Electrical & Computer ENGINEERING

18-642:
System Level Testing

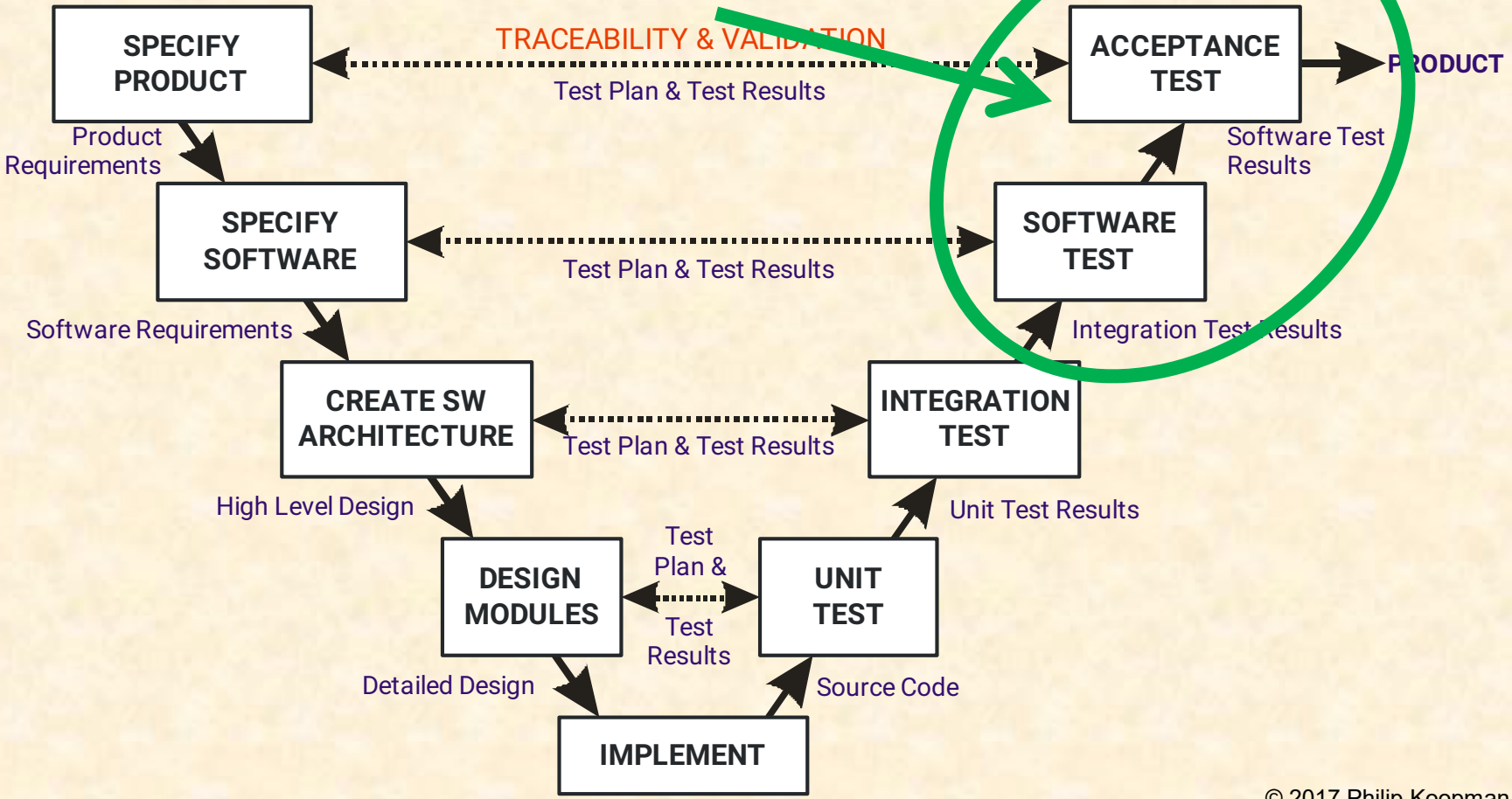
9/26/2019

"Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it."

– Brian W. Kernighan

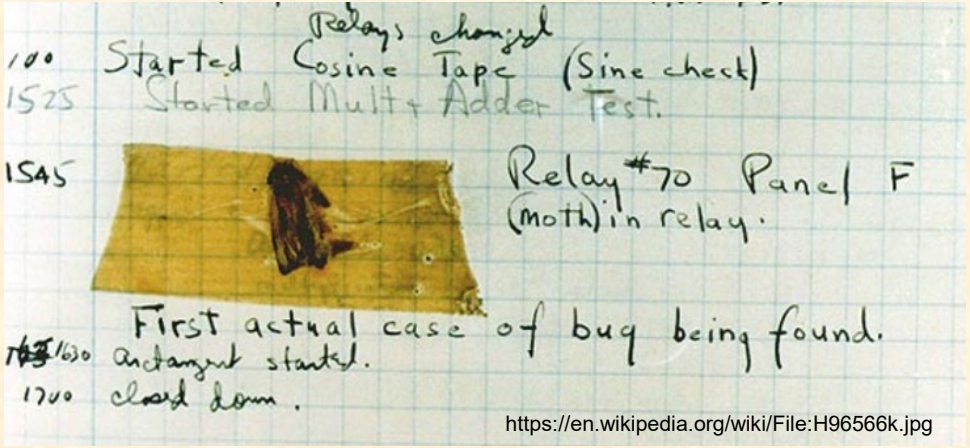
Carnegie Mellon University

YOU ARE HERE



System Level Testing

- **Anti-Patterns:**
  - **Excessive defect "escapes" to field testing**
  - **Majority of testing effort is ad hoc exploratory testing**
  - **Acceptance testing is the only testing done on system**



https://en.wikipedia.org/wiki/File:H96566k.jpg

The Famous "First Bug"
(But, Edison invented the term https://goo.gl/cVLpcX)

- **System test is last line of defense against shipping bugs**
  - System-level "acceptance test" emphasizes customer-type usage
  - Software test emphasizes aspects not visible to customer
    - E.g., is the watchdog timer turned on and working?

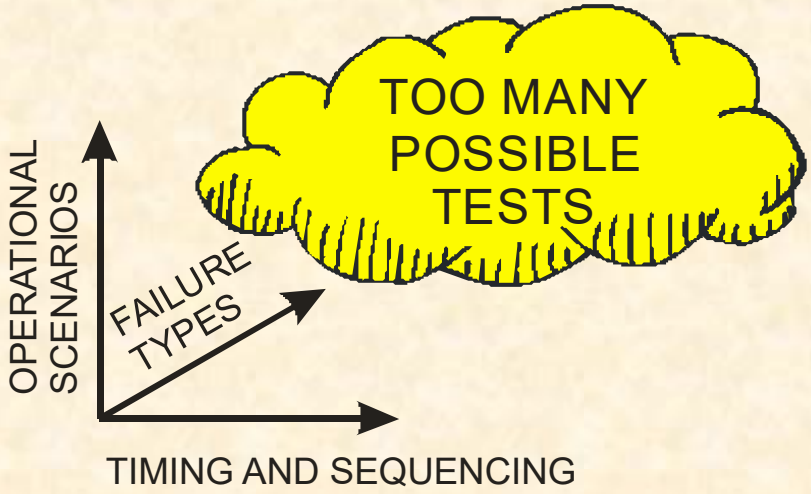
Effective System Testing

- **System test plan covers all requirements**
  - Every product requirement is tested
    - Ad hoc testing helps, but should not be primary method
  - Non-customer-visible requirements are tested
    - Especially non-functional requirements
  - Need to deal with embedded system I/O
    - Use a HAL and swap in a test simulator harness
- **Each bug found in system test is a huge deal**
  - You should find few (<5%?) bugs in system test
  - Bug found in system test is a process failure
    - System requirement defects should be most of what you find
    - Make sure “one-off” bugs aren’t just tip of the iceberg process problems

https://goo.gl/uWXQBC

Product Testing Won't Find All Bugs

- **Testing bad software simply makes it less bad**
  - Testing cannot produce good software all on its own



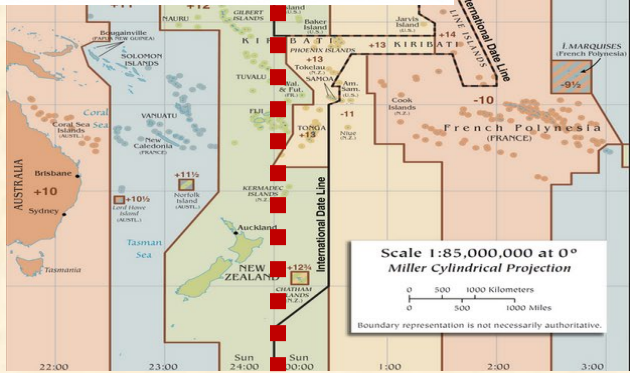
- **One third of faults take more than 5000 years to manifest**

Adams, N.E., "Optimizing preventive service of software product," IBM Journal of Research and Development, 28(1), p. 2-14, 1984. (Table 2, pg. 9, 60 kmonth column)

- Do you test for more than 5000 years of use?
- Your customers will regularly experience bugs that you will not see during testing

F-22 Raptor Date Line Incident

- **February 2007; six aircraft fly to Japan**
  - $360 million per aircraft
  - Computer crash crossing the International Date Line:
    - No navigation, communications, fuel management, …
    - Escorted to Hawaii by tankers
    - Could have lost all six if poor visibility
- **Cause:**
  - "It was a computer glitch in the millions of lines of code, somebody made an error in a couple lines of the code and everything goes." [https://goo.gl/edGdAL]
- **Related: F-16 inverts when crossing equator**
  - Found in simulation. (Perhaps an urban legend?)
  - But still should put designers on notice of such bugs



http://catless.ncl.ac.uk/Risks/3.44.html#subj1.1

Bug Farms: Concentrations of Buggy Code

- **90/10 rule applied to bug farms:**
  - 90% of the bugs are in 10% of the modules
  - Those are the most complex modules
- **Bug farms can be more than just bad code**
  - Bad design that makes it tough to write code
  - Too complex to understand and test
  - Poorly defined, confusing interfaces
- **Fixing bug farms:**
  - Refactor the module, redesign the interface
  - Often, smart to throw away and redesign that piece

Top 10 Risks of Poor Embedded Software Quality

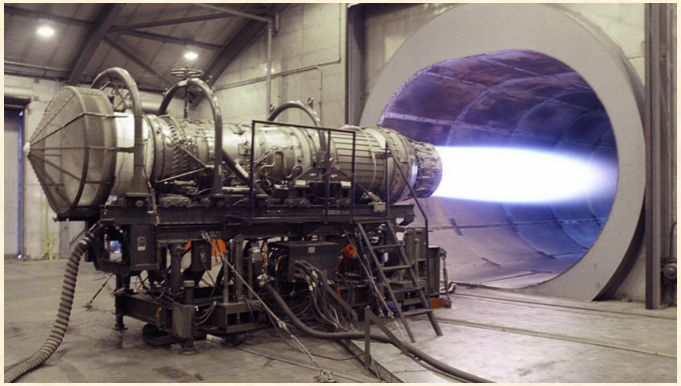
10. Your module fails unit test (Tie with #9)
9. A bug is found in peer review (Tie with #10)
8. The system fails integration or software testing
7. The system fails acceptance testing
6. You get a field problem report
5. Your boss wakes you up at 2 AM because a Big Customer is off-line
4. You get an airplane ticket to a war zone to install a software update
3. You hear about the bug on social media
2. Your corporate lawyers ask you to testify in the lawsuits

And, the Number One Worst Way To Find A Bug:

1. The reporters camped outside your house ask you to comment on it

System Test Best Practices

- **Test all system requirements**
  - Everything it's supposed to do
  - Fault management responses
  - Performance, extra-functional reqts.

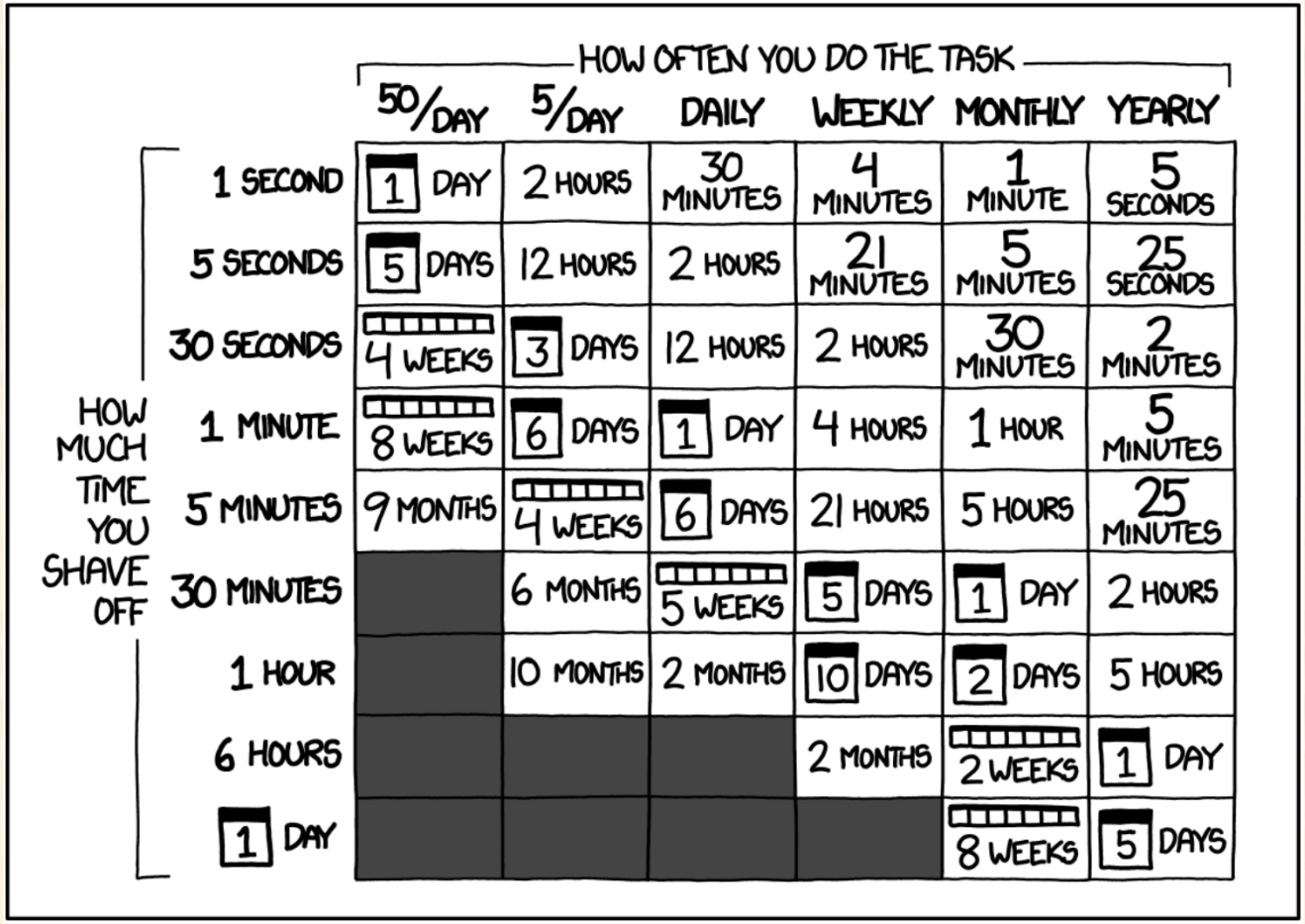


https://goo.gl/2iSZaM

- **Acceptance test vs. software test**
  - Acceptance test is from customer point of view – domain testing
  - SW test uses internal test interfaces – software testing skills
- **System test pitfalls:**
  - Impractical to get high coverage; won't find all bugs
  - Testing fault management is hard if you haven't planned for it

HOW LONG CAN YOU WORK ON MAKING A ROUTINE TASK MORE EFFICIENT BEFORE YOU'RE SPENDING MORE TIME THAN YOU SAVE?

(ACROSS FIVE YEARS)



https://xkcd.com/1205/