Jet Propulsion Laboratory
California Institute of Technology

Flight Software Problem Failure Report Process

Amalaye Oyake{@jpl.nasa.gov}

Jet Propulsion Laboratory
California Institute of Technology

Copyright 2012 California Institute of Technology.
Government sponsorship acknowledged.

What is a PFR?

- **A PFR can be described as the documented history of a Flight Software problem, which traces the history of the problem from its origination, through verification, through complete resolution and testing of the resolution.**
- **It provides a historical artifact on specific software problems, which may manifest themselves in various settings.**
- **It is a document that can be accessed by mission team members.**
- **This talk focuses on Flight Software PFRs.**

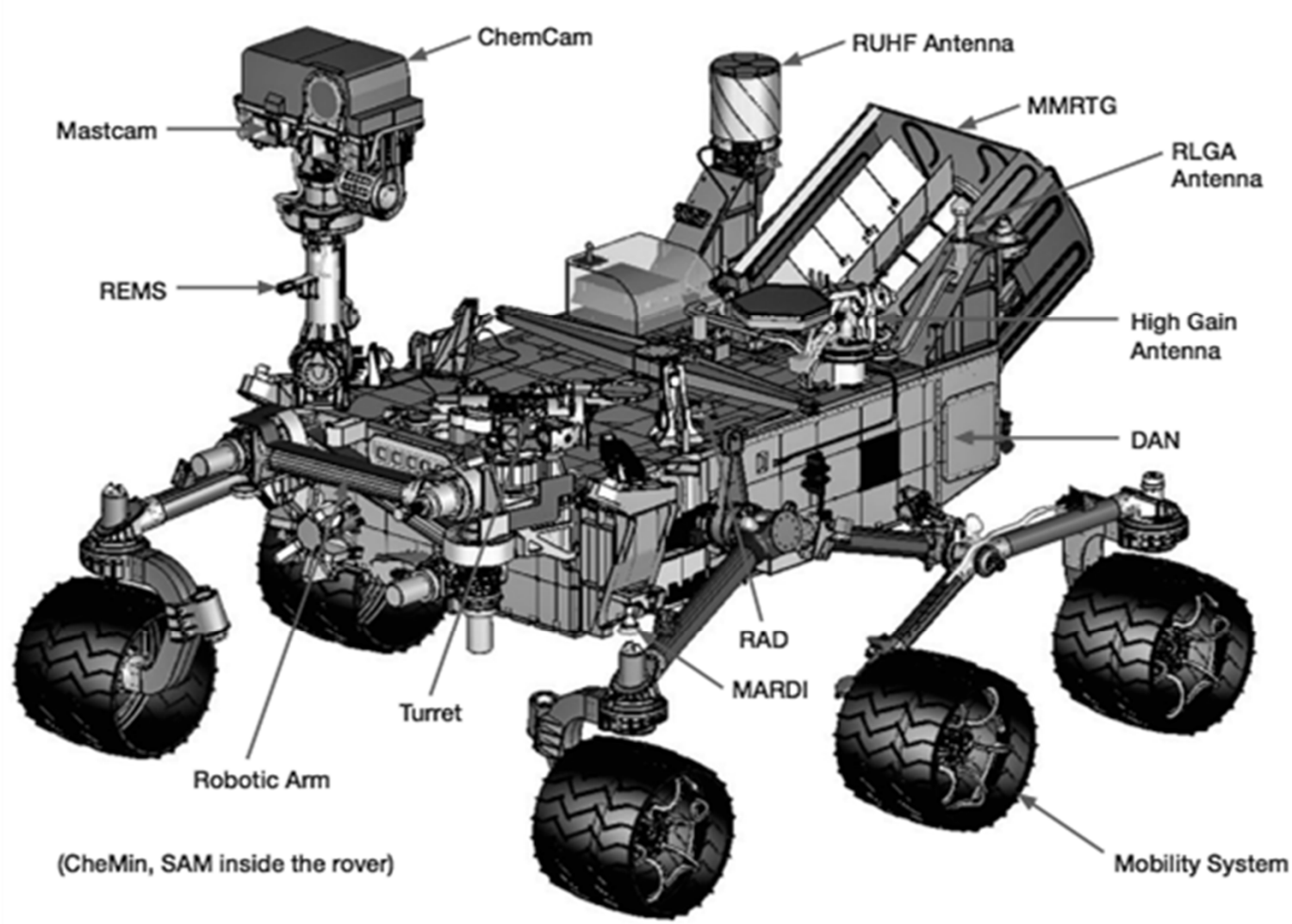
What is a PFR?

- **There are two important aspects:**
  - **The Software Problem, or Failure, or Anomalous Behavior**
  - **The complete reporting of these incidents.**
- **Unresolved software issues can pose a serious risk to a mission!**
- **Poorly understood software issues can pose a serious risk to a mission.**

The Importance of PFRs

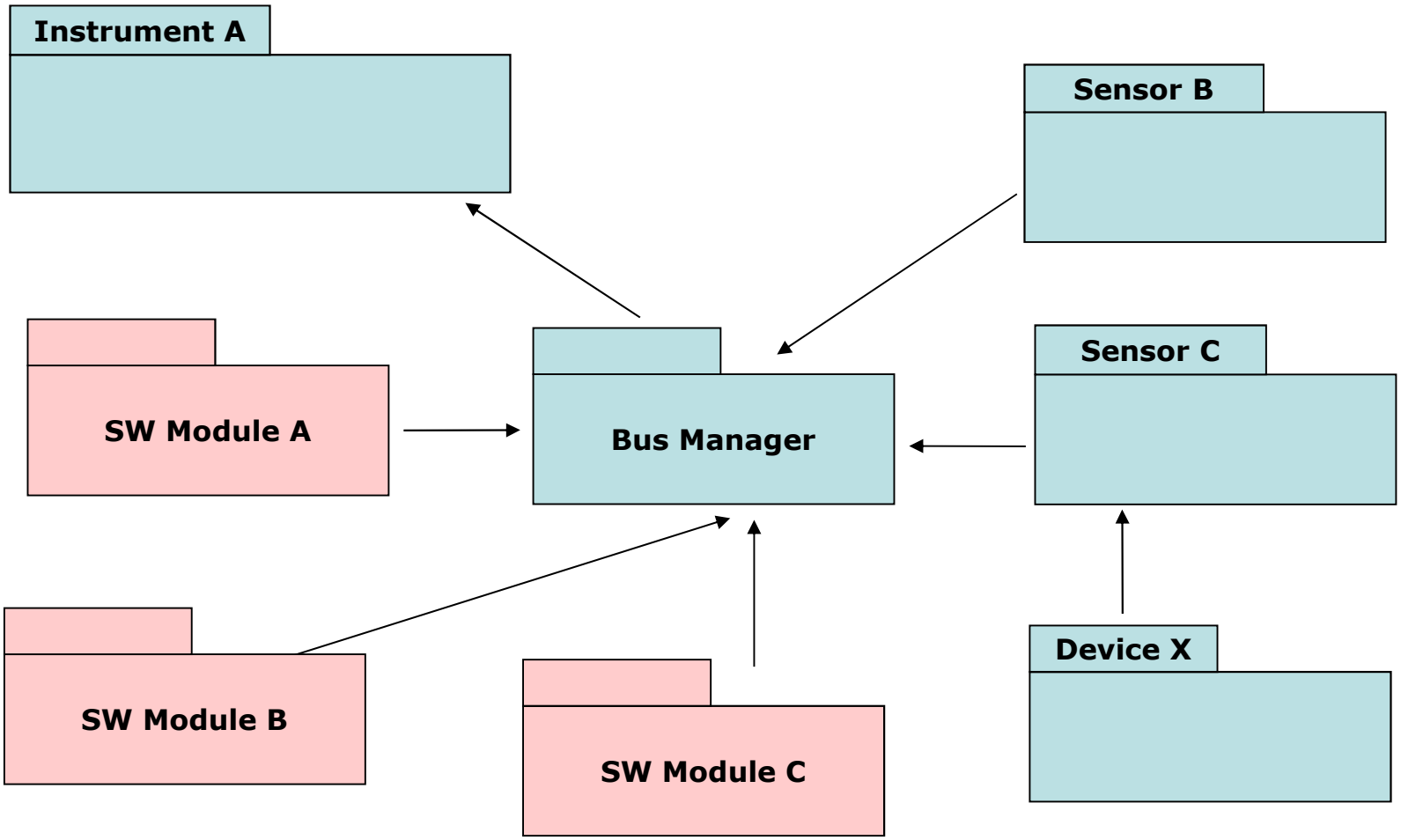
- **Software development and testing would be very difficult without tracking software problems.**
- **Problems manifest themselves in various settings or venues.**
- **If these problems can be tied to a specific issue or conditions, then the problems can be understood if there is a documented fix or workaround – The PFR.**
- **In some cases there may not be a fix at all or a fix may not be needed (*Use-As-Is or Not-A-Problem)*!**
- **It provides a measure of the reliability software and the effectiveness of the code coverage.**

Where do Software Problems Occur?



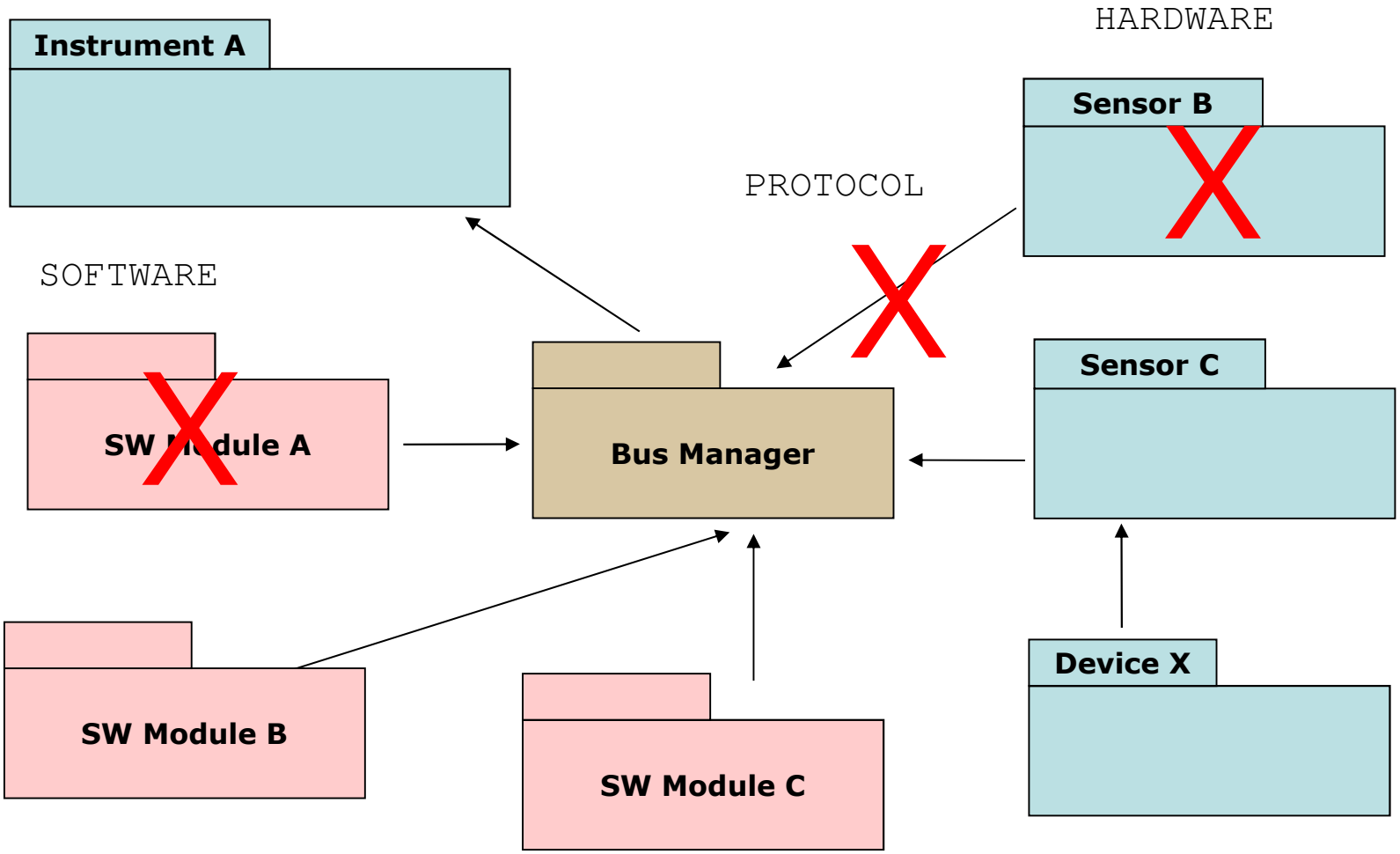
EVERYWHERE!!!

Where do Software Problems Occur?



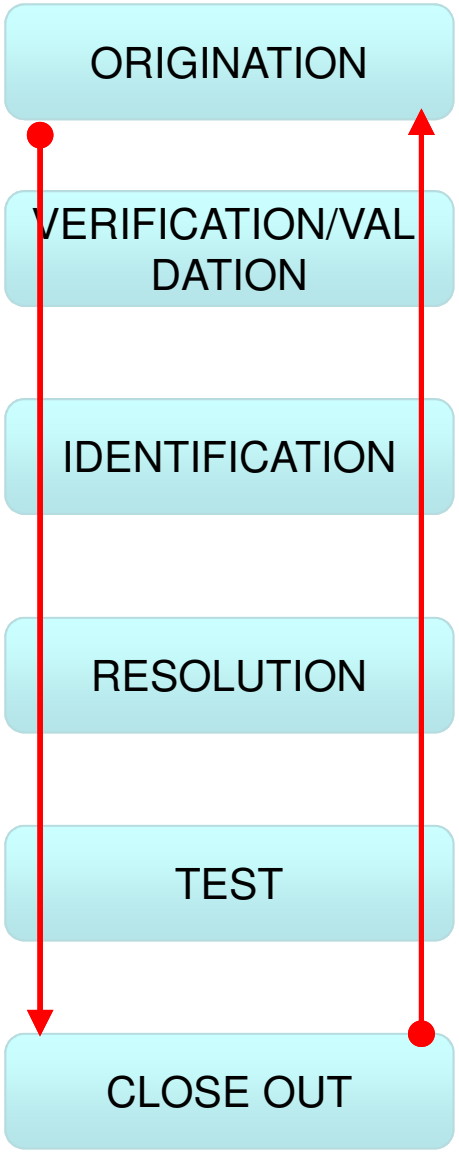
Within a spacecraft there are many hardware and software interactions.

Where do Software Problems Occur?



Problems can arise anywhere!

PFR Process



WHERE, WHEN, WHAT, HOW, WHY

IS THIS TRUE? IS IT REPEATABLE?

WHY DID THIS HAPPEN? RATE THE RISK TO THE MISSION!

WHAT IS THE FIX/CORRECTIVE ACTION?

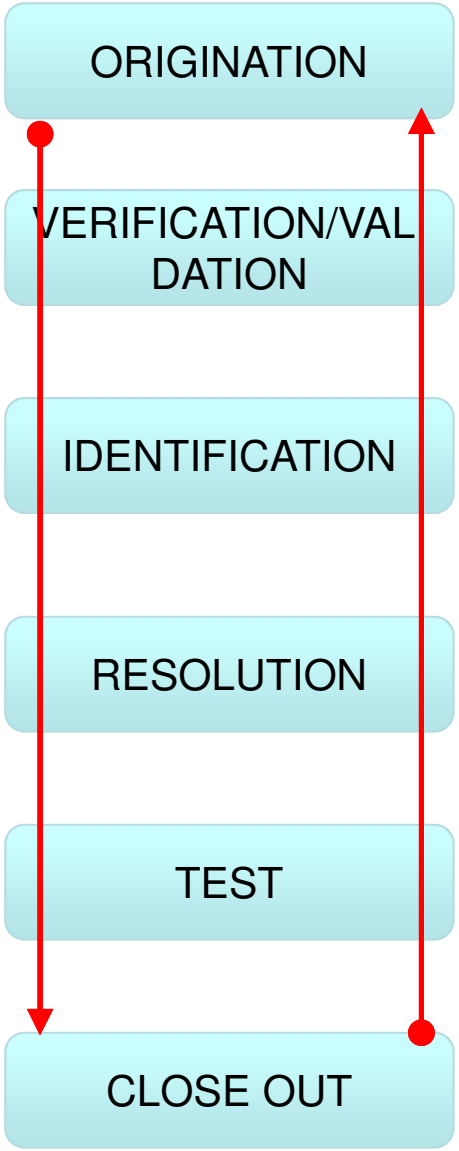
IS THE FIX COMPLETE? DID WE BREAK SOMETHING ELSE?

WE FIXED IT! TELL OTHERS! IS THERE RESIDUAL

OR

NOT TOTALLY FIXED! GO BACK OR USE-AS-IS!

PFR Process



FSW Version and Branch, GDS Version and Branch … Testbed Location, Time (In-Situ, Launch, Cruise, EDL)

MAP the PFR to Flight Software Logs – Flight Software Team has their own internal reporting system.

WHY DID THIS HAPPEN? – The FSW Developers usually understand this. START TO RATE THE RISK TO THE MISSION!

WHAT IS THE FIX/CORRECTIVE ACTION?
INITIAL PEER REVIEW OF THE FIX BEGINS.

IS THE FIX COMPLETE? DID WE BREAK SOMETHING ELSE?

WE FIXED IT! TELL OTHERS! IS THERE RESIDUAL RISK
OR
NOT TOTALLY FIXED! GO BACK OR USE-AS-IS!
WRITE A DETAILED SUMMARY.

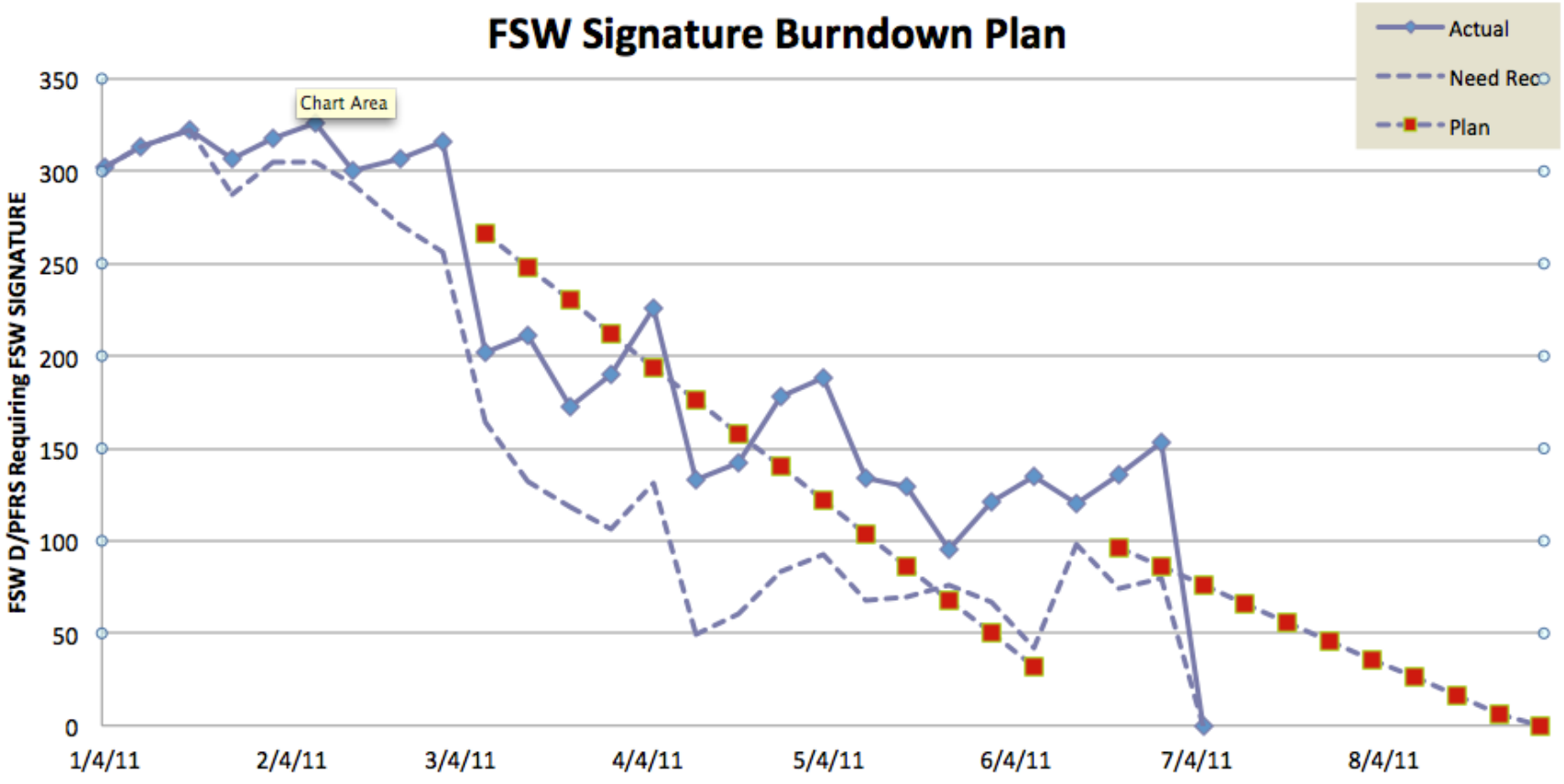
The Importance of PFRs

- **A PFR should be tied to its originator. Thus the originator is the owner (willingly or unwillingly).**
- **It is very important to understand where this issue originated – test, development, poor requirements, configuration, procedures.**
- **Various team members need to be included in the communication chain.**
- **When looking at a PFR it is important to quickly identify the RISK. It is also important to try to see if the issue is RELATED to other issues.**

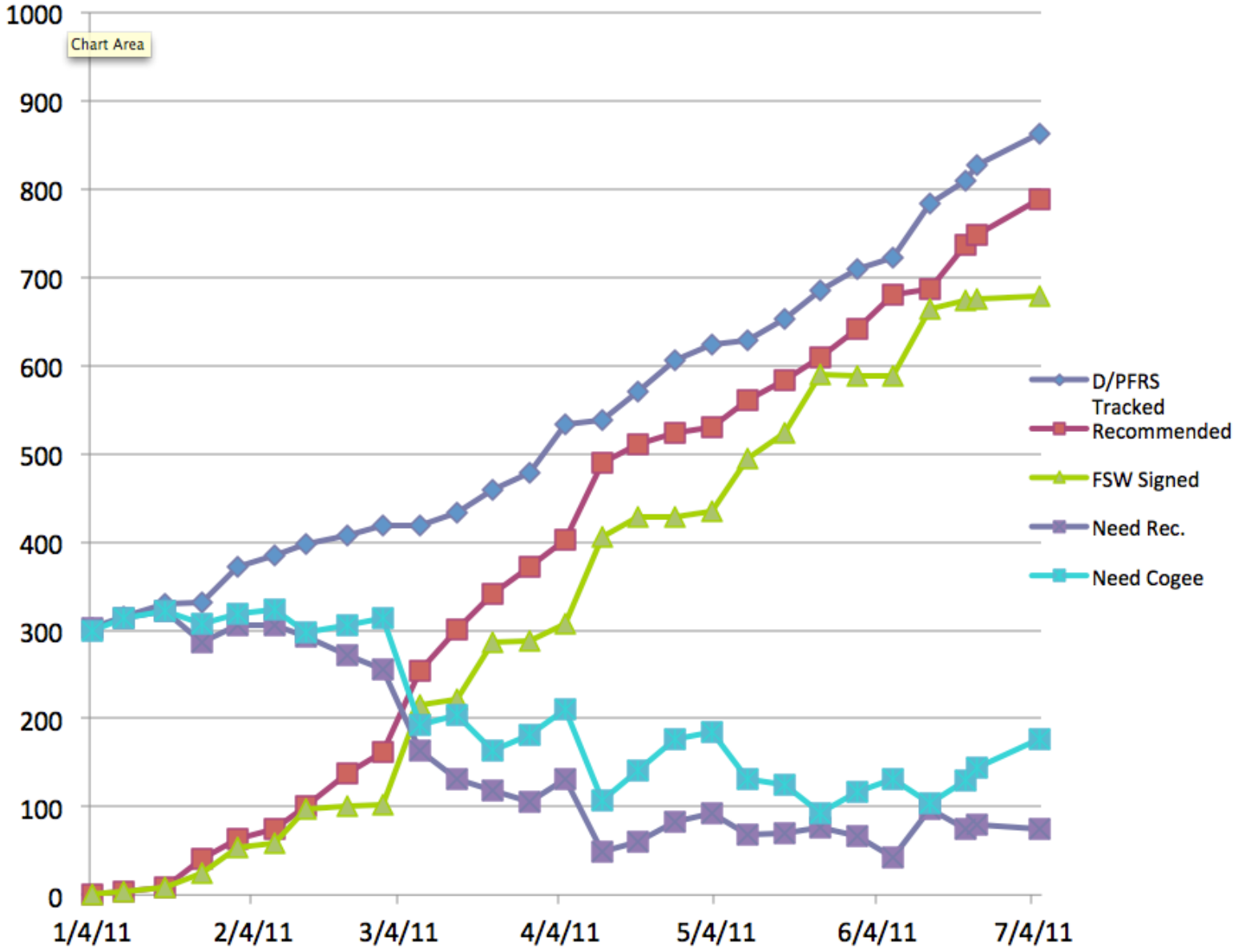
Mars Science Laboratory

- **For MSL we reviewed over 1700 FSW PFRs, 400+ GDS PFRs/ISAs, and 3400+ FSW Requirements/VIs, and also certified 69 FSW and 24 GDS SRCRs.**
- **Pre-launch we worked to close out over 300 open Flight Software issues.**
- **Post-launch we continued to work Cruise, EDL and Surface PFR issues.**
- **Some issues led to the creation of operational Flight Rules.**

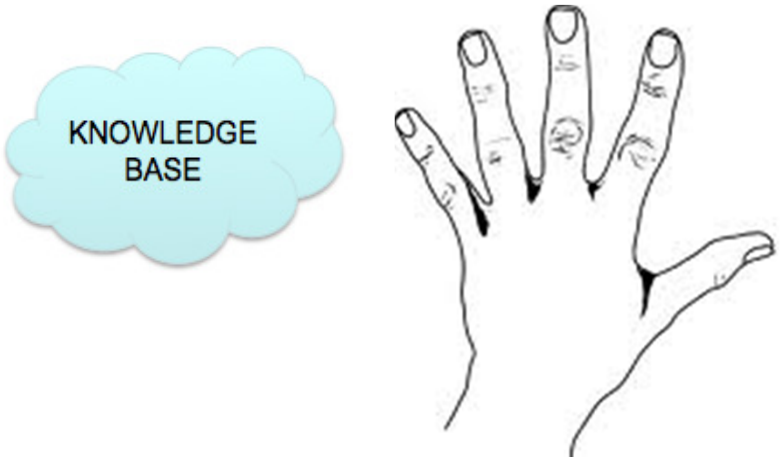
Mars Science Laboratory



Mars Science Laboratory



Left Hand – Right Hand Issues



TESTERS

SOFTWARE DEVELOPERS

SYSTEM ENGINEERS

SOFTWARE QUALITY PEOPLE

MISSION OPERATIONS

COMMAND SEQUENCING PEOPLE

INSTRUMENT TEAM

SCIENCE TEAM

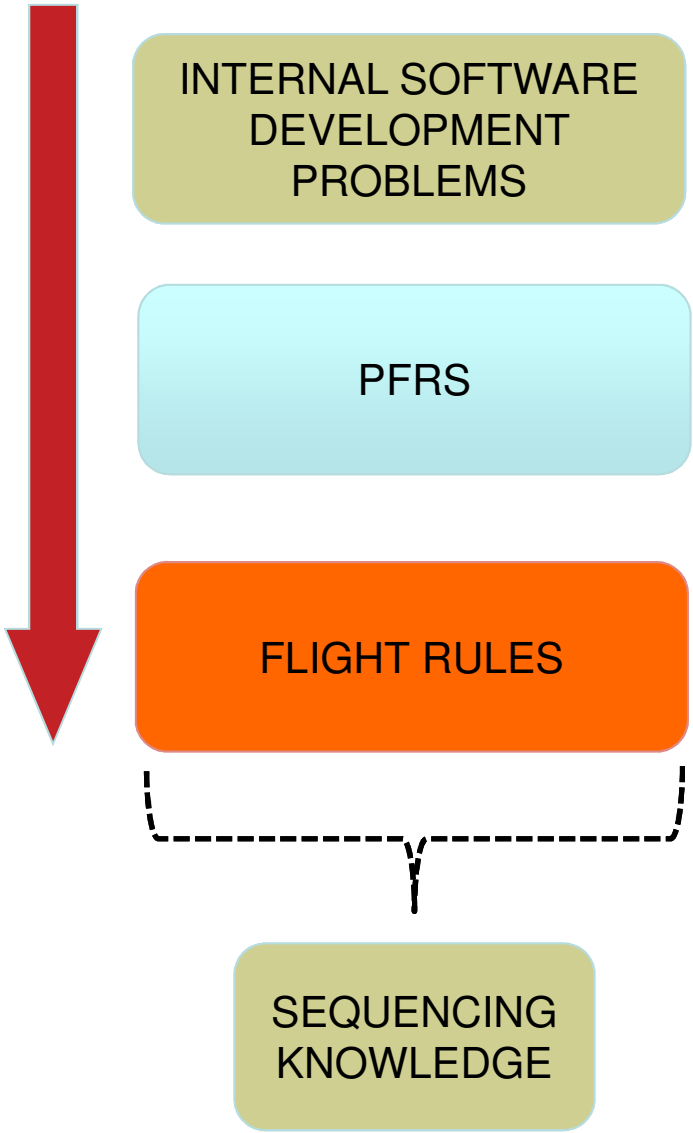
OTHER SYSTEM ENGINEERS

OTHER HARDWARE PEOPLE

MANAGERS

SOMEWHAT REMOVED MANAGERS …

Left Hand – Right Hand Issues - Example



WE DISCOVERED A BUG IN THE CODE AND OPEN A SOFTWARE TICKET!

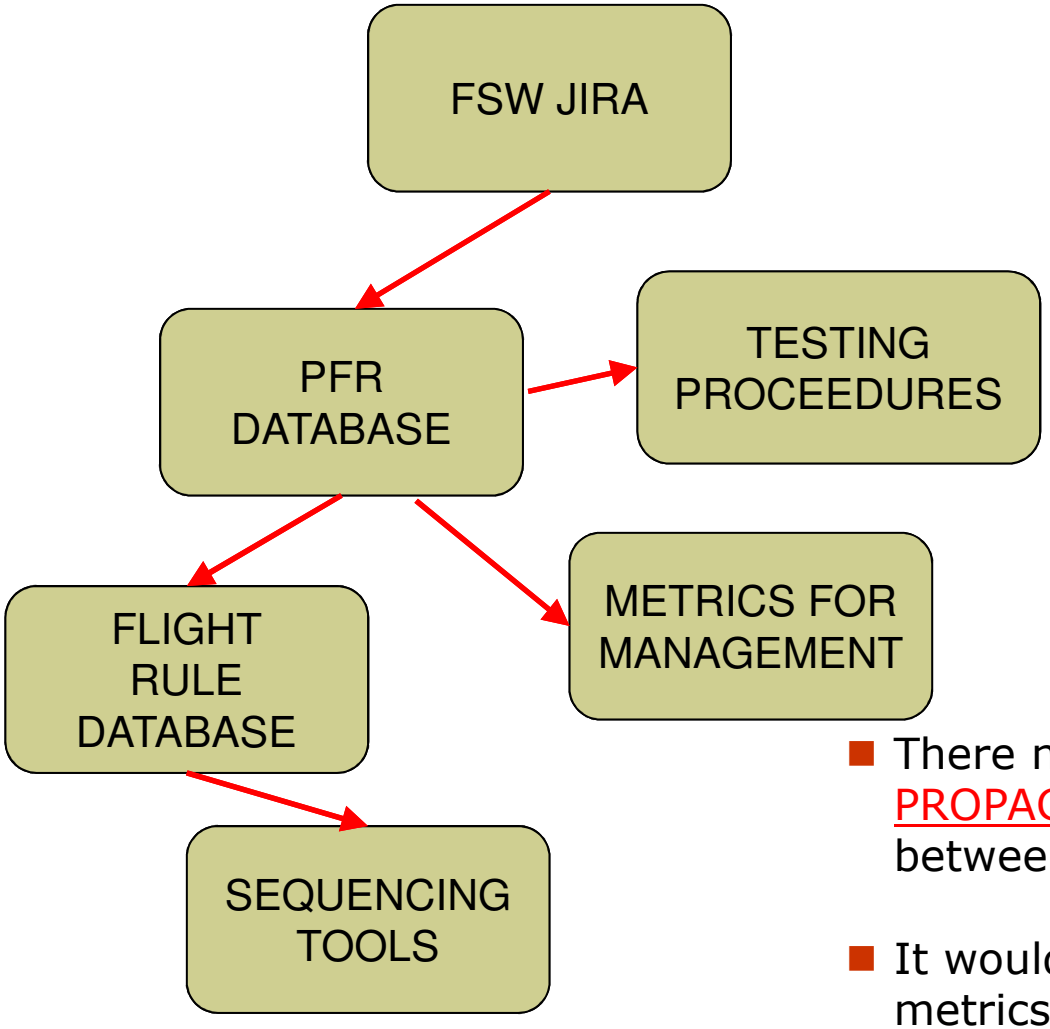
UH OH! THE BUG MANIFESTED ITSELF DURING A TEST OR DEVELOPMENT … BUT WE KNOW ABOUT IT.

WE ARE ACTUALLY NOT GOING TO FIX THIS PROBLEM IN THE CURRENT RELEASE! WE NEED TO COORDINATE WITH OPS TO DOCUMENT THIS – A FLIGHT RULE … OR THIS IS ACTUALLY NOT A PROBLEM AT ALL.

A FLIGHT RULE IS A GUIDE:
* KNOW THIS
* DON'T DO THIS … OR … DO THIS! IN THIS ORDER:
  ** *FOLLOW THIS PROCEDURE.*
* THE SEQUENCE ENGINE CHECKS FOR THEM.
* PFRS CAN LEAD TO THE CREATION OF FLIGHT RULES

A FLIGHT RULE CAN ALSO HIGHLIGHT: IDIOSYNCRATIC BEHAVIOR … ALSO CALLED "FLIGHT SOFTWARE IDIOS".

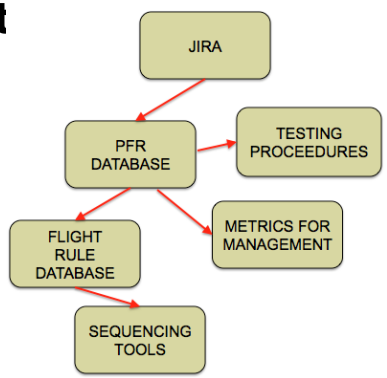
Left Hand – Right Hand Issues – The Intersection of Common Knowledge



- There needs to be a way to efficiently PROPAGATE Flight Software Issues between disparate databases and tools.
- It would be nice to have the tools and metrics automatically COUPLED together ... somehow.

Left Hand – Right Hand Issues

- **There needs to be a way to efficiently propagate Flight Software Issues between disparate databases and tools. For example if a JIRA ticket could automatically generate a PFR Report.**



- **It would be nice to have the tools and metrics automatically COUPLED together.**
- **A challenge is the use of different tools and that often share the same or similar information. There is an intersection of knowledge …**
- **Tools that allow automation/scripting are helpful.**
- **The current practice involves a lot of Microsoft Excel spreadsheets as the glue. This is time consuming.**

Flight Software Testing

- **With hundreds if not thousands of software issues, managing flight software is a challenge.**
- **Unit testing helps, testing at each stage helps (BIT, FIT, SIT, etc), regression testing helps. More testing is better at each step of the problem resolution.**
- **JPL uses a simulation environment that allows the Flight Software Environment to be tested.**
- **Real testbeds are a shared resource and testbed time must be scheduled … this is often a challenge.**
- **It would be nice to have higher fidelity simulation environment running compiled flight code in a virtualization environment. However current COTS products are expensive.**
- **Future ideas may be :**
  - **Look at tools like QEMU which has a PowerPC and ARM models.**
  - **Move some functionality away from RTOS - look at using Unix/Linux as the OS and push non-critical processes into user land and minimizing the impact of certain failures.**

THE END

- QUESTIONS?