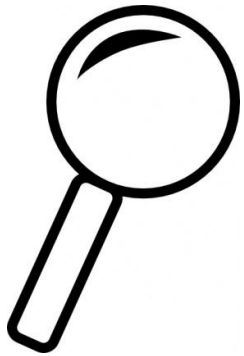
- *Seconds from Disaster* (National Geographic Channel)
- *Nova: Crash of Flight 447* (PBS)



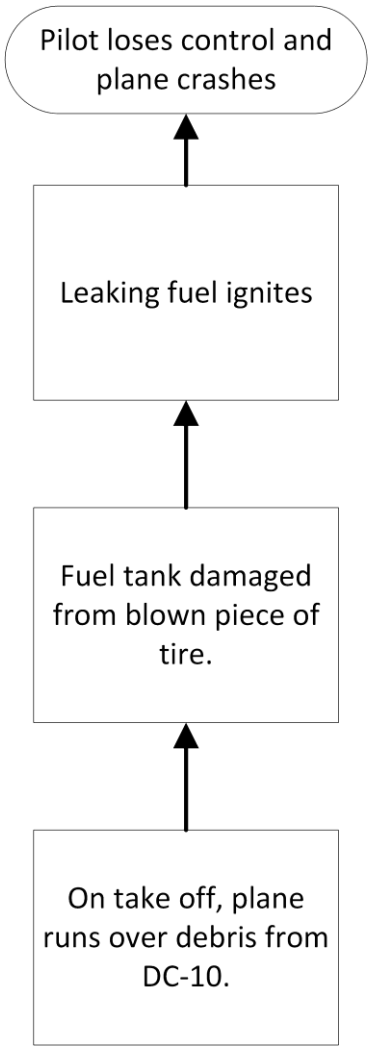
# Air France Flight 4590
## (Concord Crash)

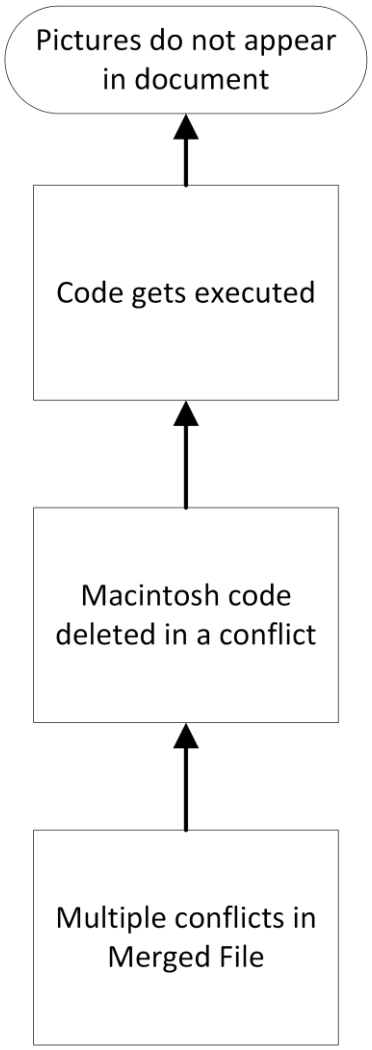
# **Bugs** like Plane Crashes are not Random

- Occur based on initial conditions and a series of events.
- Use root cause analysis to uncover conditions and events on various bugs.
- Look for patterns in the bugs.
- Propose recommendations to prevent the bugs or find them quicker.



# Chain of Events

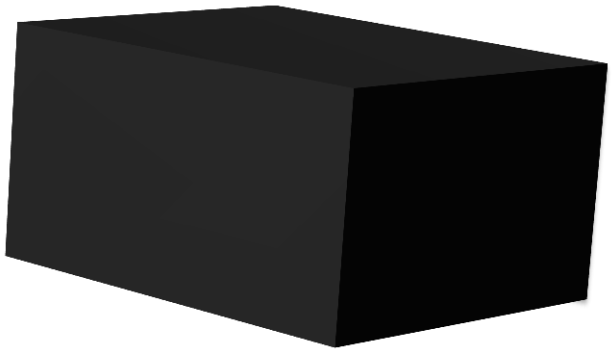



Office

# Logs

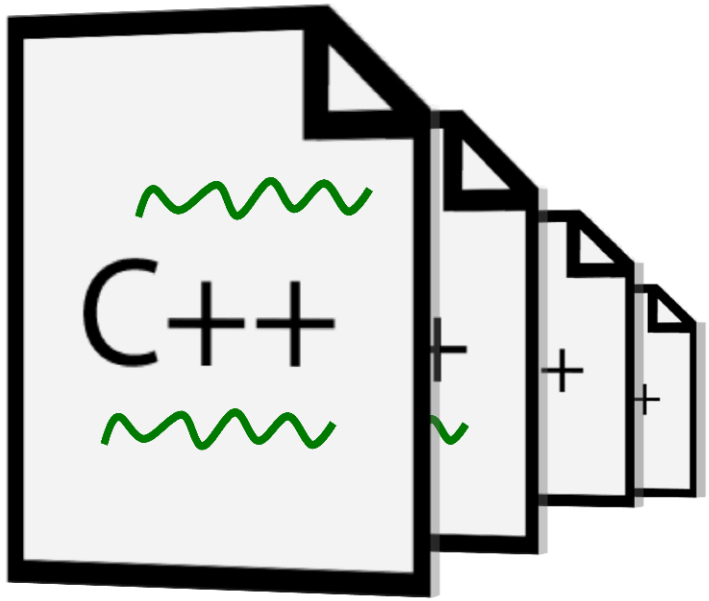
**Airplane Black Box**



Used to understand how the Pilots reacted during the incident.

≈

**Source Control History**




Work backwards from the change that fixed the bug to change that **introduced** the bug.

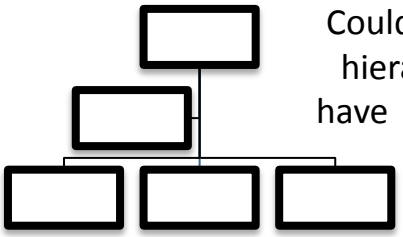
# Contextual Data

The Air France investigators asked a lot of contextual questions.

from

What was the position of the Cockpit Chair at take off?

to



Could the management hierarchy of Air France have contributed to the crash?

**Code Context**

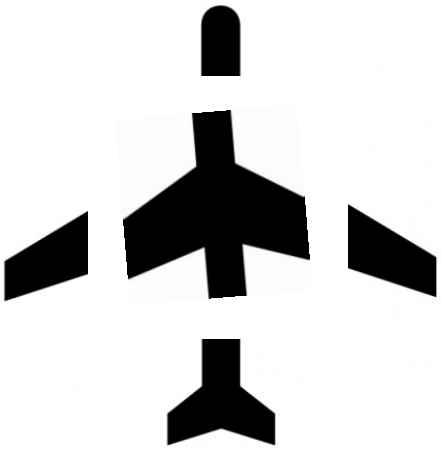
- Was the source file ported?
- Did it have Mac specific code?
- Does it interop with Mac specific Code?

```
NewAssert(cache.isEmpty == fTrue);

if (FInvalid(cache)
        err = errNone;
else
        err = displayLine(theText, CLIPPED);
NewAssert(err == errNone);
```

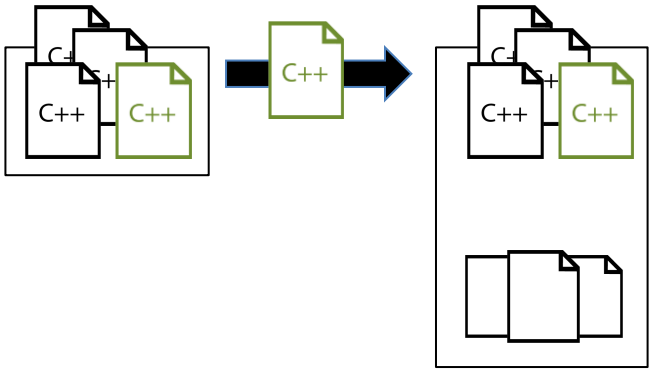
- Could it be found by regression automation?
- Could the area be targeted for testing in advance?

# Reconstruction

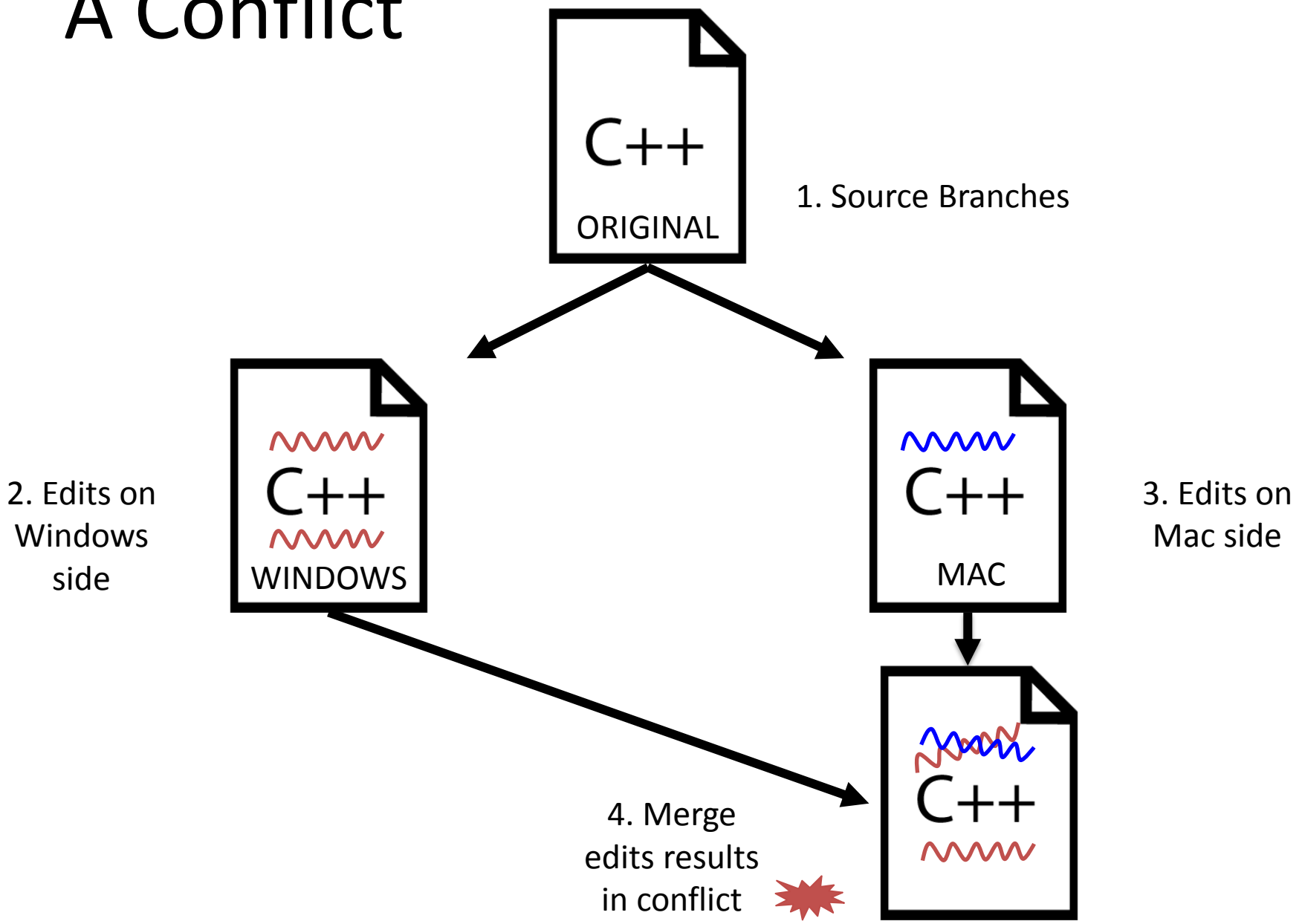


On the Concord, investigators reconstructed the Wing to see how it was damaged.

Reconstruct the Merge Process to see the developer perspective.



# A Conflict

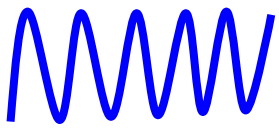

C++
ORIGINAL
1. Source Branches
2. Edits on Windows side
C++
WINDOWS
C++
MAC
3. Edits on Mac side
C++
4. Merge edits results in conflict

```
>>>> ORIGINAL ++++++++++++++++++++++++++++++++++
        Assert(cache.isEmpty == fTrue);
        lserr = displayLine(theText, CLIPPED);
        Assert(err == errNone);

==== MAC DEV ++++++++++++++++++++++++++++++++++
        Assert(cache.isEmpty == fTrue);
        err = displayLineMac(theText, CLIPPED);
        Assert(err == errNone);

==== WINDOWS DEV ++++++++++++++++++++++++++++++
        NewAssert(cache.isEmpty == fTrue);

        if (FInvalid(cache)
                err = errNone;
        else
                err = displayLine(theText, CLIPPED);
        NewAssert(err == errNone);
<<<< END ++++++++++++++++++++++++++++++++++++++
```



Mac specific API call was deleted.

## Chain of Events

1. Ported code is brought over to Mac
2. Code is changed for Mac
3. File gets merged again
4. Conflict occurs
5. Developer chooses Windows Code block
6. Mac code is removed.

## Logs (i.e. Source Control)

1. Identify the change that fixed the bug.
2. Find the original check-in that introduced the bug

## Contextual Data

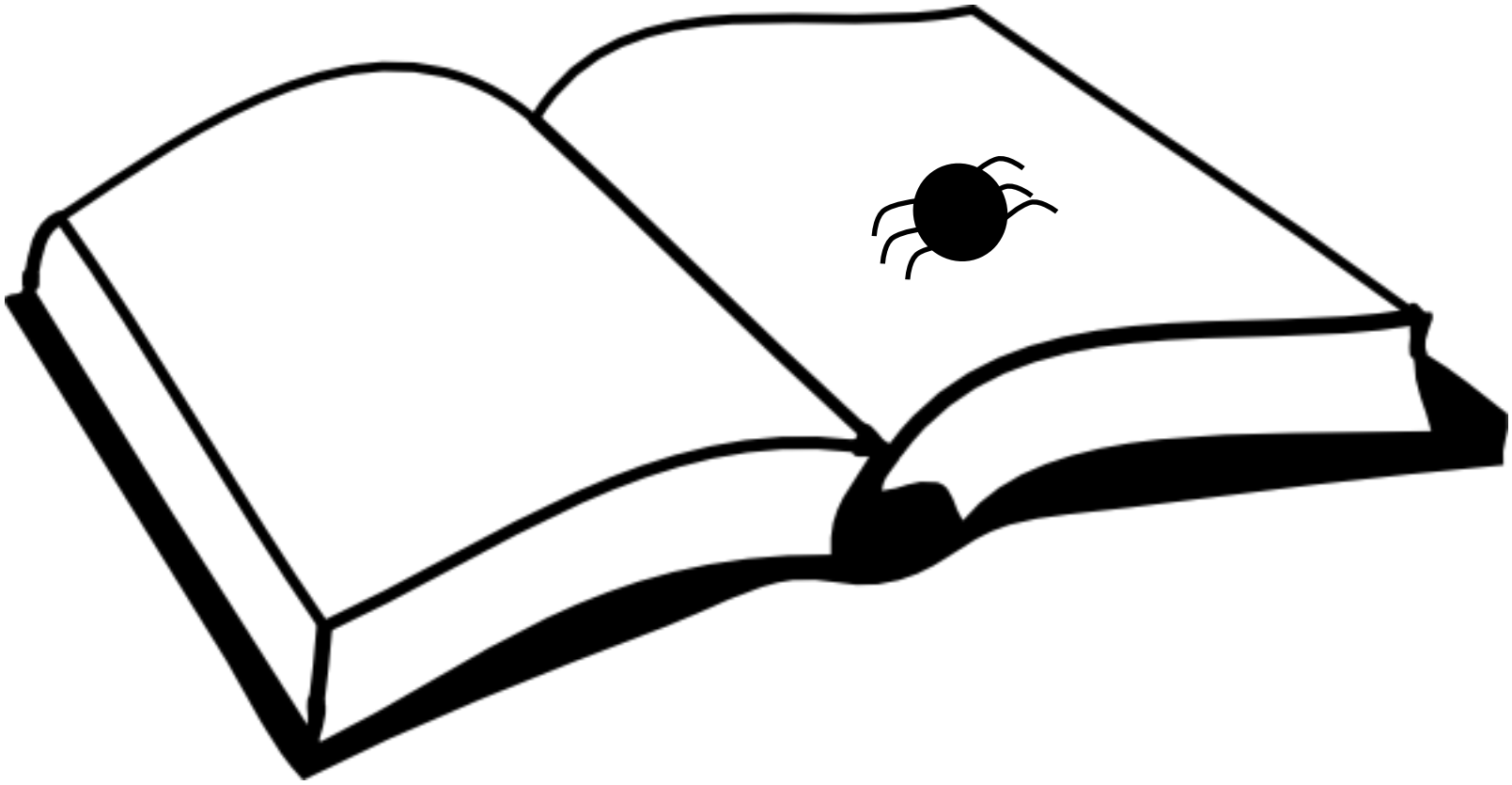
1. Mac specific code was not obvious.
2. Often chooses WINDOWS changes.

## Reconstruction of merge

- 60 other conflicts in that file.

Continue Collecting Data for each bug to find patterns.

# Construct the Story of the Bug



# Bug Patterns Found

- Incorrect conflict resolutions.
- Mistakes when integrating Office for Windows changes into Macintosh code.
- Mistakes integrating Office for Windows changes into older legacy code.
- Bugs in Office Windows ported over to Mac.

# Designing Recommendations

| Stage | How to *prevent* bugs from getting checked-in? | How to *find* bugs more efficiently? |
|---|---|---|
| **Design Phase (Pre-Merge)** | Prepare the code to reduce the number of conflicts before the merge starts. | Mark up unmarked Mac code so developers can clearly see the Mac code. |
| **Implementing (During Merge)** | Review each conflict resolution specifically looking for dropped Mac code. | Have Test Code Review the conflicts after check-in, looking for dropped Mac code. |
| **Testing Phase (Post Merge)** | N/A | Use code coverage analysis to determine whether conflicts resolutions were covered. |

Come up with Recommendations to break the chain in **as many** places as possible.

# Results for Mac Office

1. Implemented scripts to reduce conflicts by 50%.
2. Reduced bugs by training Developers to be aware of these patterns.
3. Developers changed their check-in procedures to make conflicts easier to code review.
4. Convinced Management to make large investments in tooling like Code Coverage.

# Summary

- Faced a prospect of **random** bugs and expensive ways of testing for them.

- Using the Root Cause Analysis of Air Crash Investigation techniques
  1. Piece together the Chain of Events
  2. Through logs, contextual data and reconstruction
  3. Repeat on a sampling of bugs
  4. Look for similarities to uncover patterns

- Come up with Recommendations to **break the chain** in as many places as possible.