Avoiding the “Test and Test Again Syndrome”

CAST July 10, 2007

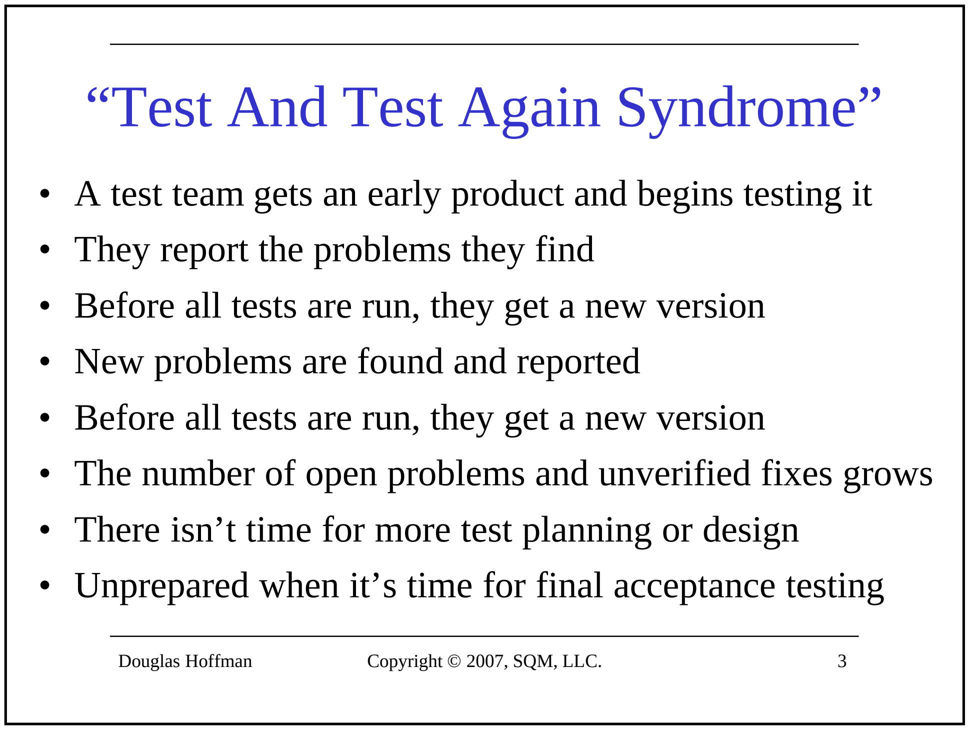
Douglas Hoffman
QA Program Manager
Hewlett-Packard
408-285-2408 (W)
408-741-4830 (H)
douglas.hoffman@HP.com

Disclaimer

The experience and opinions described in this paper are based on work completed prior to Douglas’ joining Hewlett-Packard and do not represent Hewlett-Packard’s organization, processes, or opinions.

The original work was undertaken under the auspices of Software Quality Methods, LLC.

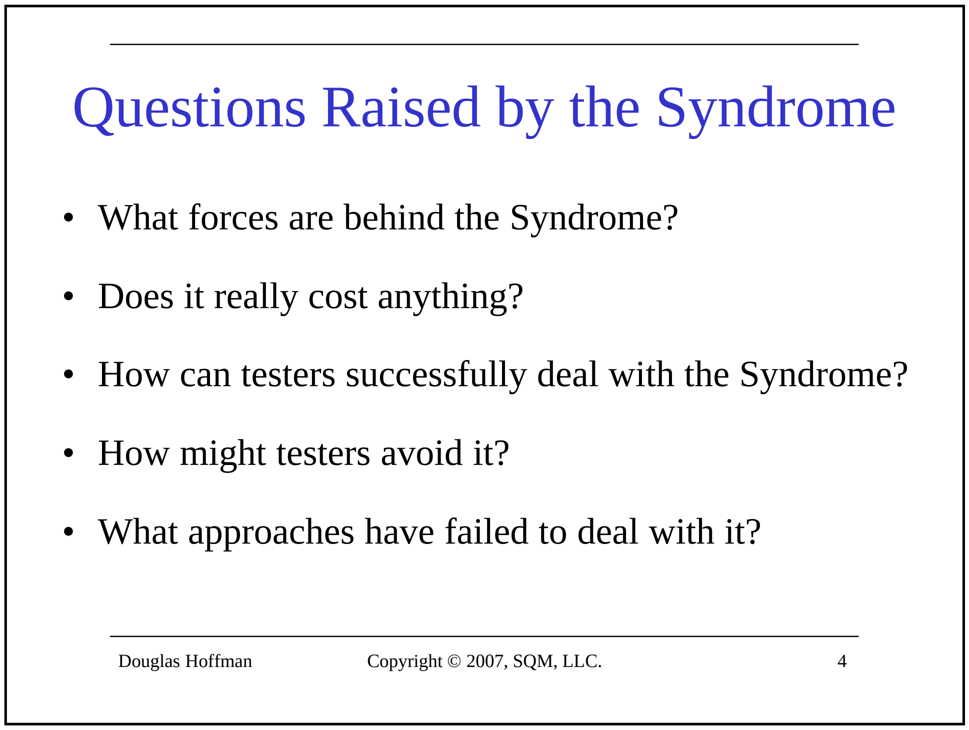
www.SoftwareQualityMethods.com



"Test And Test Again Syndrome"

- A test team gets an early product and begins testing it
- They report the problems they find
- Before all tests are run, they get a new version
- New problems are found and reported
- Before all tests are run, they get a new version
- The number of open problems and unverified fixes grows
- There isn't time for more test planning or design
- Unprepared when it's time for final acceptance testing

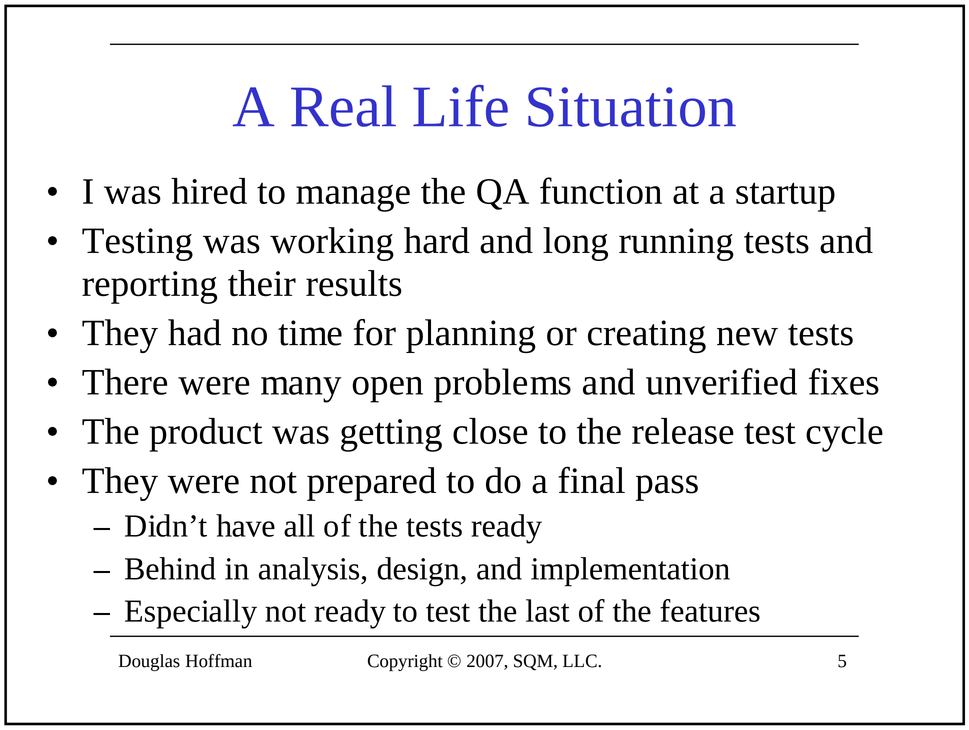
Douglas Hoffman Copyright © 2007, SQM, LLC. 3



Questions Raised by the Syndrome

- What forces are behind the Syndrome?
- Does it really cost anything?
- How can testers successfully deal with the Syndrome?
- How might testers avoid it?
- What approaches have failed to deal with it?

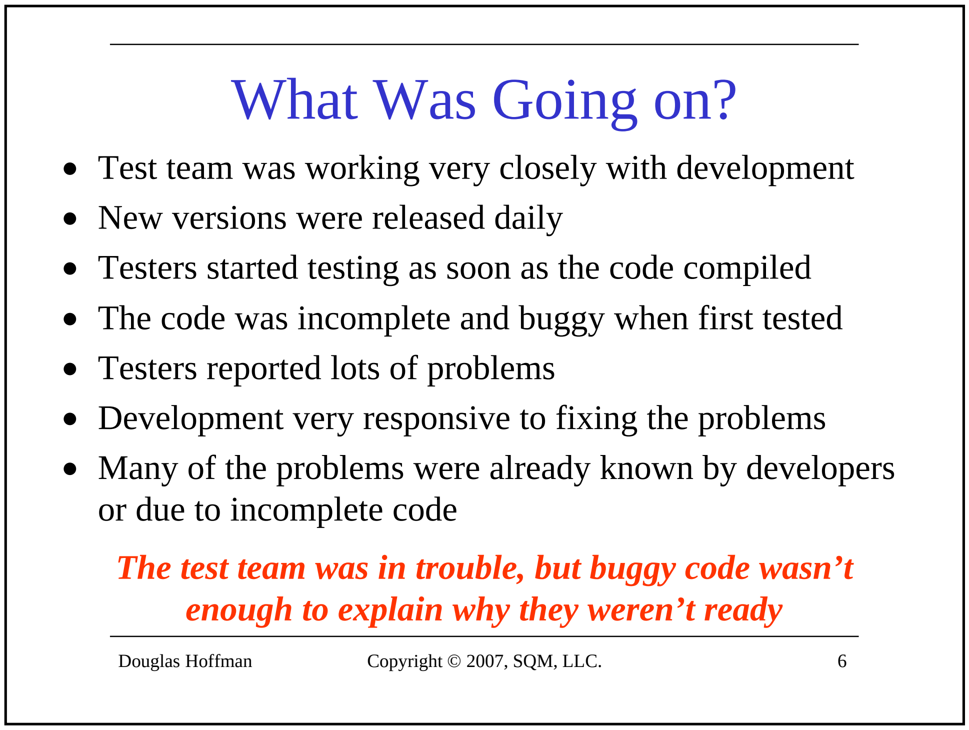
Douglas Hoffman Copyright © 2007, SQM, LLC. 4



A Real Life Situation

- I was hired to manage the QA function at a startup
- Testing was working hard and long running tests and reporting their results
- They had no time for planning or creating new tests
- There were many open problems and unverified fixes
- The product was getting close to the release test cycle
- They were not prepared to do a final pass
  - Didn’t have all of the tests ready
  - Behind in analysis, design, and implementation
  - Especially not ready to test the last of the features

Douglas Hoffman Copyright © 2007, SQM, LLC. 5

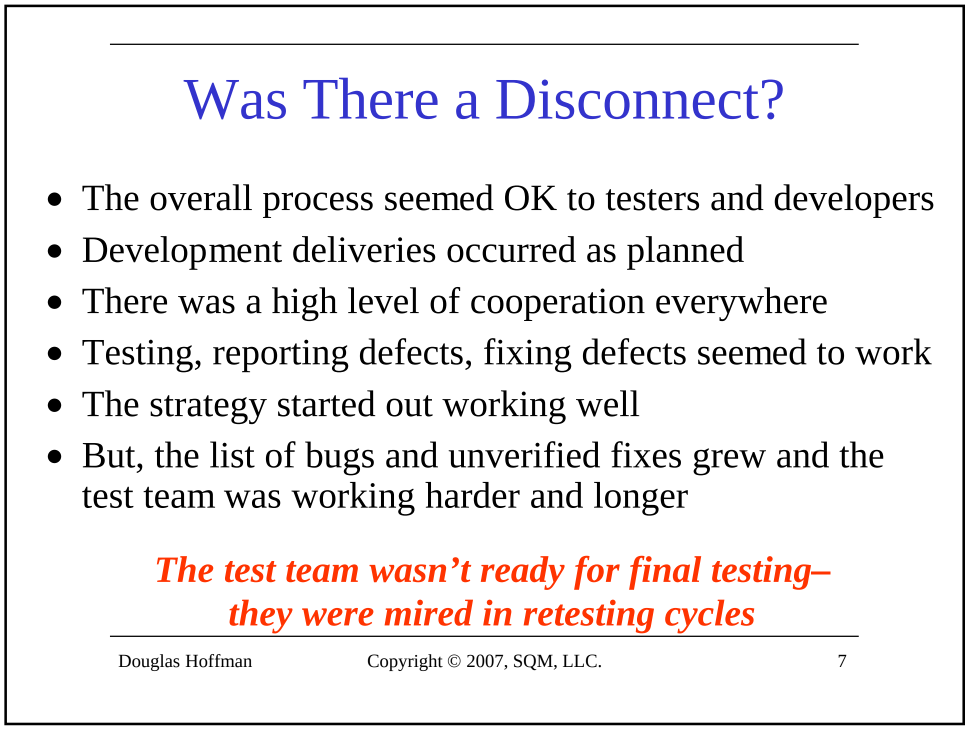


What Was Going on?

- Test team was working very closely with development
- New versions were released daily
- Testers started testing as soon as the code compiled
- The code was incomplete and buggy when first tested
- Testers reported lots of problems
- Development very responsive to fixing the problems
- Many of the problems were already known by developers or due to incomplete code

The test team was in trouble, but buggy code wasn’t enough to explain why they weren’t ready

Douglas Hoffman Copyright © 2007, SQM, LLC. 6

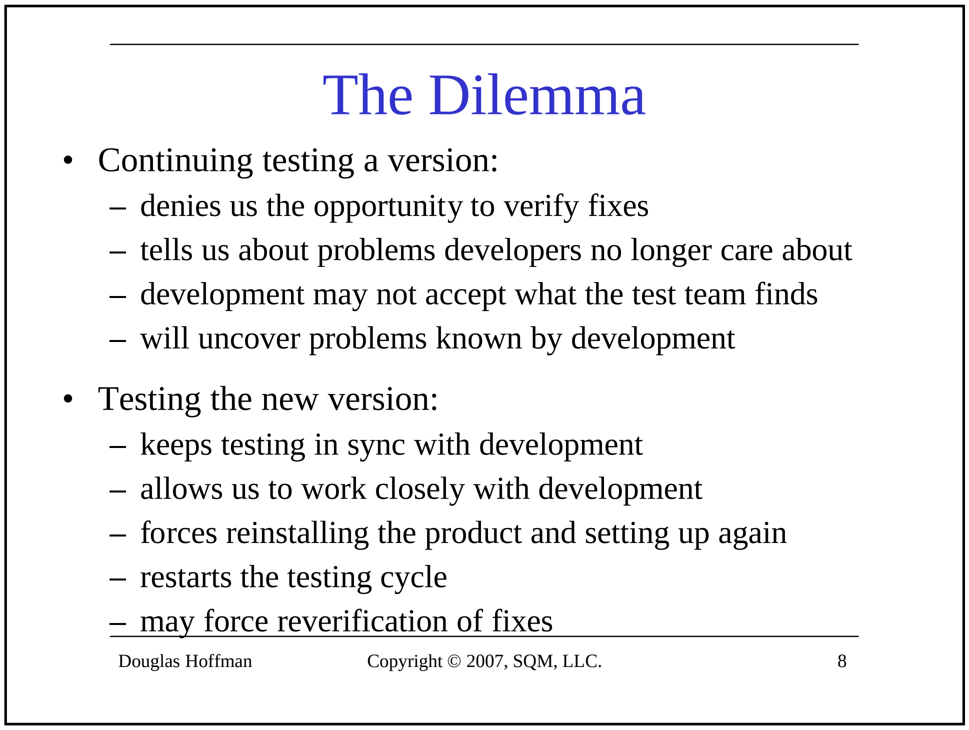


Was There a Disconnect?

- The overall process seemed OK to testers and developers
- Development deliveries occurred as planned
- There was a high level of cooperation everywhere
- Testing, reporting defects, fixing defects seemed to work
- The strategy started out working well
- But, the list of bugs and unverified fixes grew and the test team was working harder and longer

The test team wasn't ready for final testing– they were mired in retesting cycles

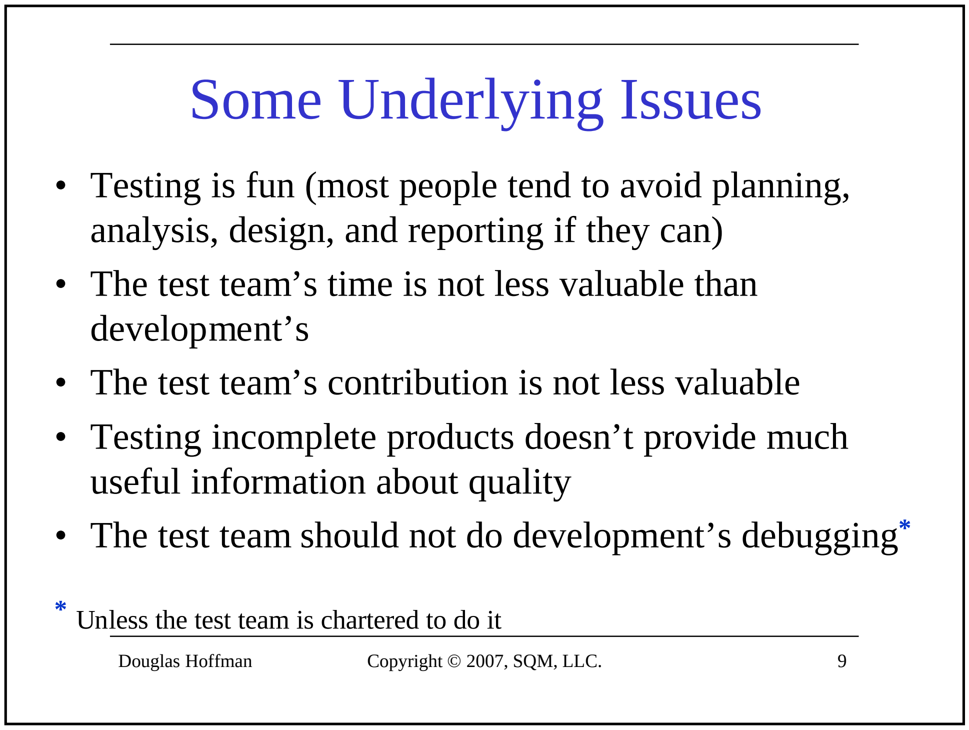
Douglas Hoffman Copyright © 2007, SQM, LLC. 7



The Dilemma

- Continuing testing a version:
  - denies us the opportunity to verify fixes
  - tells us about problems developers no longer care about
  - development may not accept what the test team finds
  - will uncover problems known by development
- Testing the new version:
  - keeps testing in sync with development
  - allows us to work closely with development
  - forces reinstalling the product and setting up again
  - restarts the testing cycle
  - may force reverification of fixes

Douglas Hoffman Copyright © 2007, SQM, LLC. 8

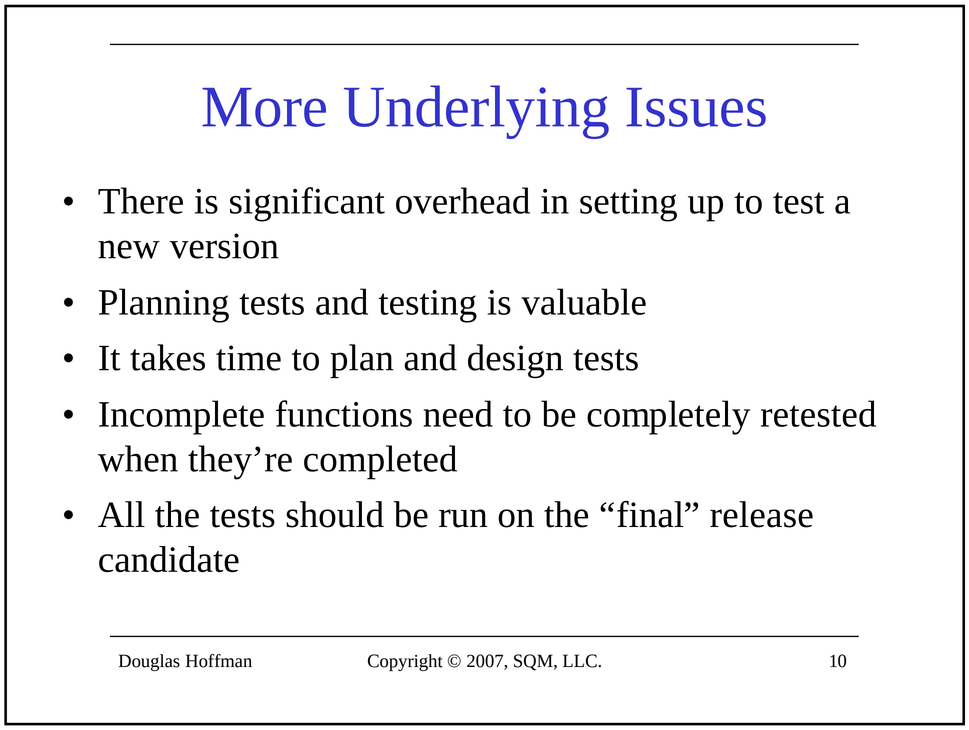


Some Underlying Issues

- Testing is fun (most people tend to avoid planning, analysis, design, and reporting if they can)
- The test team's time is not less valuable than development's
- The test team's contribution is not less valuable
- Testing incomplete products doesn't provide much useful information about quality
- The test team should not do development's debugging*

* Unless the test team is chartered to do it

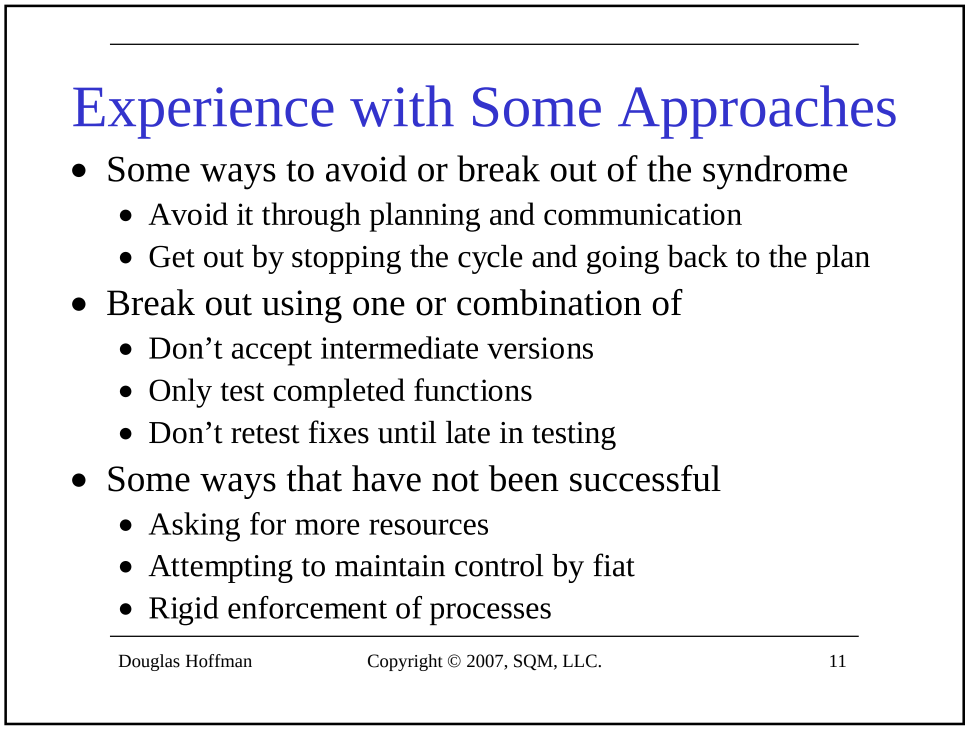
Douglas Hoffman Copyright © 2007, SQM, LLC. 9



More Underlying Issues

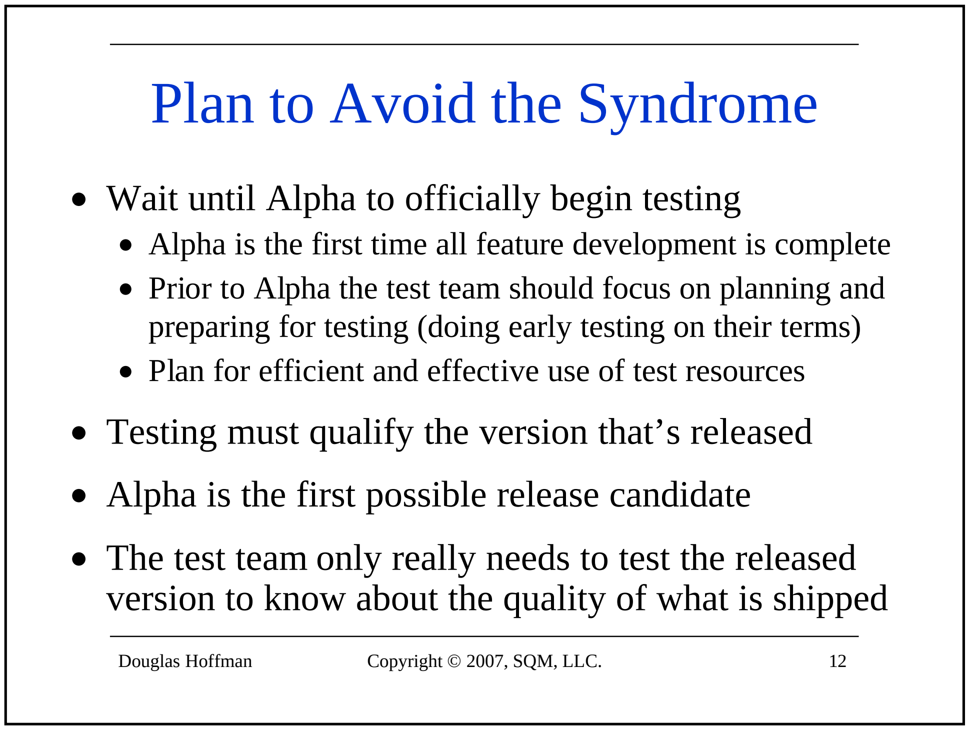
- There is significant overhead in setting up to test a new version
- Planning tests and testing is valuable
- It takes time to plan and design tests
- Incomplete functions need to be completely retested when they're completed
- All the tests should be run on the "final" release candidate

Douglas Hoffman Copyright © 2007, SQM, LLC. 10



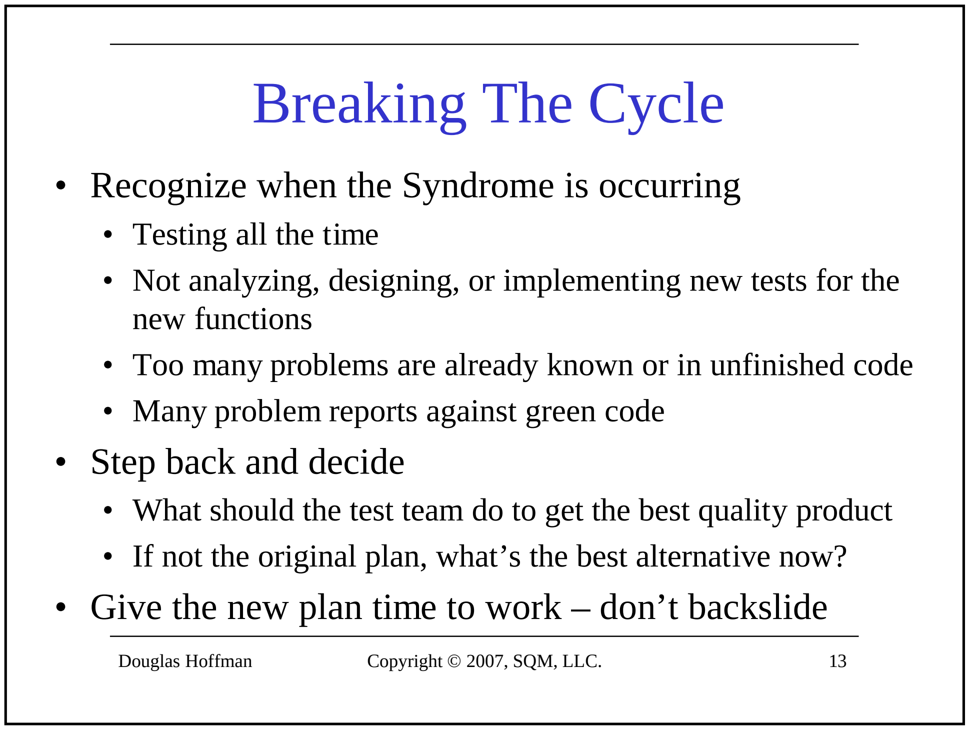
Experience with Some Approaches

- Some ways to avoid or break out of the syndrome
  - Avoid it through planning and communication
  - Get out by stopping the cycle and going back to the plan
- Break out using one or combination of
  - Don’t accept intermediate versions
  - Only test completed functions
  - Don’t retest fixes until late in testing
- Some ways that have not been successful
  - Asking for more resources
  - Attempting to maintain control by fiat
  - Rigid enforcement of processes



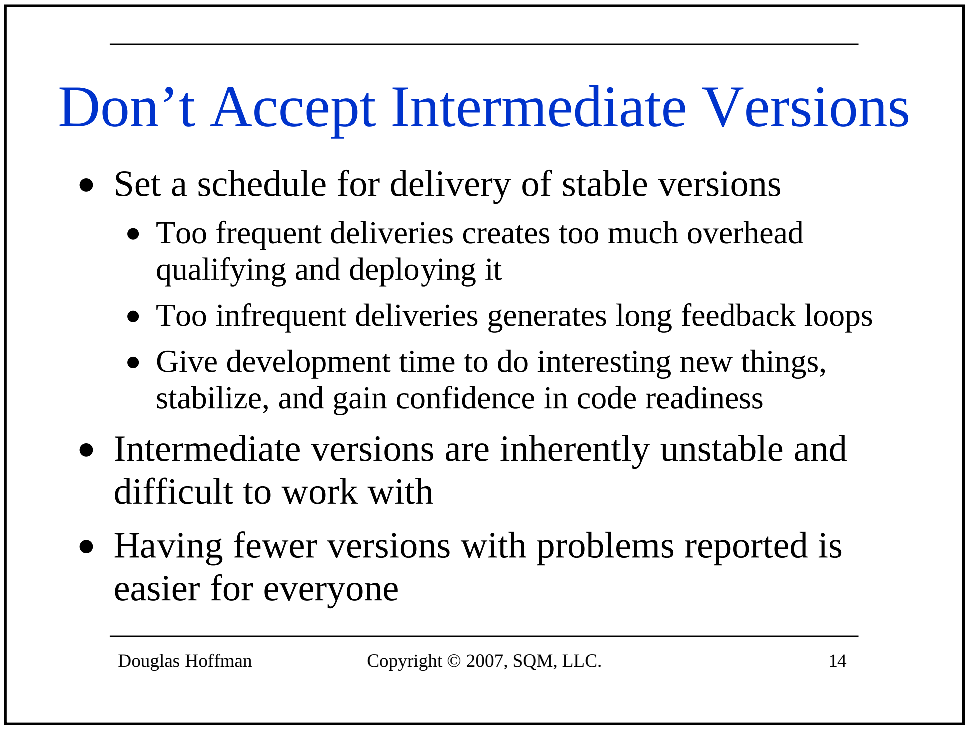
Plan to Avoid the Syndrome

- Wait until Alpha to officially begin testing
  - Alpha is the first time all feature development is complete
  - Prior to Alpha the test team should focus on planning and preparing for testing (doing early testing on their terms)
  - Plan for efficient and effective use of test resources
- Testing must qualify the version that’s released
- Alpha is the first possible release candidate
- The test team only really needs to test the released version to know about the quality of what is shipped



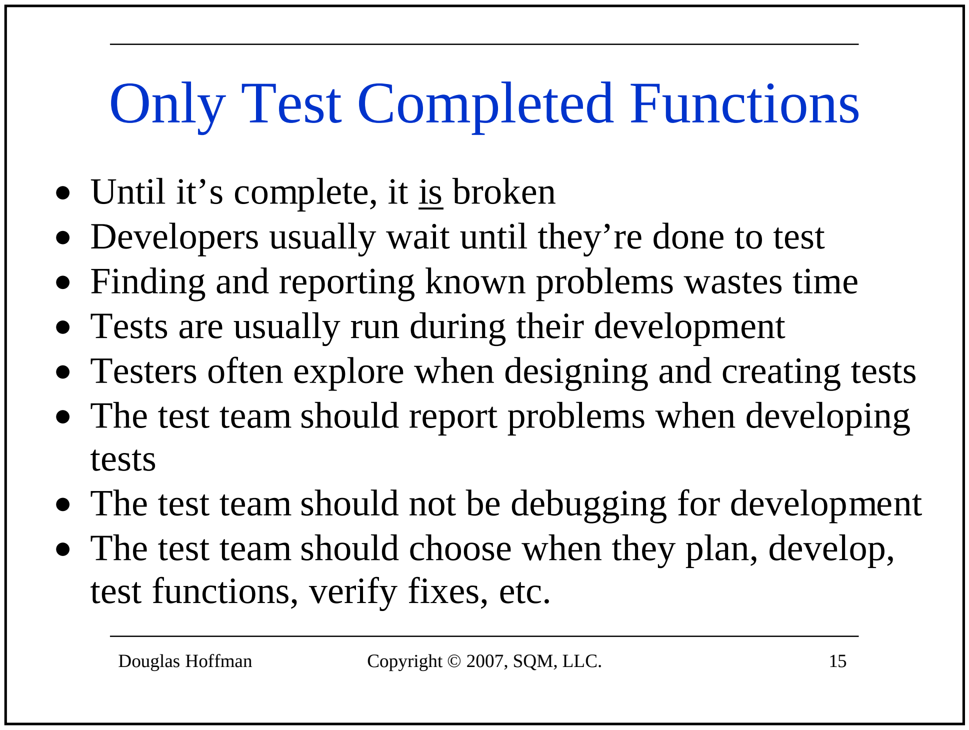
Breaking The Cycle

- Recognize when the Syndrome is occurring
  - Testing all the time
  - Not analyzing, designing, or implementing new tests for the new functions
  - Too many problems are already known or in unfinished code
  - Many problem reports against green code
- Step back and decide
  - What should the test team do to get the best quality product
  - If not the original plan, what's the best alternative now?
- Give the new plan time to work – don't backslide



Don't Accept Intermediate Versions

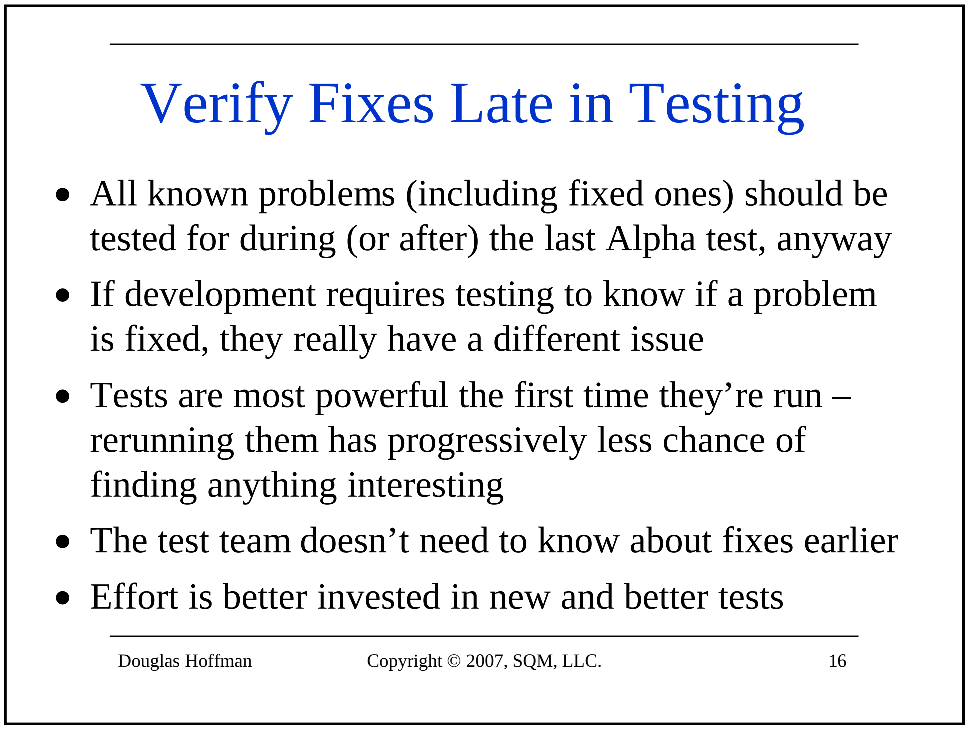
- Set a schedule for delivery of stable versions
  - Too frequent deliveries creates too much overhead qualifying and deploying it
  - Too infrequent deliveries generates long feedback loops
  - Give development time to do interesting new things, stabilize, and gain confidence in code readiness
- Intermediate versions are inherently unstable and difficult to work with
- Having fewer versions with problems reported is easier for everyone



Only Test Completed Functions

- Until it’s complete, it is broken
- Developers usually wait until they’re done to test
- Finding and reporting known problems wastes time
- Tests are usually run during their development
- Testers often explore when designing and creating tests
- The test team should report problems when developing tests
- The test team should not be debugging for development
- The test team should choose when they plan, develop, test functions, verify fixes, etc.

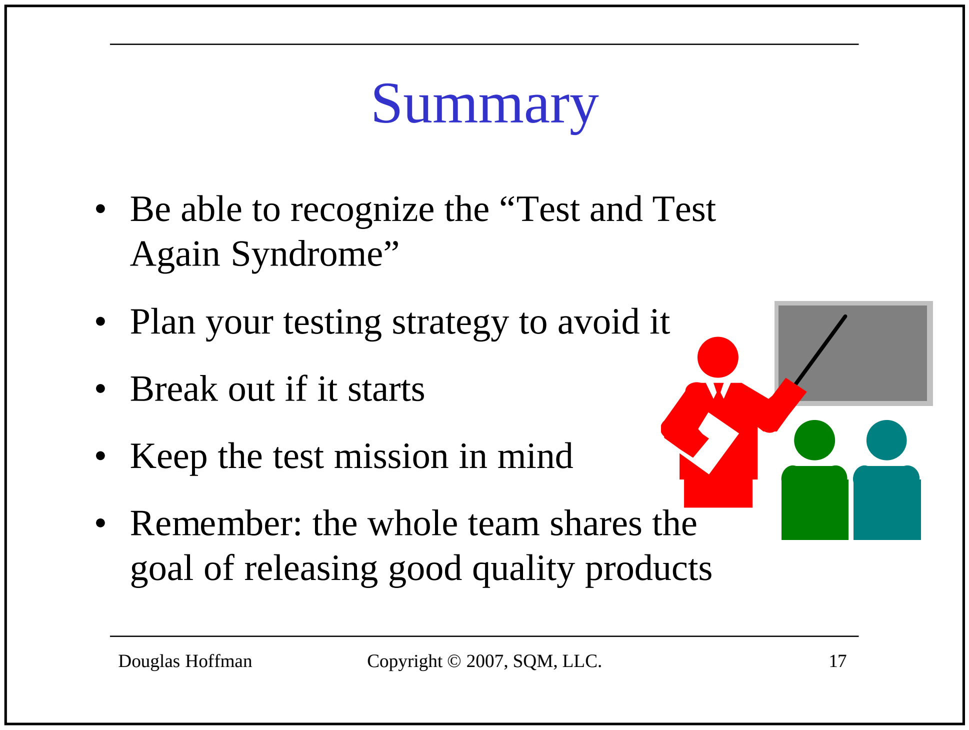
Douglas Hoffman Copyright © 2007, SQM, LLC. 15



Verify Fixes Late in Testing

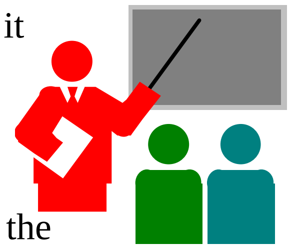
- All known problems (including fixed ones) should be tested for during (or after) the last Alpha test, anyway
- If development requires testing to know if a problem is fixed, they really have a different issue
- Tests are most powerful the first time they’re run – rerunning them has progressively less chance of finding anything interesting
- The test team doesn’t need to know about fixes earlier
- Effort is better invested in new and better tests

Douglas Hoffman Copyright © 2007, SQM, LLC. 16



Summary

- Be able to recognize the “Test and Test Again Syndrome”
- Plan your testing strategy to avoid it
- Break out if it starts
- Keep the test mission in mind
- Remember: the whole team shares the goal of releasing good quality products



Douglas Hoffman Copyright © 2007, SQM, LLC. 17