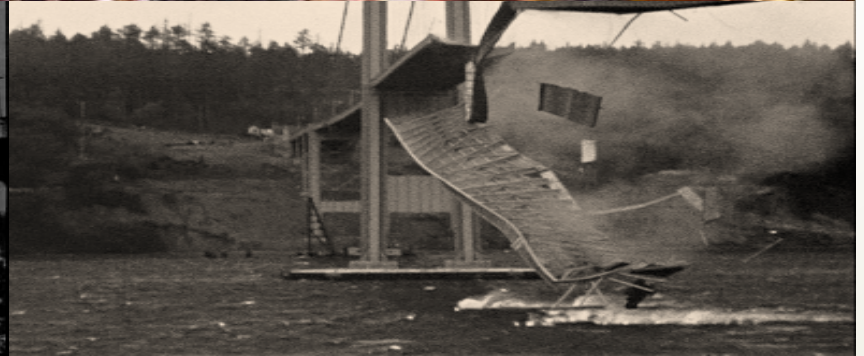
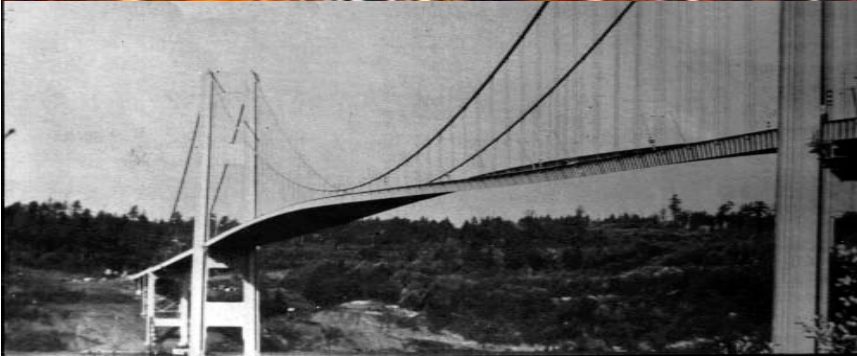


Six Hard-Won Lessons *Build and Deliver Software Quality*



RBCS
TIME TESTED.
TESTING IMPROVED.
www.RBCS-US.com



Why Does Software Quality Matter?

- ❖ Software quality problems can...
 - ❖ Damage brand image and loyalty (ask Toyota)
 - ❖ Result in significant or even total loss of market share (ask Apple about the Newton)
 - ❖ Cost a lot of money to repair in production (10X increases not unusual)
 - ❖ Waste software engineering team time (should build new features rather than fix bugs)
 - ❖ Delay the release of software (lowers profits and market share)
 - ❖ Create safety issues (for certain systems)
- ❖ We know a lot about how to build and delivery software quality...we just don't always do it



Six Proven Lessons

- ❖ The following pages illustrate six lessons learned in building and delivering quality
 - ❖ Case studies show both success and failure to apply the lessons
 - ❖ I indicate tips for success with ⚡
 - ❖ I indicate warnings for common mistakes with ⚠
- ❖ All figures and case studies are real thanks in many cases to the kind permission of my clients and my associates



Six Lessons in Software Quality

- ⊕ Good, testable requirements yield large, often unsuspected dividends
- ⊕ Manage key quality risks throughout the lifecycle
- ⊕ Don't underestimate the value and importance of integration and integration testing
- ⊕ You can't achieve quality without teamwork
- ⊕ You can't achieve quality without competent team members
- ⊕ Manage change, even if you embrace it

I illustrate these lessons through anecdotes (most first hand) of these lessons successfully applied—and painfully ignored!



Lesson 1: Gifts from Good Requirements

- ⊕ A good set of sufficiently-detailed, well-understood, testable requirements is the foundation of quality
- ⊕ In some cases, the dividends from such requirements can exceed expectations
- ⊕ Conversely, in other cases, the problems created by bad requirements can be a long-term source of unpleasant surprises

⚡ If we understand the requirements, we can build quality

△ The Agile Manifesto doesn't say, "No specified requirements"



Long Term Costs of Bad Requirements

- ❖ One company that creates medical software had a serious requirements problem
- ❖ Requirements were vague, gappy, untestable, and poorly managed
- ❖ They discovered that dozens of their systems had serious data integrity issues that created safety risks for patients
- ❖ They have spent hundreds of thousands of dollars trying to resolve the problem
- ❖ They are now focused on better requirements... better late than never, but still



Long Term Gifts of Good Requirements

- ✦ A client in the medical systems business has a machine-readable template used for requirements
- ✦ This template makes generating complete, testable requirements easy
- ✦ It also ties to their traceability system for easier FDA audits
- ✦ Since the requirements are machine-readable, we were able to create a test tool that uses the requirements to automatically generate and execute an infinite variety of tests

✦ Read about the test tool on the Articles page of www.rbc-us.com in “Quality Goes Bananas”



Lesson 2: Manage Risks Start to Finish

- ✦ Many quality risks are evident during requirements and design
- ✦ A single bad component can affect the entire system
- ✦ Interface problems between components can create a wide variety of failures
- ✦ Badly designed system architectures can't be patched into perfection, so get design bugs out early

✦ Use models to identify and remove quality risks during design

✦ Cover quality risks in component, integration, and system test

△ Throwing more hardware at design problems might not work



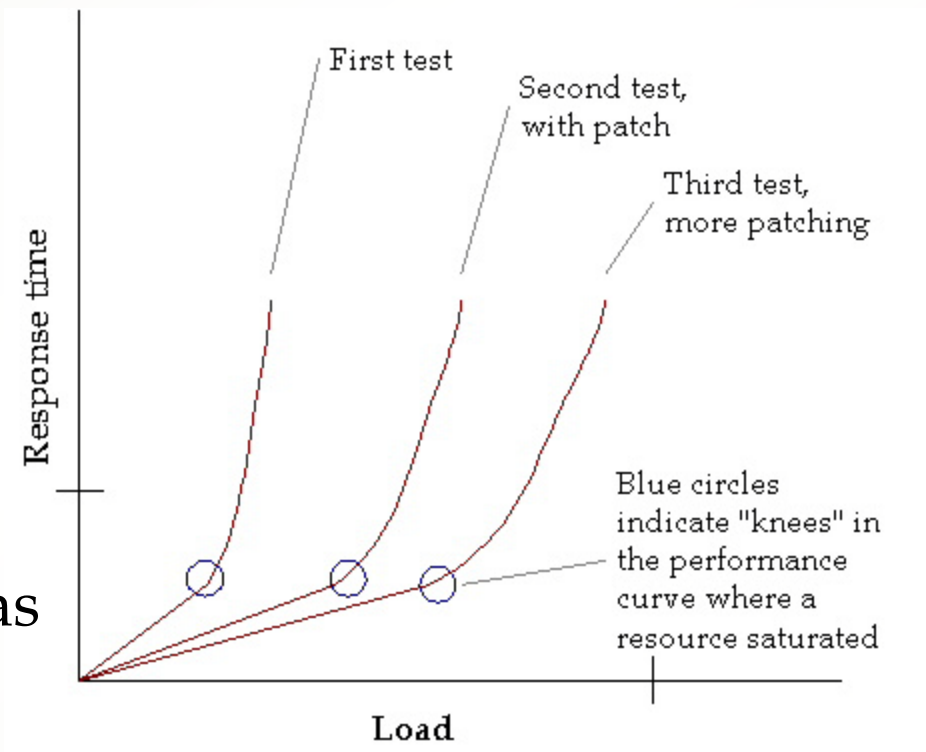
Quality Risk Management, but Too Late

- ✦ On one project, a senior technical staff member had built a spreadsheet model that predicted many of the performance problems we found in later testing
- ✦ However, there was no follow-up during design, implementation, or component testing
- ✦ Late discovery of performance problems during integration and system testing lead to patching of server performance problems...



Problems with Patching Design Bugs

- ❖ Patching a problem would improve performance
- ❖ However, the next bottleneck would be discovered during testing
- ❖ “Peeling the onion” was a slow process which didn’t resolve the performance problems



Lines on the axes indicate desired response time and supported load level



Test Early, Test Often

- ✦ On another project, we started with a spreadsheet during initial server system design
- ✦ That initial server design was turned into a simulation, which was fine-tuned as detailed design work continued
- ✦ Performance testing of units and subsystems was compared against the simulation
- ✦ Few surprises were found during system testing, when the simulation was mostly confirmed

✦ Iteratively improve the models and the tests to resolve discrepancies between the predicted and observed results



Double-Checking

- ⊕ An extract from our performance results presentation
- ⊕ Simulation data for IMAP servers
 - ⊞ 45% server utilization at 25K users
 - ⊞ 75% server utilization (saturation) at 40K users
- ⊕ Worst-case snapshots (25K users)

Server	CPU Idle
MTA1	68%
MTA2	79%
Maildb1	67%
Maildb2	89%
IMAP1	59%
IMAP2	55%

∴ Test data supports simulation results @ 25K users

⚡ Read more details on the Basic Library page of www.rbc-us.com in “The Right Stuff”



Lesson 3: Integration and Integration Testing

- ✦ Units that work by themselves don't necessarily work together
- ✦ Units that worked with older versions of software don't always work with newer versions
- ✦ Quality risks can accumulate when we don't have integrated, running software
- ✦ Quality risks can accumulate when we don't test control flows and data flows across interfaces

✦ Use continuous integration to detect broken builds, interfaces

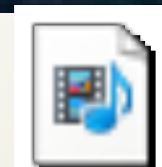
✦ Use automated unit testing frameworks for integration testing

△ Late discovery of integration problems can delay projects



Integration Mistakes on the Way to Space

- ✦ When one piece of the Arienne 5 software used big, strange words (64 bit floating vs. 16 bit integer), it confused another piece of software
- ✦ One new piece of software, one old piece of software
- ✦ Tip over, go sideways, blow up
- ✦ There goes US\$ 500M
- ✦ Oops



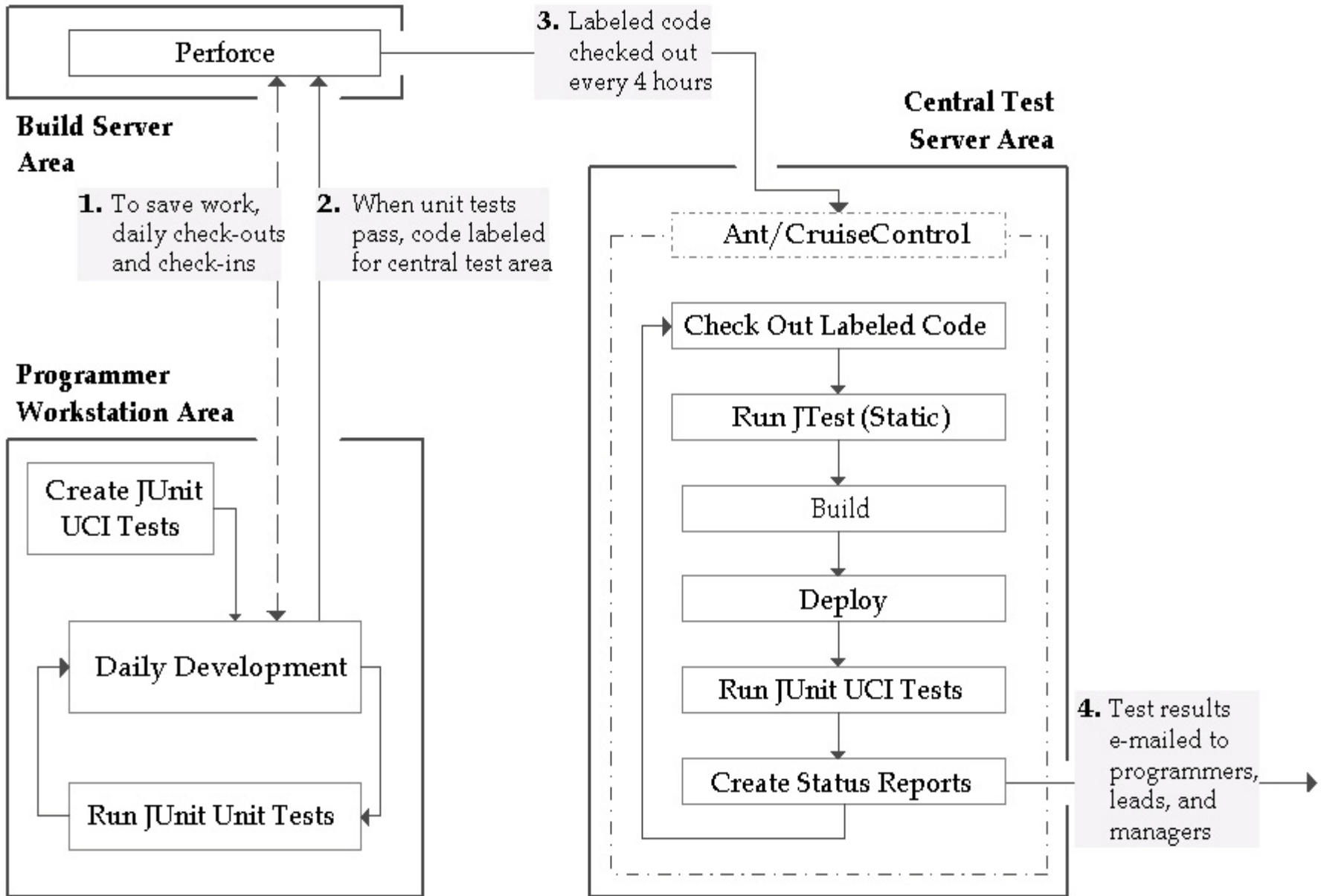
ariane 5 video.mp4



Proper Testing, Including Integration

- ✦ For a client, we implemented an automated unit, component, and integration test tool and process
 - ❖ Unit testing: Performed directly at the public and private interfaces
 - ❖ Component testing: Performed at the services and behaviors of the component under test, which could interact with other components
 - ❖ Integration testing: Performed at the interfaces and behaviors of communicating components, whether in the same layer or in different layers
- ✦ Continuous integration
- ✦ Testing occurred every four hours based on new code

⚡ Read more details on the Basic Library page of www.rbc-us.com in “Mission Made Possible”





Lesson 4: Teamwork Essential for Quality

- ✦ Quality is built into the product step by step
- ✦ Every person on the project team has a role
- ✦ All project groups (REs, BAs, dev, test, users, marketing, sales, tech support) must work together to achieve a quality product
- ✦ Strife between individuals or teams undermines quality

✦ Establish quality as a common objective for *all* project groups

△ Damaged work relationships are difficult to repair



Rather Fight than Win

- ✦ For one client, we found antagonism between the teams during a quality process assessment
 - ✦ Non-test groups: “Testing has no value.”
 - ✦ Test group: “Business analysts and programmers are lazy and take shortcuts that damage quality.”
- ✦ No one would work together to improve quality
- ✦ Our assessment report to executives...
 - ✦ Good news: 18-24 months away from world class quality
 - ✦ Bad new: Can't get there with the current team
- ✦ Ultimately, the executives outsourced all IT work



Achieving Quality Together

- ❖ For another client, we worked on a project to deliver a leading edge computer system
 - ❖ Requirements and design in San Jose, CA
 - ❖ System vendor in Taiwan
 - ❖ Component vendor in Salt Lake City, UT
 - ❖ Test lab in Taiwan
 - ❖ Test lab in Los Angeles, CA
- ❖ By coordinating activities across all groups and working as a team, we delivered on-time, in budget, with high quality



Lesson 5: Competent Team Members

- ✦ Not only must people work well together, they must also work well individually
- ✦ People without the proper skills cannot build quality software
- ✦ Part of managing software quality is managing the skills of the people building it

✦ Skills can be assessed via a skills inventory, as described in my book *Managing the Testing Process, 3e*.

△ Intellectually complex tasks require 10,000 hrs. (= 5 yrs.) of experience to master



Web Site Development: Failure

- ✦ In 2007, RBCS hired a company (using a marketing firm) to re-build our web site
- ✦ I didn't interview the people who would do the work, trusting the marketing firm to check skills and references
- ✦ Big mistake!
- ✦ It took over four months to finish development, with most of the time spent fixing bugs and testing
- ✦ After development, the site was plagued with quality problems
- ✦ Worse yet, we were locked in to the company because they'd built this site using proprietary technology



Web Site Development: Success

- ✦ In 2009, sick of the incompetence and quality problems, we hired another company to re-build our site
- ✦ I had experience with them through my work on the ASTQB, and knew them to be good
- ✦ Same requirements, same dev and test process, but different dev team
- ✦ While the project was not without bugs, the quality was much higher
- ✦ We continue to use them – through choice, not lock-in – for ongoing web site maintenance

✦ Turnover introduces known and unknown skills gaps into teams, so keep your good people.

△ Avoid outsourcing critical decisions that affect the quality of externally-visible assets (like company web sites).



Lesson 6: Manage Change

- ⊕ Change is unavoidable during projects
- ⊕ However, that does not mean that we should accept all proposed changes without any consideration
- ⊕ Smart organizations make smart changes
- ⊕ Other organizations allow themselves to thrash until the last days of a project, responding to every whim and suggestion

⚡ Understand the impact of change across all teams

△ All change has costs and risks, not only opportunities and benefits



Any Change, Any Time, No Control

- ✚ During UAT on one project, users were told, “Anything you don’t like, report it as a bug, we’ll fix it.”
- ✚ There were no documented requirements
- ✚ UAT was scheduled for six weeks
- ✚ Users reported over 800 bugs (most were really change requests)
- ✚ UAT ballooned to twelve weeks (100% schedule overrun)
- ✚ Project failed, with lack of change management the major cause



Careful Change Management

- ❖ During system testing on another project, we had a cross-functional team to triage bugs and evaluate changes
- ❖ All groups assessed changes and participated in change decisions
- ❖ Rate of change was high early in testing, but reduced as we neared the end (and the risks of changes increased)
- ❖ Risks and costs of changes were balanced against opportunities and benefits



Where Does Your Quality Stand?

- ✦ Assess your quality and testing practices:
 - ❑ How well – or poorly – does your company do in applying each of these lessons?
 - ❑ Which common mistakes are you making?
 - ❑ Which tips for success could you apply?
- ✦ Avoid the real problems that poor quality can produce by assessing and improving your software quality throughout the lifecycle

△ Don't let field failures be your first sign of problems

⚡ Direct questions to info@rbc-us.com



...*Contact RBCS*

For over a dozen years, RBCS has delivered services in consulting, outsourcing and training for software and hardware testing. Employing the industry's most experienced and recognized consultants, RBCS conducts product testing, builds and improves testing groups and hires testing staff for hundreds of clients worldwide. Ranging from Fortune 20 companies to start-ups, RBCS clients save time and money through improved product development, decreased tech support calls, improved corporate reputation and more. To learn more about RBCS, visit www.rbc-us.com.

Address: RBCS, Inc.
31520 Beck Road
Bulverde, TX 78163-3911
USA

Phone: +1 (830) 438-4830

Fax: +1 (830) 438-4831

E-mail: info@rbc-us.com

Web: www.rbc-us.com