

---

# Early Testing Without the “Test and Test Again” Syndrome

Better Software '04

Douglas Hoffman  
Software Quality Methods, LLC.  
24646 Heather Heights Place  
Saratoga, California 95070-9710  
Phone 408-741-4830  
Fax 408-867-4550  
doug.hoffman@acm.org  
[www.SoftwareQualityMethods.com](http://www.SoftwareQualityMethods.com)

Copyright © 2004, Software Quality Methods, LLC. No part of these graphic overhead slides may be reproduced, or used in any form by any electronic or mechanical duplication, or stored in a computer system, without written permission of the author.

---

## “Test And Test Again” Syndrome

- The test team gets a build begins testing it
- They report the bugs they find
- Before all tests are run, they get a new build
- New bugs are found and reported
- Before all tests are run, they get a new build
- The number of open bugs and unverified fixes grows
- There isn't time for more test planning or design
- They are not prepared for final acceptance testing

---

## A Situation

- I was brought into a startup to manage the QA function
- The test team was working hard and long hours testing and reporting their results – all their time was spent doing testing (none on planning or creating tests)
- There were lots of open bugs and unverified fixes
- We were getting close to the final test cycle for release
- We were not ready to do final acceptance testing
  - Didn't have all of the tests ready
  - Behind in analysis, design, and implementation
  - Especially not prepared for the last of the features

---

## What Was Going on?

- Test team was working very closely with development
- Testers started testing as soon as the code compiled
- The code was really buggy
- Testers reported lots of problems
- Many of the problems were known by development or due to incomplete code
- New builds were released daily
- Development very responsive to fixing the bugs

*We were in trouble, but buggy code wasn't enough to explain why we weren't ready*

---

## The Original Plan

- Test development strategy:
  - Analyze, design, and develop tests in the order of feature delivery
  - Do some testing when each feature is first delivered
  - Move on to the next features
  - Repeat the cycle throughout product implementation
- Rerun all the tests when the product is ready for release
  - Look for bugs
  - Verify fixes

---

## How Work Progressed

- Test Plan was done before first code was available
- Test analysis and design done for the first functions
- Received and tested the first product functions
- Reported the bugs on the first functions
- Took delivery of the next functions with some fixes for the first functions
- Regression tested the first functions, closed some bugs, opened other new ones
- Received frequent bug fix versions
- Tested second functions, reported bugs
- Took delivery of the next functions
- Etc.

---

## Is There a Disconnect?

- Overall test process sounds OK
- Development deliveries occurred as planned
- High level of cooperation everywhere
- Testing, reporting bugs, fixing bugs working
- The strategy started out working well
- The list of bugs and unverified fixes grew

*But, in the end we weren't ready – we were mired in testing and retesting cycles*

---

## Expected Strategy Execution

- Analyze first features and develop tests
- Do test analysis and design for the first functions
- Receive and test the first product functions
- Report the bugs on the first functions
- Take delivery of the next functions with some fixes
- Regression test the first functions, close some bugs, open others
- Test second functions, report bugs
- Receive new version with bug fixes
- Regression test, close some bugs, open others

---

## The Beginnings of Trouble

- Daily builds were delivered for testing containing fixes and partially completed new functions
- The testers had to choose between
  - Finishing testing the current build (ignoring the new one)
  - Installing and restarting testing on the new build

*We were headed for trouble either way*

---

## The Dilemma

- Continuing testing:
  - gives us the opportunity to verify fixes
  - tells us about bugs developers no longer care about
  - development may not accept what we find
  - will uncover bugs known by development
- Testing the new build:
  - keeps testing in sync with development
  - forces reinstalling the product and setting up again
  - restarts the testing cycle (we are repeating ourselves)
  - may force reverification of fixes

---

## Some Underlying Issues

- Testing is fun (most people tend to avoid planning, analysis, design, and reporting if they can)
- The test group's time is not less valuable than development's
- The test group's contribution is not less valuable
- The test group should not do development's debugging\*

\* Unless we are chartered to do it

---

## More Underlying Issues

- There is significant overhead in setting up to test a new build
- Planning tests and testing is valuable
- It takes time to plan and design tests
- Testing incomplete code doesn't provide much useful information about quality
- We will have to completely retest incomplete functions when they're completed
- We should run all our tests on the "final" release candidate

---

## How I've Been Successful

- Several ways to avoid or get out of the syndrome
  - Avoid it through planning and communication
  - Get out by stopping the cycle and going back to the plan
- One of (or combination of parts of):
  - Don't accept intermediate builds
  - Only test completed functions
  - Don't retest fixes until Alpha
  - Wait until Alpha to officially begin testing
- Not by
  - Asking for more resources
  - Holding development at arm's length

---

## Don't Accept Intermediate Builds

- Have a schedule for delivery of stable builds
  - 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 builds are inherently unstable
- The plan is for testing of functions, not fixes (we should choose when we verify fixes)
- It is much easier on everyone if we limit the number of different builds bugs are reported against

---

## Only Test Completed Functions

- Testing should not be debugging for development
- Until it's complete, we know a function will break
- Developers usually test when they think they're done
- Finding and reporting known bugs wastes everyone's time
- Test development usually requires running the test
- We test when designing and creating tests
- We report all bugs we find while developing tests

---

## Wait Until Alpha to Officially Begin Testing

- Alpha is the first time all feature development is complete
- Prior to Alpha we should focus on planning and preparing for testing (do early testing on our terms)
- We must qualify the version we release
- Alpha is the first possible release candidate
- We only really need to test the version we release

---

## Don't Retest Fixes Until Alpha

- Tests are most powerful the first time they're run
- We should invest instead in new and better tests
- We will need to test for all known bugs (including fixed ones) in (or after) Alpha anyway
- The test group doesn't need to know earlier
- If development must know if a bug is fixed, they really are asking a different question

---

## How To Break Out of The Cycle

- Recognize we're in it
  - Testing all the time
  - Not analyzing, designing, or implementing new tests on new functions
  - Too many bug reports against green code
  - Too many problems are known or in unfinished code
- Step back and decide
  - What should we be doing to get the best quality product
  - If not the original plan, what's the best alternative?
- Stick by your decision

---

## Summary

- Understand the test and test again cycle
- Plan your testing strategy
- Avoid it if possible – plan around it
- Break out if it starts
- Keep your eye on the goal
- Remember that the whole team shares the goal of releasing good quality code

