同济大学软件学院
School of Software Engineering, Tongji University

Software Testing Foundation Level

Chapter 5: Test Management

刘琴 Qin Liu

2023-04-25

5. Test Management

1. Test organization
2. Test planning and estimation
3. Test progress monitoring and control
4. Configuration management
5. Risk and testing
6. Incident management

Chapter 5:
Test Management

Section 1:
Test organization

Test Organization

- Key concepts
  - – The importance of independent testing
  - – The benefits and drawbacks of independent testing
  - – Different team members to be considered for the organization of a test team
  - – Tasks of typical test leader and tester
- Terms to remember

Spectrum of Independence

- No independent testers, developers test their own code
- Independent testers within the development teams
- Independent test team or group within the organization, reporting to project management or executive management.
- Independent testers from the business organization or user community
- Independent test specialists for specific test targets such as usability testers, security testers or certification testers
- Independent testers outsourced or external to the organization

Why Independence?

- Given complex and/or critical applications, one needs…
  - Multiple levels of testing
  - Some or all of the levels done by independent testers
- Development testing, while important, is typically is much less effective at finding bugs (Jones quotes 50% as max. DRE)
- If independent testers want/have the authority to require and define processes and rules, they should have clear direction

Independence Considerations

Benefits

- See more, other and different defects
- If in doubt, it's a bug
- Verify assumptions of specification and implementation
- Credibility
- Tester career path

Potential traps

- Isolation from the development team
- Seen as bottleneck
- Programmers lose sense of responsibility for quality

Test Leads

- Devise test strategies, plans
- Write or review test policy
- Consult on testing for other project activities
- Test estimation
- Test resource acquisition
- Lead specification, preparation, implementation, and execution of tests
- Monitor and control the test execution
- Adapt the test plan based on test results
- Ensure configuration management of testware
- Ensure traceability
- Measure test progress, evaluate the quality of the testing and the product
- Plan any test automation
- Select tools and organize any tester training
- Ensure implementation of the test environment
- Schedule tests
- Write test summary reports

Testers

- Review and contribute to test plans
- Analyze, review and assess user requirements, specifications
- Create test suites, cases, data and procedures
- Set up the test environment
- Implement tests on all test levels
- Execute and log the tests, evaluate results and document problems found
- Monitor testing using the appropriate tools
- Automate tests
- Measure performance of components and systems
- Review each others’ tests

Refining the Tester Position

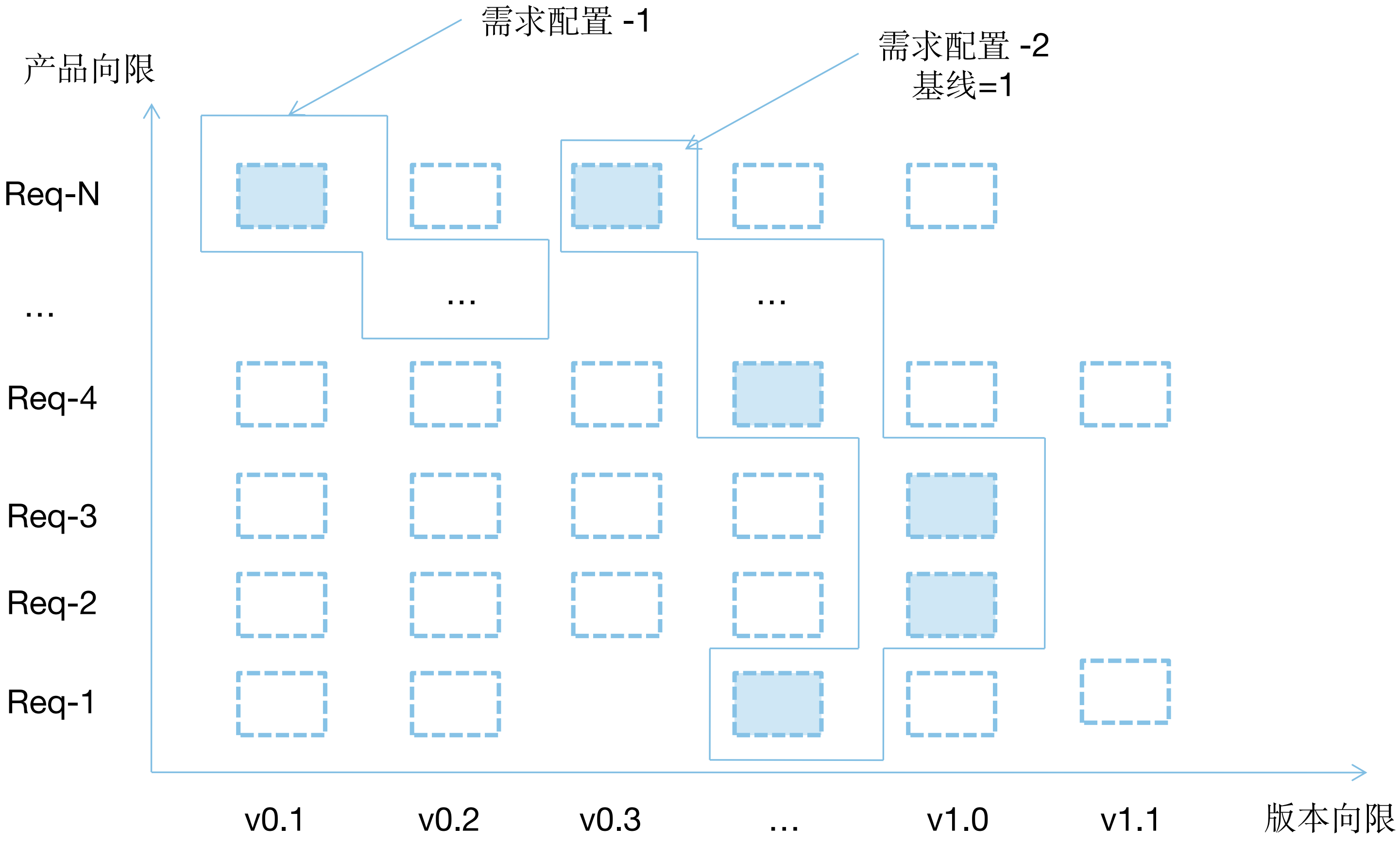
- Test engineers
  - Technical peers of programmers
  - Chose testing as a specialty
  - Write test cases, organize test suites
  - Create, customize and use advanced test tools
  - Have unique skills

The right positions for your team depends on the skills you need.

Be sure to balance skills and positions across the test team—skills are complimentary and the whole can be greater than the sum of the parts.

- Test technicians
  - Skilled and experienced tester
  - May be an aspiring test engineer
  - Runs tests
  - Reports bugs
  - Updates test status
  - Assists the test engineers
- Other test team members
  - System and database administrators
  - Release and configuration engineers
  - Test toolsmiths

Testing vs. Requirement Management



需求配置 -1

需求配置 -2
基线=1

产品向限

Req-N

…

Req-4

Req-3

Req-2

Req-1

…

…

v0.1

v0.2

v0.3

…

v1.0

v1.1

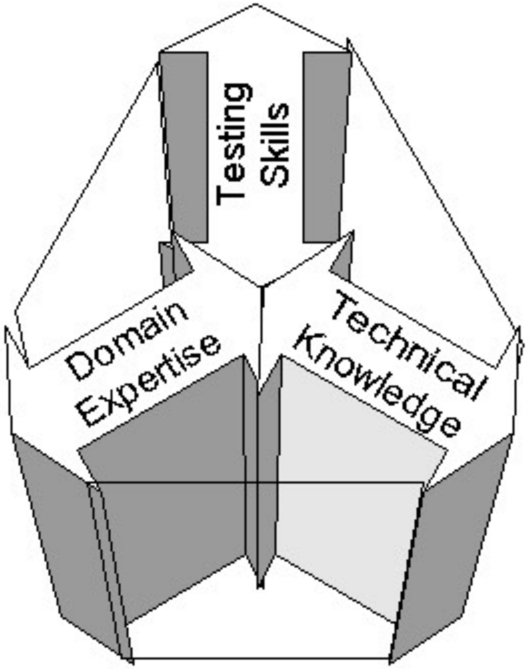
版本向限

Using Amateur Testers

- On many projects, amateur testers (people who do not test for their living) are used as part (or all) of the test team
- Such people typically include project managers, quality managers, programmers, business and domain experts, or infrastructure or IT operators
- Such a test team often possesses strong skills in some areas (business domain or technology), but weak skills in others, and no skills or substantial experience with testing
- Test is a special field, with special skills (of which the basics are covered in this course)

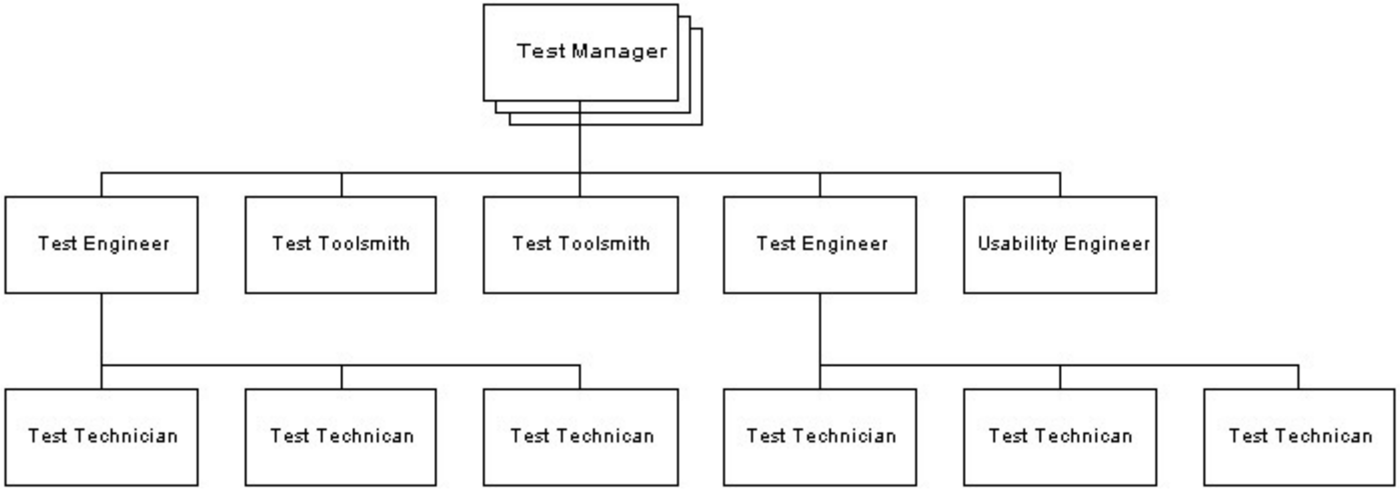
Balancing the Skills

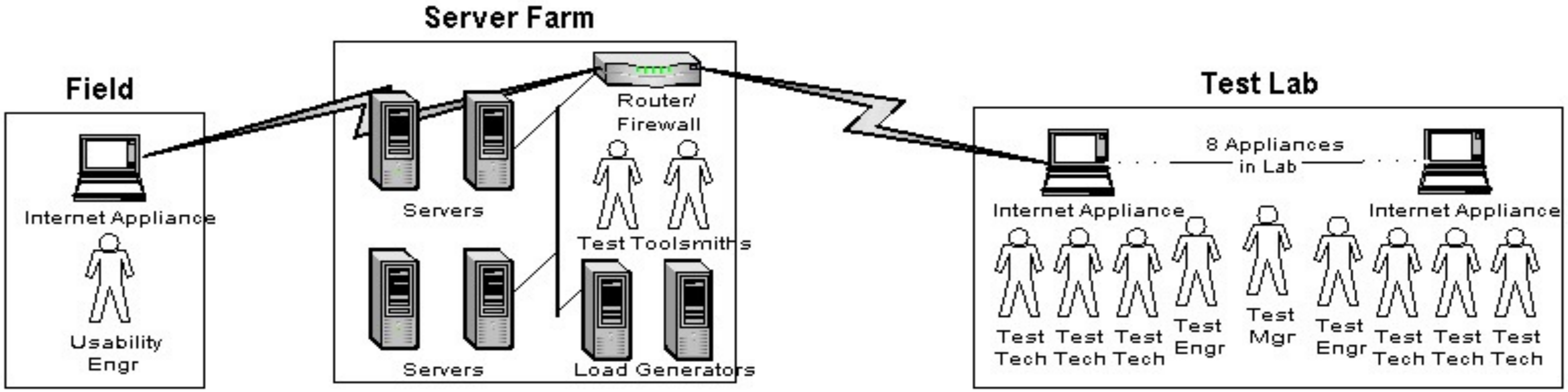
- Good test teams have right mix of skills based on tasks and activities
- Application (business) domain expert
  - Understands intended behavior
- Skilled tester
  - Knows quality risks and test techniques
- Technical guru
  - Aware of technical issues and limitations
- What is the right mix for…
  - …Internet appliance testing?
  - …nuclear medicine testing?
  - …your project?



The appropriate depth and length of each arrow in the figure depends on the project, process, and product

Case Study: Positions, Organization, Project





Chapter 5:
Test Management

Section 1:
Exercise

Exercise: Omninet Test Team

- Imagine you are the test manager for the Omninet project, in charge of integration and system testing.
- Sketch an organizational chart that shows how you would organize the Omninet test team.
- Where would the test team fit into the Omninet project team?
- Discuss.

Chapter 5:
Test Management

Section 2:
Test planning and estimation

Test Planning and Estimation

- Key concepts
  - Levels and objectives of test planning
  - Purpose and content of the test plan
  - Test strategies
  - Test schedules and prioritization of test cases
  - Test planning for a project, levels, targets
  - Test preparation and execution for the plan
  - Test exit criteria
  - Estimation via metrics and expertise
  - Test estimation factors
- Terms to remember

Developing Test Plans

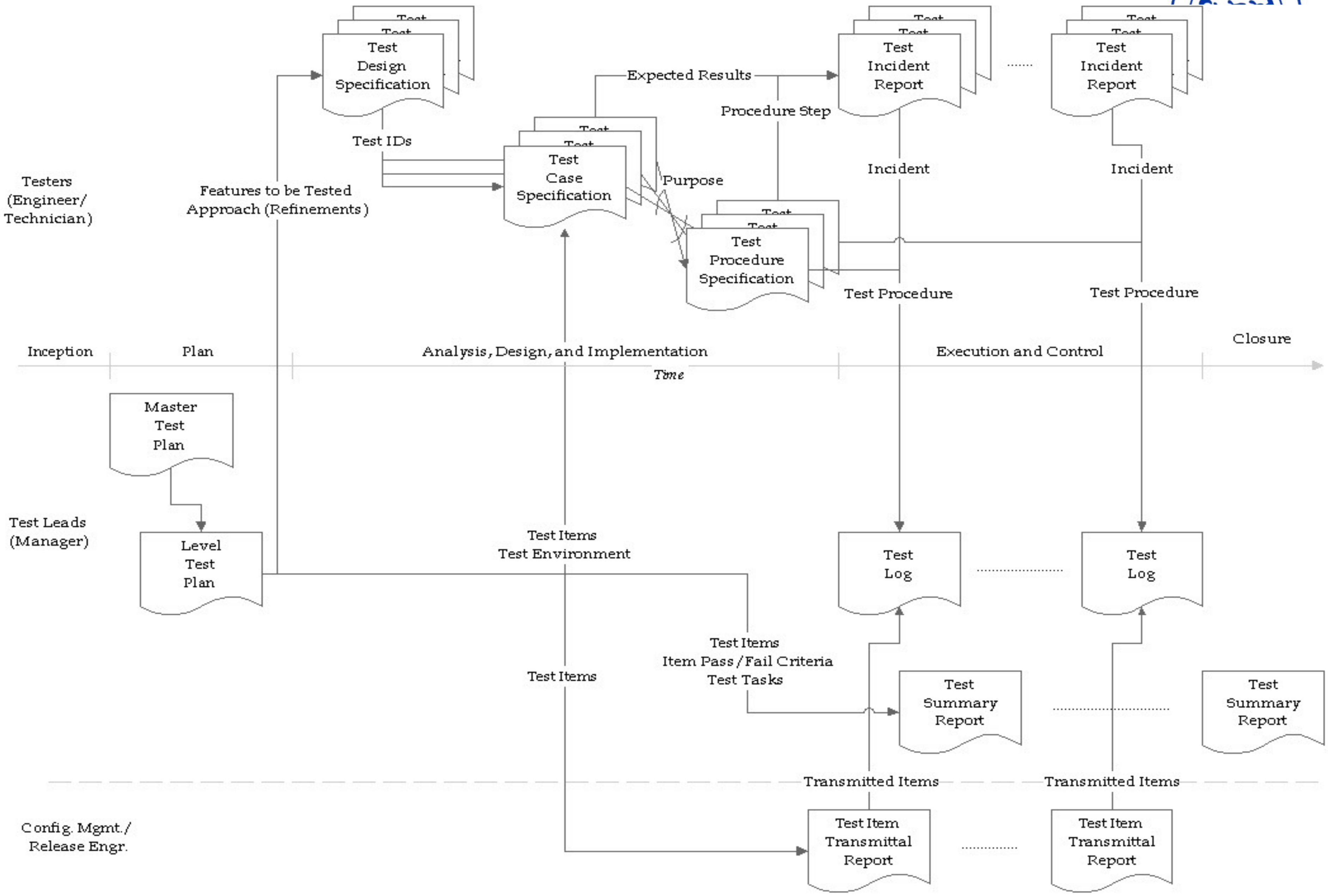
- Why write (and update) test plans?
  - Confront challenges, crystallize thinking, adapt to change
  - Communicate plan to testers, peers, managers
- Consider multiple test plans when tests have…
  - Different time periods (e.g., phases and levels)
  - Different methodologies and tools (e.g., performance and functionality)
  - Different objectives (e.g., system test and beta test)
  - Different audiences (e.g., hardware and software test)
- …but then you may want a master test plan
- Circulate one or two drafts
  - Promotes early feedback and discussion
  - Prevents wasted time if you're on the wrong track

Test Planning Activities

- Define test approach, test levels
- Integrate, coordinate testing into the life cycle
- Decide who, what, when, how of testing
- Assign resources for test tasks
- Define the test documentation
- Set the level of detail for test cases, procedures in order to provide enough information to support reproducible test preparation and execution
- Select test monitoring, controlling, and reporting metrics, charts, and reports (deliverables)

IEEE 829 Test Plan

- A test plan is a subproject plan for the testing part of a project, and includes the sections shown in the next slide
- You can adapt the IEEE 829 outline for use for each detail (e.g., level or phase) test plan as well as the master test plan
- You can create you own template or outline, too
- Test planning influences (and is influenced by) test policy of the test organization, the scope of testing, objectives, risks, constraints, criticality, testability, and the availability of resources



Testers (Engineer/ Technician)

Test Leads (Manager)

Config. Mgmt./ Release Engr.

Inception

Plan

Analysis, Design, and Implementation

Execution and Control

Closure

Time

Master Test Plan

Level Test Plan

Features to be Tested Approach (Refinements)

Test Design Specification

Test IDs

Test Case Specification

Expected Results

Procedure Step

Purpose

Test Procedure Specification

Test Incident Report

Test Incident Report

Incident

Incident

Test Procedure

Test Procedure

Test Items Test Environment

Test Items

Test Items Item Pass/Fail Criteria Test Tasks

Test Log

Test Log

Test Summary Report

Test Summary Report

Transmitted Items

Transmitted Items

Test Item Transmittal Report

Test Item Transmittal Report

IEEE 829 Test Plan Outline

- Test plan identifier
- Introduction
- Test items (i.e., what's delivered for testing)
- Features to be tested
- Features not to be tested
- Approach (strategies, organization, extent of testing)
- Item pass/fail criteria
- Test criteria (e.g., entry, exit, suspension and resumption)
- Test deliverables (e.g., reports, charts, etc.)
- Test tasks (or at least key milestones)
- Environmental needs
- Responsibilities
- Staffing and training needs
- Schedule
- Risks and contingencies (quality [product] and project risks)
- Approvals

Transitions: Entry Criteria

- Entry criteria measure whether the system is ready for a particular test phase
  - Deliverables ready?
  - Lab ready?
  - Teams ready?
- These tend to become increasingly rigorous as the phases proceed

Sample Entry Criteria

System Test can begin when:

1. Bug tracking and test tracking systems are in place.

2. All components are under formal, automated configuration and release management control.

3. The Operations team has configured the System Test server environment, including all target hardware components and subsystems. The Test Team has been provided with appropriate access to these systems.

4. The Development Teams have completed all features and bug fixes scheduled for release.

5. The Development Teams have unit-tested all features and bug fixes scheduled for release.

6. Less than 50 must-fix bugs (per Sales, Marketing, and Customer Service) are open against the release.

7. The Development Teams provide software to the Test Team 3 business days prior to starting System Test.

8. The Test Team completes a 3 day "smoke test" and reports on the results to the System Test Phase Entry meeting.

9. The Project Management Team agrees in a System Test Phase Entry Meeting to proceed. The following topics will be resolved in the meeting:

- ✓ *Whether code is complete.*
- ✓ *Whether unit-testing is complete.*
- ✓ *Assign a target fix date for any known "must-fix" bugs (no later than 1 week after System Test Phase Entry).*

Transitions: Continuation Criteria

- Continuation criteria measure whether testing can efficiently, effectively proceed
  - Test environment problems
  - Test-blocking bugs in system under test
- "Continuation criteria" is a polite way of saying "stopping criteria" in the reverse
  - Stopping a test phase is seldom popular

Sample Continuation Criteria

System Test will continue if:
1. All software released to the Test Team is accompanied by Release Notes.
2. No change is made to the system, whether in source code, configuration files, or other setup instructions or processes, without an accompanying bug report. Should a change be made without a bug report, the Test Manager will open an urgent bug report requesting information and escalate to his manager.
3. The open bug backlog ("quality gap") remains less than 50. The daily and rolling closure periods remain less than 14 days (on average, bugs are fixed within two weekly release cycles).
4. Twice-weekly bug review meetings occur until System Test Phase Exit to manage the open bug backlog and bug closure times.

Transitions: Exit Criteria

- Exit criteria measure whether the test phase can be deemed complete
  - Thoroughness measures, such as coverage of code, functionality or risk
  - Estimates of defect density or reliability measures
  - Cost
  - Residual risks, such as defects not fixed or lack of test coverage in certain areas
  - Schedules such as those based on time to market
- Remember that these are business decisions

Sample Exit Criteria

System Test will end when:

1. No changes (design/code/features), except to address System Test defects, occurred in the prior 3 weeks.

2. No panic, crash, halt, wedge, unexpected process termination, or other stoppage of processing has occurred on any server software or hardware for the previous 3 weeks.

3. No client systems have become inoperable due to a failed update during System Test.

4. The Test Team has executed all the planned tests against the GA-candidate software.

5. The Development Teams have resolved all "must-fix" bugs per Sales, Marketing, and Customer Service.

6. The Test Team has checked that all issues in the bug tracking system are either closed or deferred, and, where appropriate, verified by regression and confirmation testing.

7.. The test metrics indicate: product stability and reliability; completion of all planned tests; adequate coverage of the critical quality risks.

8. The Project Management Team agrees that the product, as defined during the final cycle of System Test, will satisfy the customer's reasonable expectations of quality.

9. The Project Management Team holds a System Test Phase Exit Meeting and agrees that we have completed System Test.

Developing a Work-Breakdown-Structure

- What are the major stages of a testing subproject?
  - Planning
  - Staffing (if applicable)
  - Test environment acquisition and configuration
  - Test development
  - Test execution
- Break down to discrete tasks within each stage
- What test suites are required for the critical risks?
  - E.g., functionality, performance, error handling, etc.
- For each suite, what tasks are required?
- Tasks should be short in duration (e.g., a few days)
  - “Inch-pebbles” as well as “milestones”

Estimation

- There are two general approaches for estimation (including test estimation)
  - Estimating individual tasks by the owner of these tasks or by experts (bottom-up via work-breakdown-structure)
  - Estimating the testing effort based on metrics of former or similar projects or based on typical values (top-down or bottom up via parametric models, rules of thumb, etc.)
- While both are useful, drawing upon team wisdom to create a work-breakdown-structure, then applying models and rules of thumb to check and adjust the estimate, is usually more accurate (and more defensible)

Factors to Consider in Test Estimation

- Testing is complex, influenced by:
  - Process factors: pervasive testing, change control, process maturity, lifecycle, development and test processes, earlier test phases, (estimated vs. actual) bug levels and fixing
  - Material factors: tools, test system, test and debugging environments, project documentation, similarity
  - People factors: skills, expectations, support, relationships
  - Delaying factors: complexity, many stakeholders, too much newness, geographical distribution, need for detailed test documentation, tricky logistics, fragile test data
- Understand estimation techniques and these factors
- Deviation from the test estimate can arise from outside factors and events before or during testing

Test Strategies

- Analytical: e.g., risk-based and requirement-based
- Model-based: e.g., stochastic or statistical testing
- Methodical: following a pre-planned methodology, e.g., failure-based, checklist-based, and quality-based
- Process/standard-compliant: e.g., IEEE 829 standard or Agile methodology like test-driven development
- Dynamic: an approach more reactive to events and conditions than pre-planned or pre-designed
- Consultative or directed: test coverage driven primarily by outside (non-tester) advice, guidance
- Regression-averse: reuse of existing test material, extensive automation of tests, standard test suites

Chapter 5:
Test Management

Section 2:
Exercise

A2.1.10: Omninet Test Plan

- Outline a test plan for Omninet
- Using the IEEE 829 template as a starting point, write down two to five bullet items or high-level statements in each section.
- Discuss.

Chapter 5:
Test Management

Section 3:
Test progress monitoring and control

Test Progress Monitoring and Control

- Key concepts
  - Common metrics used for monitoring test preparation and execution
  - Understand and interpret test metrics
  - The purpose and content of the test report document according to IEEE 829 standard
- Terms to remember

Good Thinking!

- **What measurement** would be helpful to monitor the test progress?
  - Schedule?/process?(test level matching)
  - Actual starting date is delayed/adjusted?
  - Not good enough?
  - Test results pass rate! Coverage of the Test Cases!! (Tester provide reports) Bug Reporting Rates? Fix Rate? Workload.
- Test Requr, Test Object, Test Case, Coverage, Test Results, testers?

Test Leads

- Devise **test strategies**, plans
- Write or review **test policy**
- Consult on testing for **other project activities**
- Test estimation
- **Test resource** acquisition
- Lead **specification, preparation, implementation, and execution of tests**
- Monitor and control the test execution
- Adapt the test plan based on **test results**
- Ensure configuration management of **testware**
- Ensure traceability
- Measure test progress, evaluate the quality of the testing and the product
- Plan any **test automation**
- Select **tools** and organize any tester training
- Ensure implementation of the **test environment**
- **Schedule** tests
- Write **test summary reports**

Common Test Metrics

- Percentage completion of test case preparation (or percentage of planned test cases prepared)
- Percentage completion of preparing the test environment
- Test case execution (e.g. number of test cases run/not run, and number of test cases passed/failed)
- Bug/Defect information (e.g. defect density, defects found and fixed, failure rate, and retest results)
- Coverage of requirements, risks or code by tests, including pass/fail results
- Level of confidence (subjective) of testers in the product
- Dates of test milestones
- Testing costs, including the cost of finding the next defect or running the next test compared to the benefit

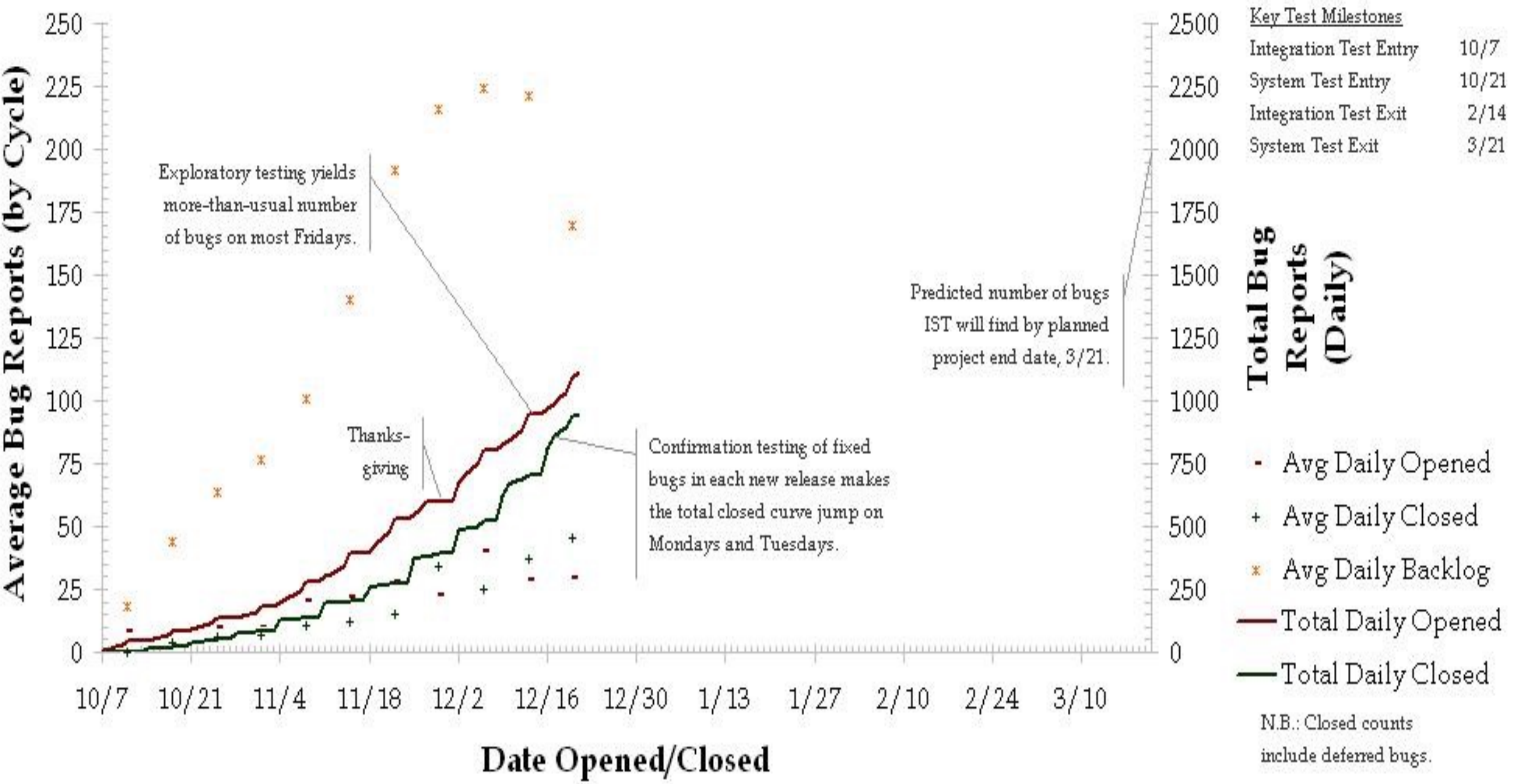
Test Reporting

- Summarize/analyze the test results
  - – Key events (e.g., meeting exit criteria)
  - – Analysis (for recommendations, guidance) of…
    - …remaining defects
    - …cost/benefit of more testing
    - …outstanding risks
    - …level of confidence
- Metrics gathered to assess:
  - – Test objectives adequacy for test level
  - – Test approaches adequacy
  - – Testing effectiveness per objectives

Test Control

- Guiding and corrective actions due to test information and metrics, which may affect testing activities—or other software life cycle activities
- Examples of **test control actions**:
  - Risk-triggered test re-prioritizing
  - Test schedule adjustments due to availability of a test environment
  - Set an entry criterion requiring re-testing of bug fixes by developers before integration into a build

Sumatra Integration and System Test Execution Phases
System Quality Problems Analysis



Key Test Milestones

Integration Test Entry	10/7
System Test Entry	10/21
Integration Test Exit	2/14
System Test Exit	3/21

Exploratory testing yields more-than-usual number of bugs on most Fridays.

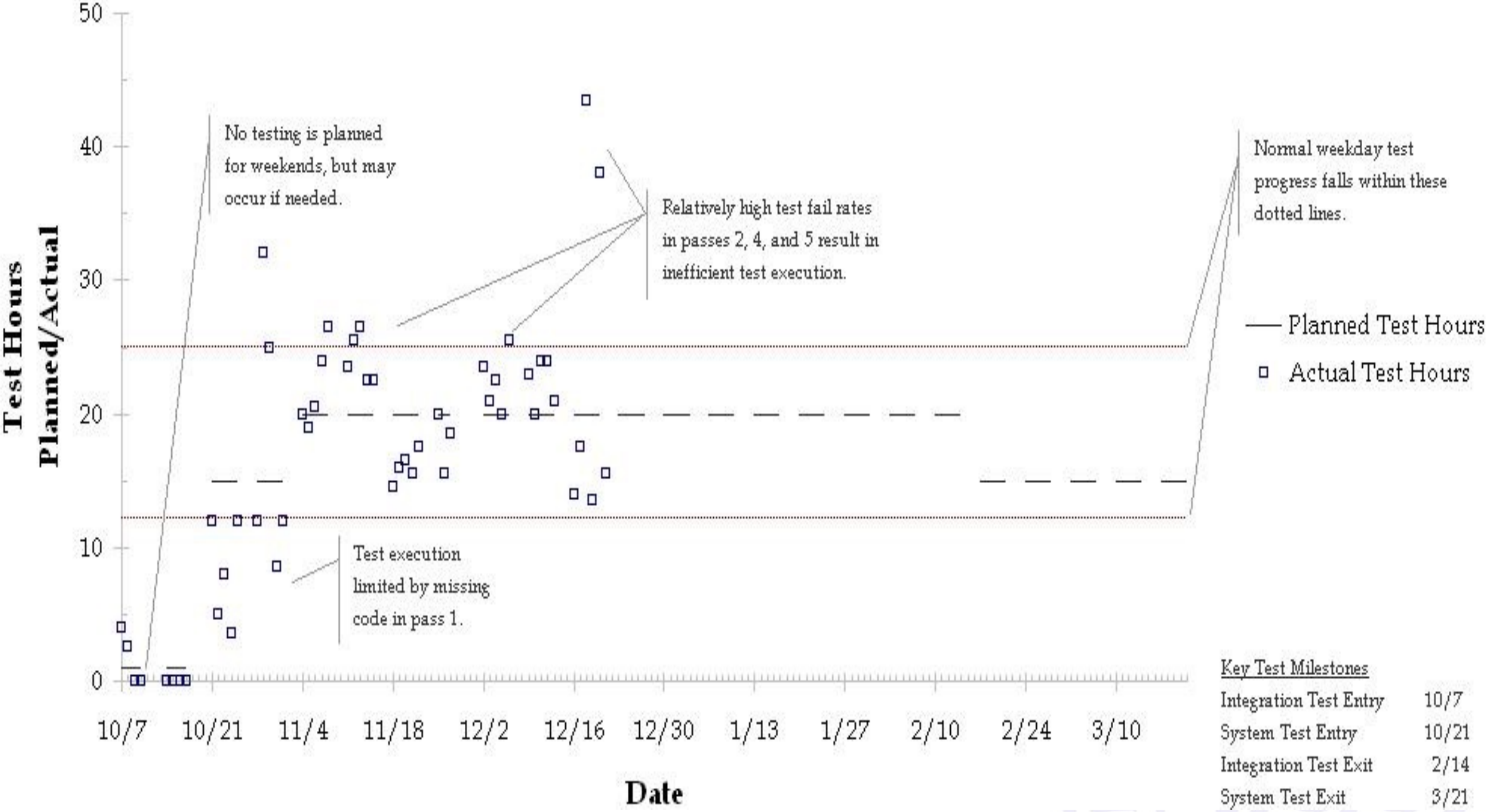
Thanks-giving

Confirmation testing of fixed bugs in each new release makes the total closed curve jump on Mondays and Tuesdays.

Predicted number of bugs IST will find by planned project end date, 3/21.

N.B.: Closed counts include deferred bugs.

Sumatra Integration and System Test Execution Phases
Test Progress



No testing is planned for weekends, but may occur if needed.

Relatively high test fail rates in passes 2, 4, and 5 result in inefficient test execution.

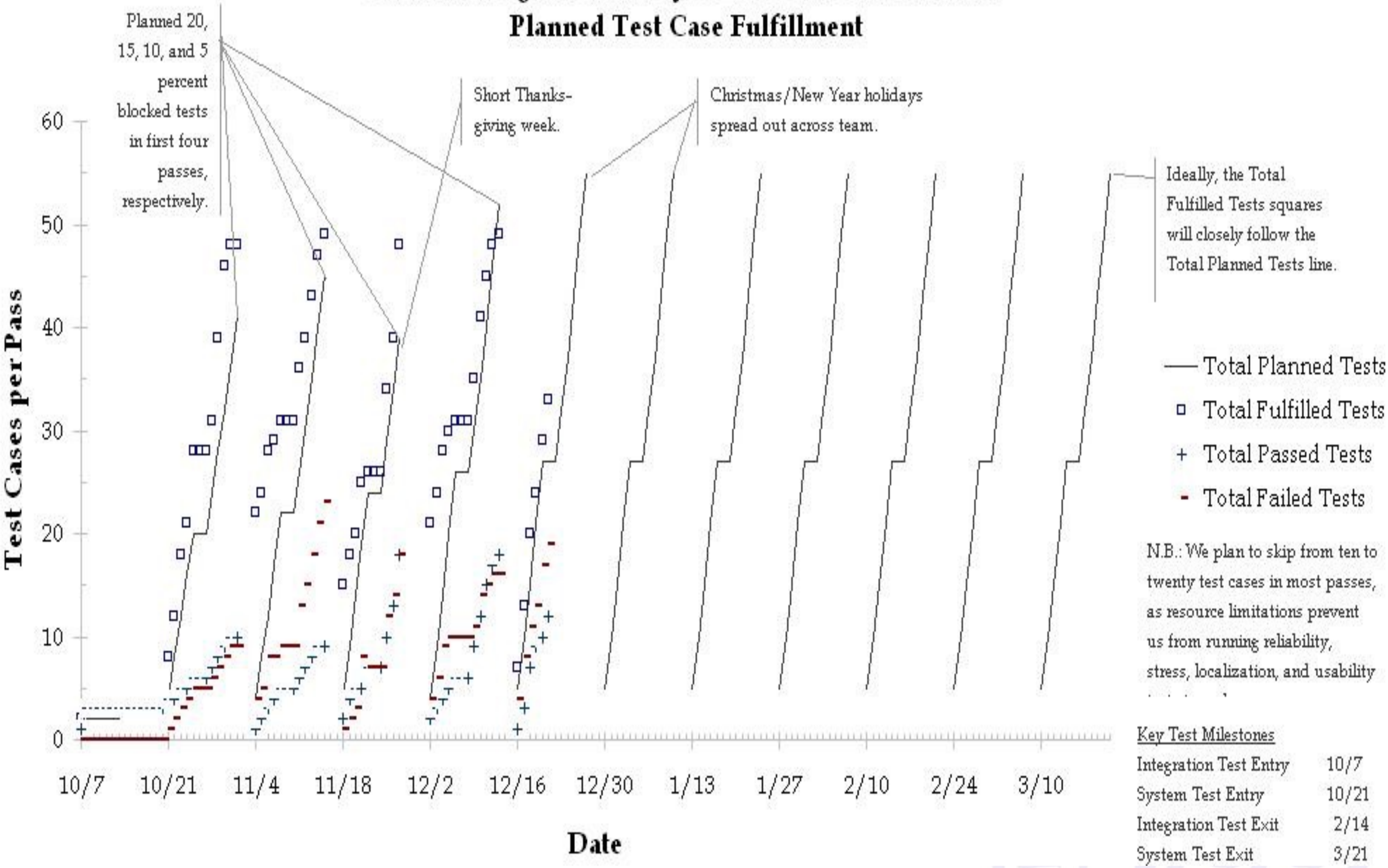
Normal weekday test progress falls within these dotted lines.

Test execution limited by missing code in pass 1.

Key Test Milestones

Integration Test Entry	10/7
System Test Entry	10/21
Integration Test Exit	2/14
System Test Exit	3/21

Sumatra Integration and System Test Execution Phases
Planned Test Case Fulfillment



Planned 20, 15, 10, and 5 percent blocked tests in first four passes, respectively.

Short Thanks-giving week.

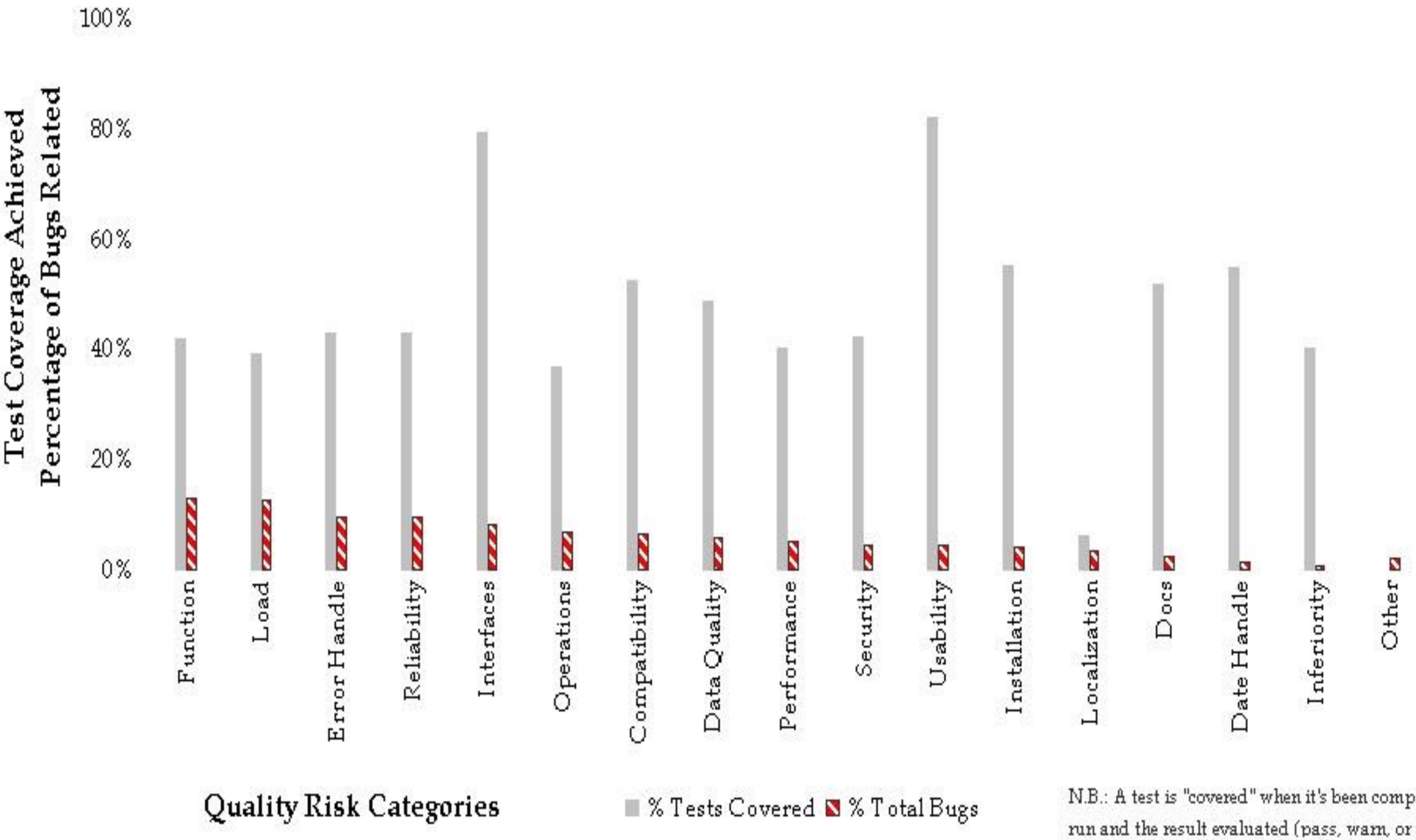
Christmas/New Year holidays spread out across team.

Ideally, the Total Fulfilled Tests squares will closely follow the Total Planned Tests line.

N.B.: We plan to skip from ten to twenty test cases in most passes, as resource limitations prevent us from running reliability, stress, localization, and usability

Key Test Milestones	
Integration Test Entry	10/7
System Test Entry	10/21
Integration Test Exit	2/14
System Test Exit	3/21

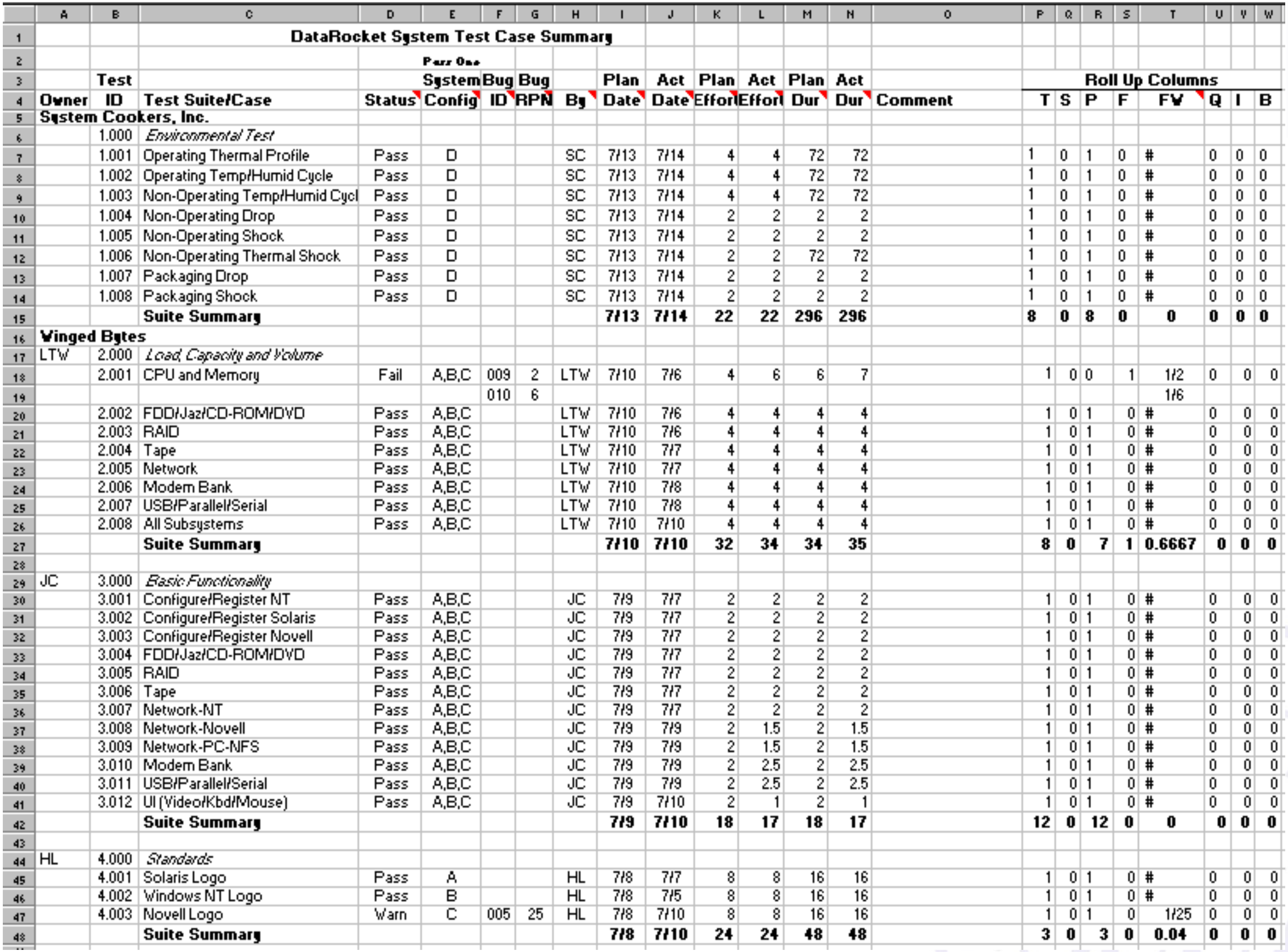
Sumatra Integration and System Test Execution Phases
Quality Risks Bugs and Test Coverage



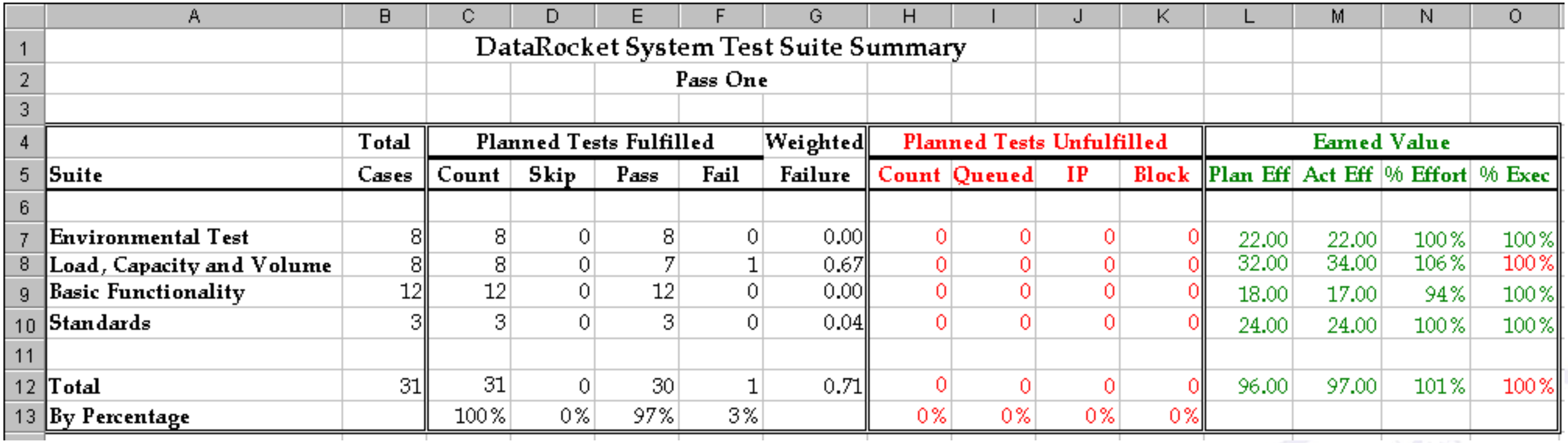
N.B.: A test is "covered" when it's been completely run and the result evaluated (pass, warn, or fail).

IEEE 829 Test Summary Report

- A test summary report describes the results of a given level or phase of testing, and includes the following sections
  - Test summary report identifier
  - Summary (e.g., what was tested, what the conclusions are, etc.)
  - Variances (from plan, cases, procedures)
  - Comprehensive assessment
  - Summary of results (e.g., final metrics, counts)
  - Evaluation (of each test item vis-à-vis pass/fail criteria)
  - Summary of activities (resource use, efficiency, etc.)
  - Approvals
- May be delivered within or at end of a test level



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1			**DataRocket System Test Case Summary**																				
2					**Pass One**																		
3		**Test**			**System**	**Bug**	**Bug**		**Plan**	**Act**	**Plan**	**Act**	**Plan**	**Act**		**Roll Up Columns**							
4	**Owner**	**ID**	**Test Suite/Case**	**Status**	**Config**	**ID**	**RPN**	**By**	**Date**	**Date**	**Effor**	**Effor**	**Dur**	**Dur**	**Comment**	**T**	**S**	**P**	**F**	**FW**	**Q**	**I**	**B**
5	**System Cookers, Inc.**																						
6		1.000	*Environmental Test*																				
7		1.001	Operating Thermal Profile	Pass	D			SC	7/13	7/14	4	4	72	72		1	0	1	0	#	0	0	0
8		1.002	Operating Temp/Humid Cycle	Pass	D			SC	7/13	7/14	4	4	72	72		1	0	1	0	#	0	0	0
9		1.003	Non-Operating Temp/Humid Cycl	Pass	D			SC	7/13	7/14	4	4	72	72		1	0	1	0	#	0	0	0
10		1.004	Non-Operating Drop	Pass	D			SC	7/13	7/14	2	2	2	2		1	0	1	0	#	0	0	0
11		1.005	Non-Operating Shock	Pass	D			SC	7/13	7/14	2	2	2	2		1	0	1	0	#	0	0	0
12		1.006	Non-Operating Thermal Shock	Pass	D			SC	7/13	7/14	2	2	72	72		1	0	1	0	#	0	0	0
13		1.007	Packaging Drop	Pass	D			SC	7/13	7/14	2	2	2	2		1	0	1	0	#	0	0	0
14		1.008	Packaging Shock	Pass	D			SC	7/13	7/14	2	2	2	2		1	0	1	0	#	0	0	0
15			**Suite Summary**						**7/13**	**7/14**	**22**	**22**	**296**	**296**		**8**	**0**	**8**	**0**	**0**	**0**	**0**	**0**
16	**Winged Bytes**																						
17	LTW	2.000	*Load, Capacity and Volume*																				
18		2.001	CPU and Memory	Fail	A,B,C	009	2	LTW	7/10	7/6	4	6	6	7		1	0	0	1	1/2	0	0	0
19						010	6													1/6			
20		2.002	FDD/Jaz/CD-ROM/DVD	Pass	A,B,C			LTW	7/10	7/6	4	4	4	4		1	0	1	0	#	0	0	0
21		2.003	RAID	Pass	A,B,C			LTW	7/10	7/6	4	4	4	4		1	0	1	0	#	0	0	0
22		2.004	Tape	Pass	A,B,C			LTW	7/10	7/7	4	4	4	4		1	0	1	0	#	0	0	0
23		2.005	Network	Pass	A,B,C			LTW	7/10	7/7	4	4	4	4		1	0	1	0	#	0	0	0
24		2.006	Modem Bank	Pass	A,B,C			LTW	7/10	7/8	4	4	4	4		1	0	1	0	#	0	0	0
25		2.007	USB/Parallel/Serial	Pass	A,B,C			LTW	7/10	7/8	4	4	4	4		1	0	1	0	#	0	0	0
26		2.008	All Subsystems	Pass	A,B,C			LTW	7/10	7/10	4	4	4	4		1	0	1	0	#	0	0	0
27			**Suite Summary**						**7/10**	**7/10**	**32**	**34**	**34**	**35**		**8**	**0**	**7**	**1**	**0.6667**	**0**	**0**	**0**
28																							
29	JC	3.000	*Basic Functionality*																				
30		3.001	Configure/Register NT	Pass	A,B,C			JC	7/9	7/7	2	2	2	2		1	0	1	0	#	0	0	0
31		3.002	Configure/Register Solaris	Pass	A,B,C			JC	7/9	7/7	2	2	2	2		1	0	1	0	#	0	0	0
32		3.003	Configure/Register Novell	Pass	A,B,C			JC	7/9	7/7	2	2	2	2		1	0	1	0	#	0	0	0
33		3.004	FDD/Jaz/CD-ROM/DVD	Pass	A,B,C			JC	7/9	7/7	2	2	2	2		1	0	1	0	#	0	0	0
34		3.005	RAID	Pass	A,B,C			JC	7/9	7/7	2	2	2	2		1	0	1	0	#	0	0	0
35		3.006	Tape	Pass	A,B,C			JC	7/9	7/7	2	2	2	2		1	0	1	0	#	0	0	0
36		3.007	Network-NT	Pass	A,B,C			JC	7/9	7/7	2	2	2	2		1	0	1	0	#	0	0	0
37		3.008	Network-Novell	Pass	A,B,C			JC	7/9	7/9	2	1.5	2	1.5		1	0	1	0	#	0	0	0
38		3.009	Network-PC-NFS	Pass	A,B,C			JC	7/9	7/9	2	1.5	2	1.5		1	0	1	0	#	0	0	0
39		3.010	Modem Bank	Pass	A,B,C			JC	7/9	7/9	2	2.5	2	2.5		1	0	1	0	#	0	0	0
40		3.011	USB/Parallel/Serial	Pass	A,B,C			JC	7/9	7/9	2	2.5	2	2.5		1	0	1	0	#	0	0	0
41		3.012	UI (Video/Kbd/Mouse)	Pass	A,B,C			JC	7/9	7/10	2	1	2	1		1	0	1	0	#	0	0	0
42			**Suite Summary**						**7/9**	**7/10**	**18**	**17**	**18**	**17**		**12**	**0**	**12**	**0**	**0**	**0**	**0**	**0**
43																							
44	HL	4.000	*Standards*																				
45		4.001	Solaris Logo	Pass	A			HL	7/8	7/7	8	8	16	16		1	0	1	0	#	0	0	0
46		4.002	Windows NT Logo	Pass	B			HL	7/8	7/5	8	8	16	16		1	0	1	0	#	0	0	0
47		4.003	Novell Logo	Warn	C	005	25	HL	7/8	7/10	8	8	16	16		1	0	1	0	1/25	0	0	0
48			**Suite Summary**						**7/8**	**7/10**	**24**	**24**	**48**	**48**		**3**	**0**	**3**	**0**	**0.04**	**0**	**0**	**0**



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1			DataRocket System Test Suite Summary												
2					Pass One										
3															
4		Total	Planned Tests Fulfilled				Weighted	Planned Tests Unfulfilled				Earned Value			
5	Suite	Cases	Count	Skip	Pass	Fail	Failure	Count	Queued	IP	Block	Plan Eff	Act Eff	% Effort	% Exec
6															
7	Environmental Test	8	8	0	8	0	0.00	0	0	0	0	22.00	22.00	100%	100%
8	Load, Capacity and Volume	8	8	0	7	1	0.67	0	0	0	0	32.00	34.00	106%	100%
9	Basic Functionality	12	12	0	12	0	0.00	0	0	0	0	18.00	17.00	94%	100%
10	Standards	3	3	0	3	0	0.04	0	0	0	0	24.00	24.00	100%	100%
11															
12	Total	31	31	0	30	1	0.71	0	0	0	0	96.00	97.00	101%	100%
13	By Percentage		100%	0%	97%	3%		0%	0%	0%	0%				

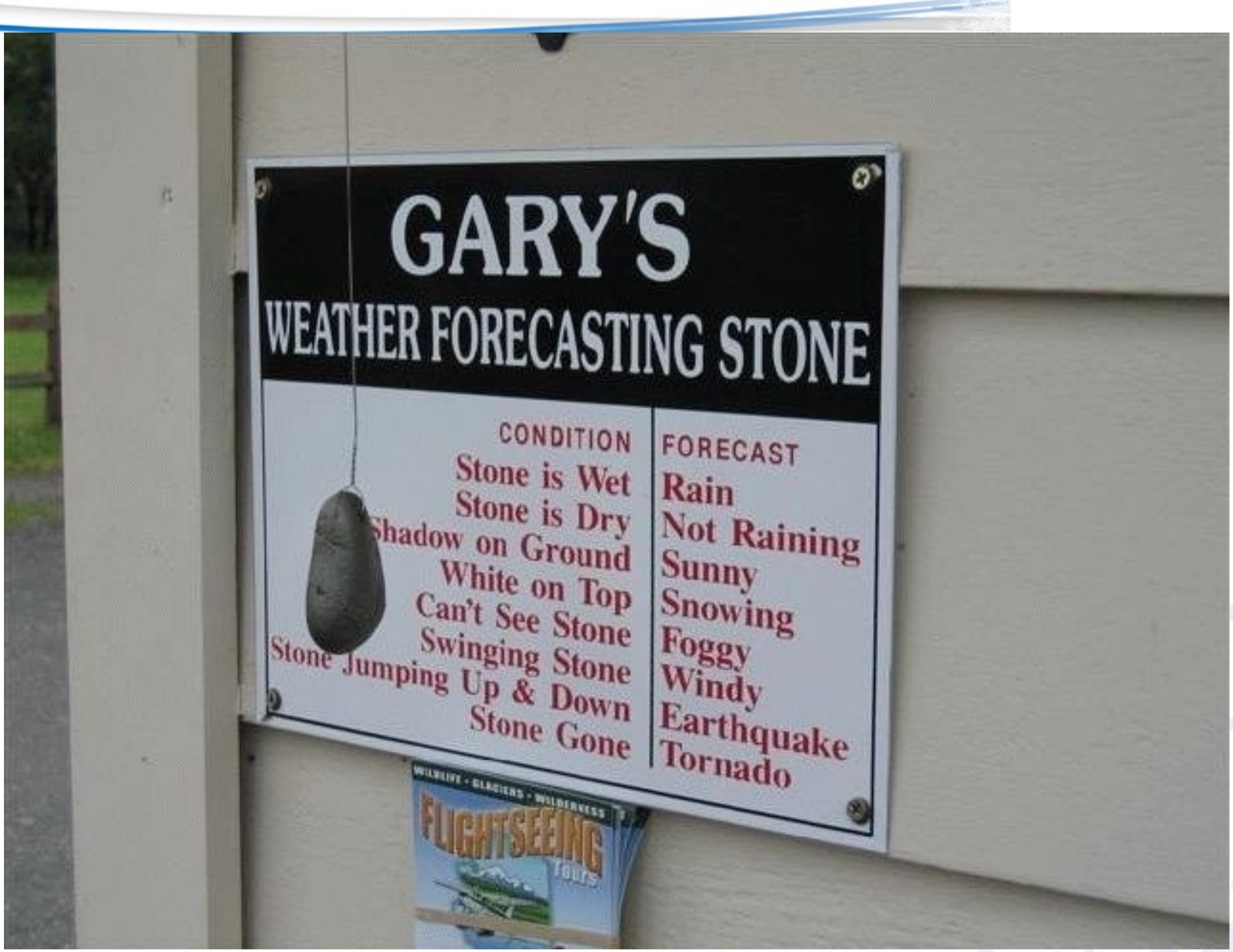
IEEE 829 Test Log

- A test log records the relevant details about test execution, and includes the following sections
  - Test log identifier
  - Description (items under test [with version numbers], test environment, etc.)
  - Activity and event entries (test-by-test, event-by-event information on the test execution process, the results of the tests, environmental changes or issues, bugs, incidents, or anomalies observed, testers involved, any suspension or blockage of testing, changes to the plan, impact of change, etc.)
- Tracking spreadsheets that capture these and many other details are often used for this purpose

Chapter 5:
Test Management

Section 3:
Amusement

Is This Helpful?



Recap1.5 Psychological Factors Summary

1. **Balancing Certainty & Progress**
2. **Avoiding Test Result Misinterpretation**
3. **Being Professional Pessimism**
4. **Balancing Curiosity**

Chapter 5: Test Management

Section 3: Exercise

Exercise: Reporting Bad News

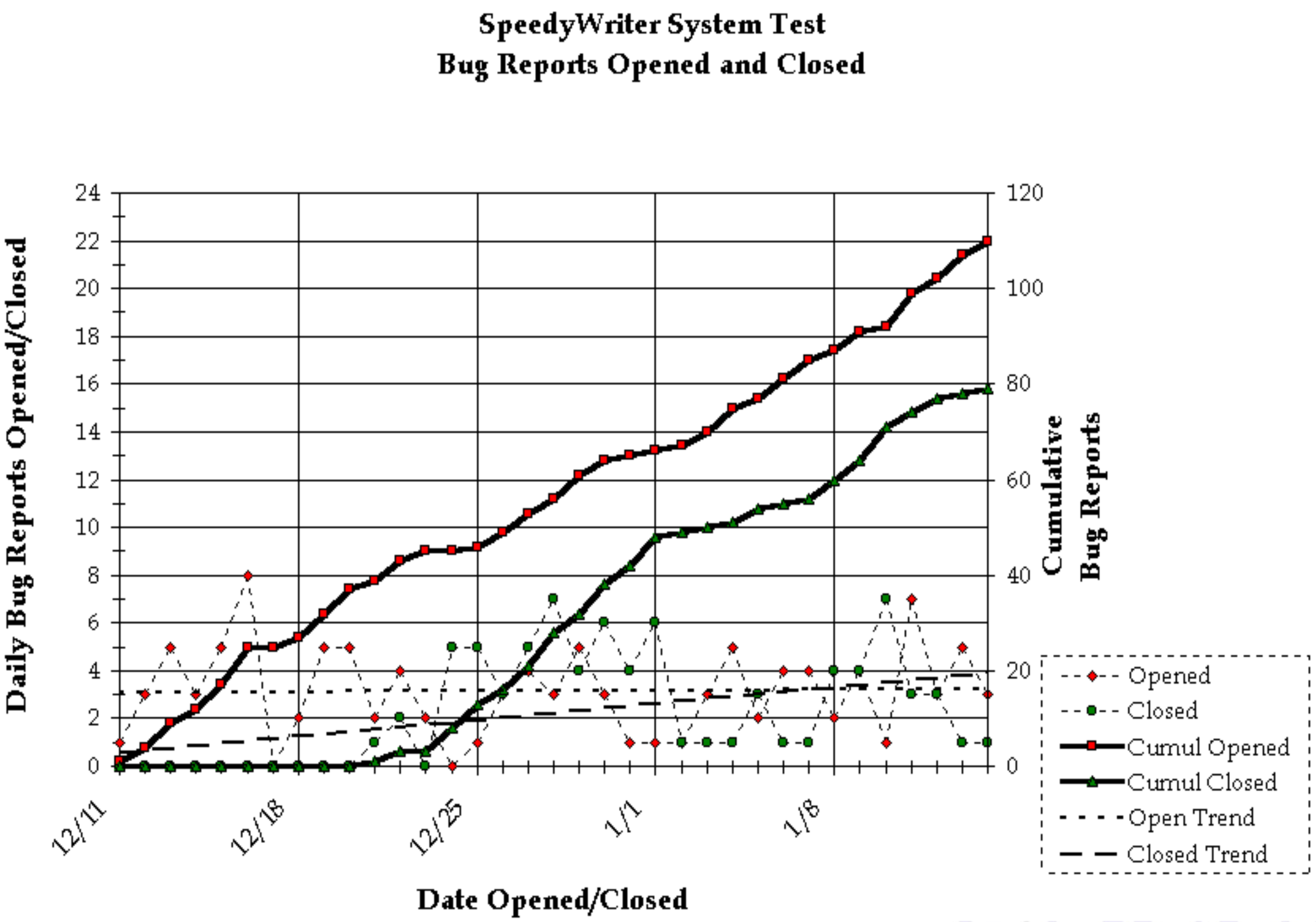
- On the following slides, you’ll see a text description following by charts illustrating a project in trouble.
- How would you present these test results to a project team?
- Discuss.

The Common Themes

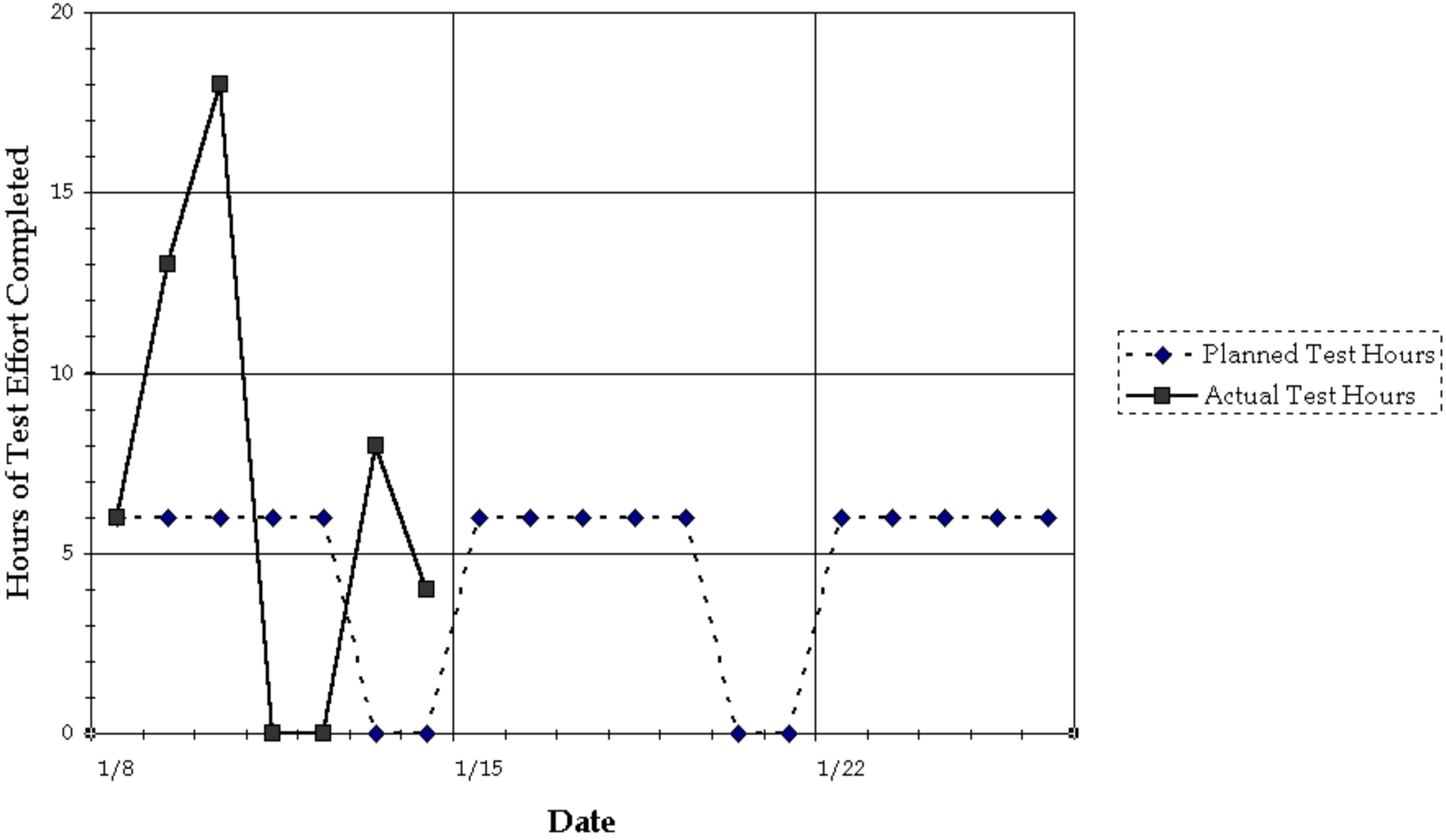
- The project is SpeedyWriter, a browser-hosted word processing application
- Development ran the Component and Integration Test phase
  - Component Test: December 11 through 29
  - Integration Test: December 18 through January 12
- The project is now in the System Test Phase
  - Cycle one: January 8 through 14
  - Today is January 15
  - The System Test Phase Exit Meeting is planned for January 29 (after two more one-week test cycles)

1: Too Many Bugs, Not Enough Time

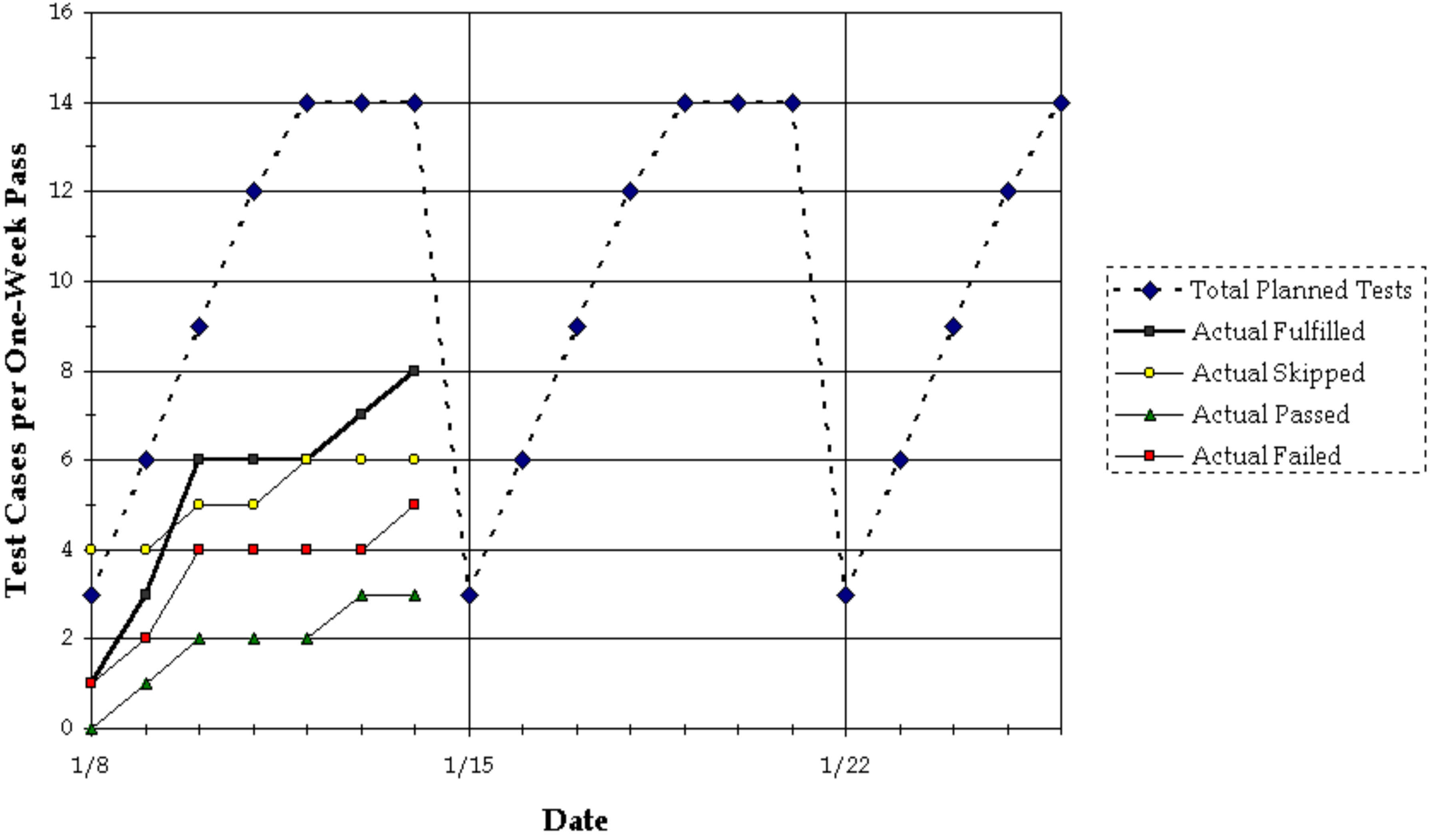
- The Test team is working productively and finding a few more bugs than expected
- Furthermore, a large backlog (about 30) exists
- Given the current bug find rate—which is not decreasing—and the backlog, the plan to finish testing in two weeks is in jeopardy
- The high bug find rate forced you to skip a large number of tests in cycle one


SpeedyWriter System Test
Bug Reports Opened and Closed
Daily Bug Reports Opened/Closed
24
22
20
18
16
14
12
10
8
6
4
2
0
120
100
80
60
40
20
0
Cumulative Bug Reports
12/11
12/18
12/25
1/1
1/8
Date Opened/Closed
Opened
Closed
Cumul Opened
Cumul Closed
Open Trend
Closed Trend

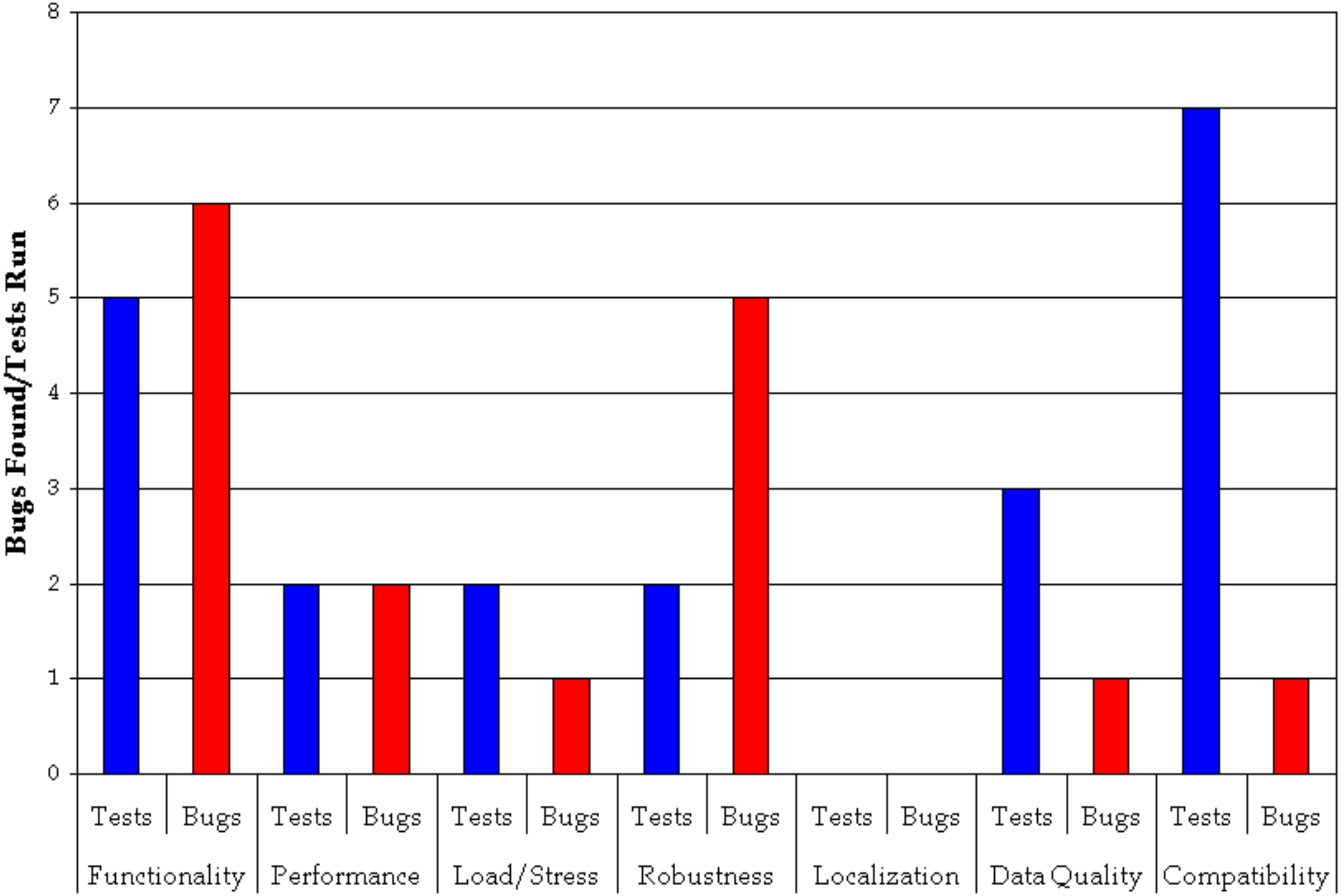
SpeedyWriter System Test
Test Hours Progress



SpeedyWriter System Test
Planned Test Fulfillment



SpeedyWriter Dashboard
Quality Risk Coverage Bugs/Tests



1: Too Many Bugs, Not Enough Time

- The Test team is working productively and finding a few more bugs than expected
- Furthermore, a large backlog (about 30) exists
- Given the current bug find rate—which is not decreasing—and the backlog, the plan to finish testing in two weeks is in jeopardy
- The high bug find rate forced you to skip a large number of tests in cycle one

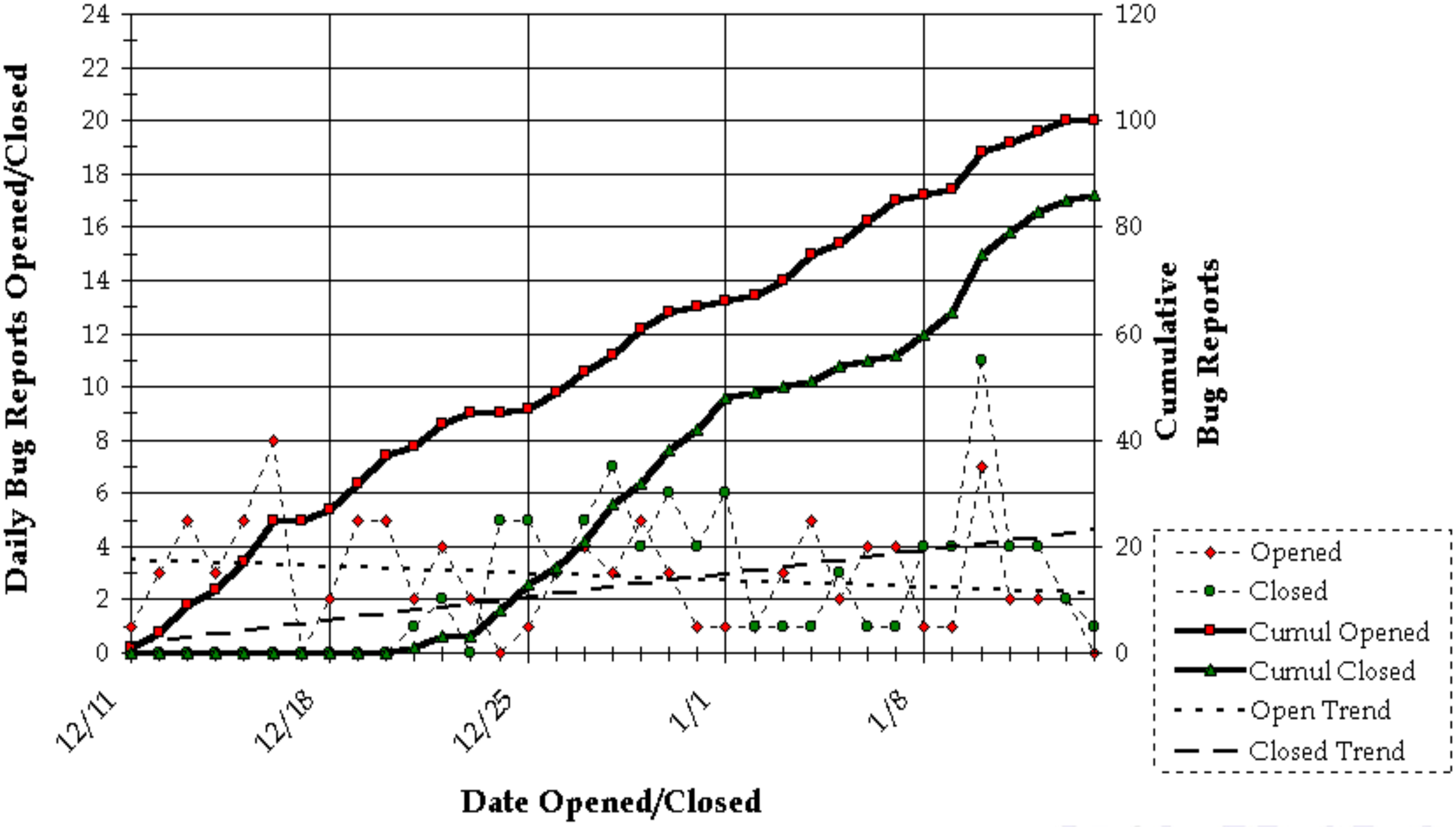
- **Threatened by**
  - The large back log
  - High rate of bug discovery
  - The most significant bugs?
  - Bug Priority (most low level)?

- **Possible Solution?**
  - *delaying the release date?*
  - *aggressively triaging the bugs and reprioritizing the tests (to focus on the most essential areas)*
  - *dropping problematic or as-yet-untested functionality.*

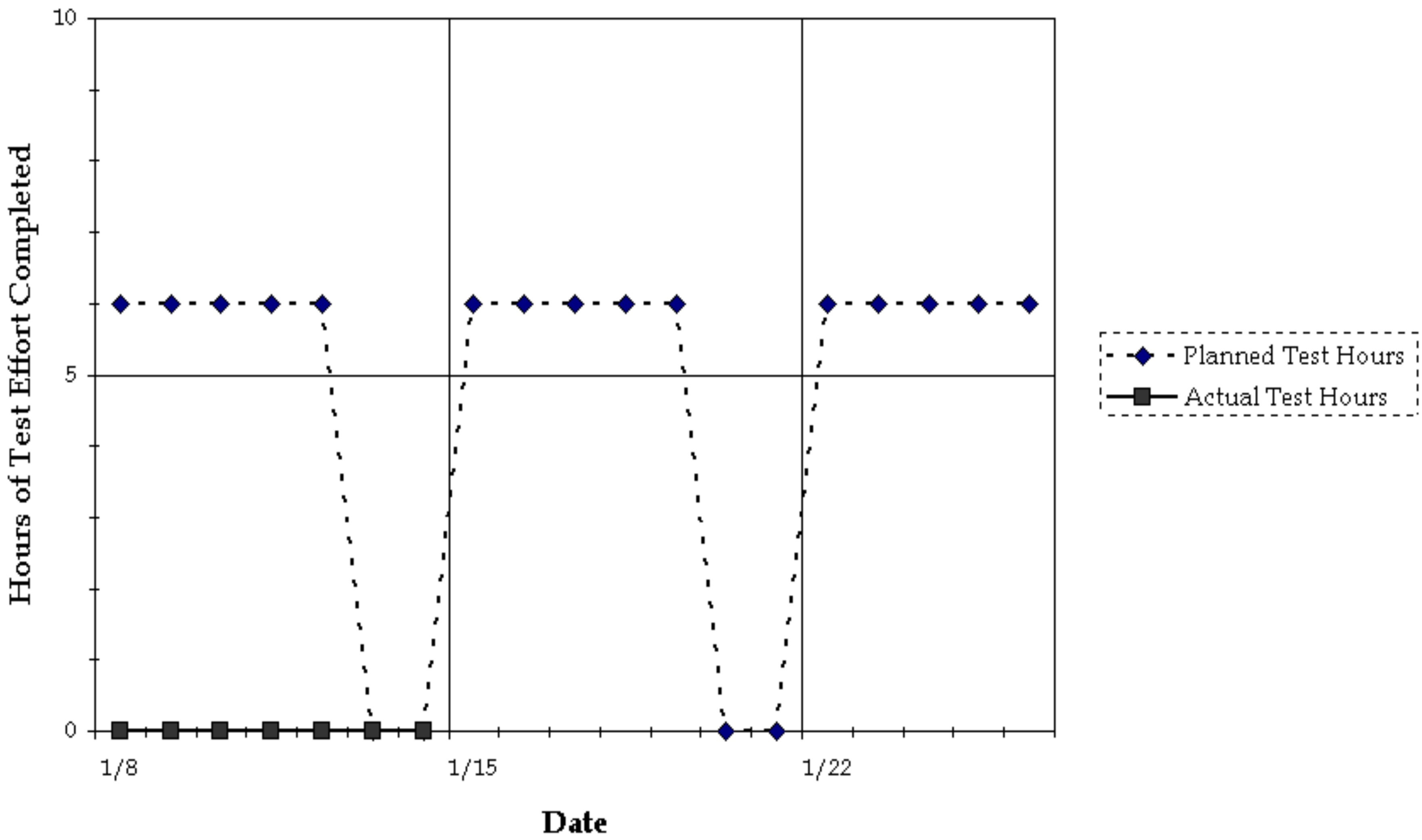
2: No Stable Test Environment

- The Test team is totally blocked
- The Operations and Development teams were unable to install a working build
  - Monday: Wrong software version on Web server
  - Tuesday: Operating system reinstall required on Web server
  - Wednesday: Five uninstallable builds delivered
  - Thursday: Clients couldn’t see server on net
  - Friday: Application crashed server repeatedly on start
  - Saturday and Sunday: Browser died (GPF) repeatedly. Pages and cell phone calls not returned
- Testing would start but then stop as the “onion peeling” exercises occurred

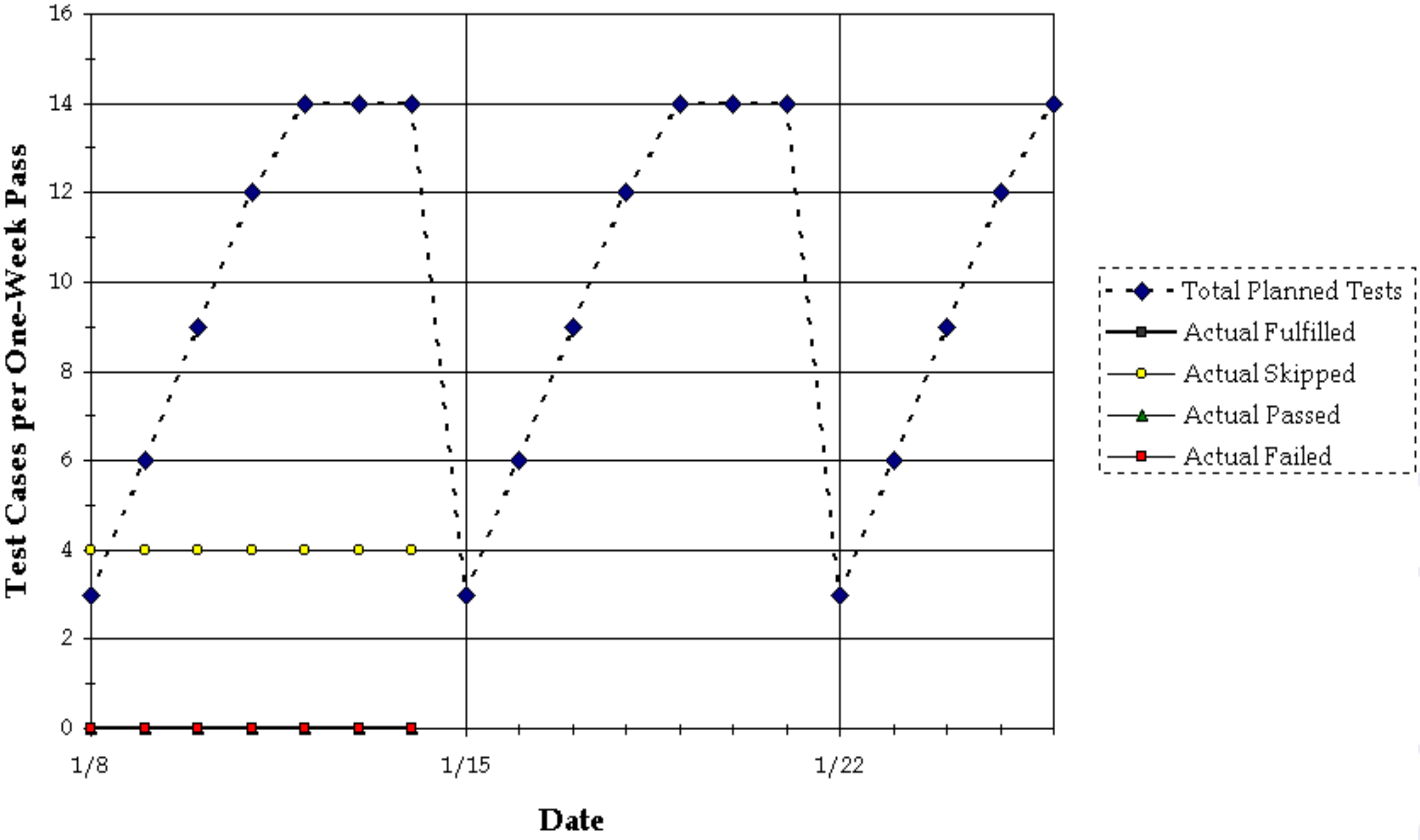
SpeedyWriter System Test
Bug Reports Opened and Closed

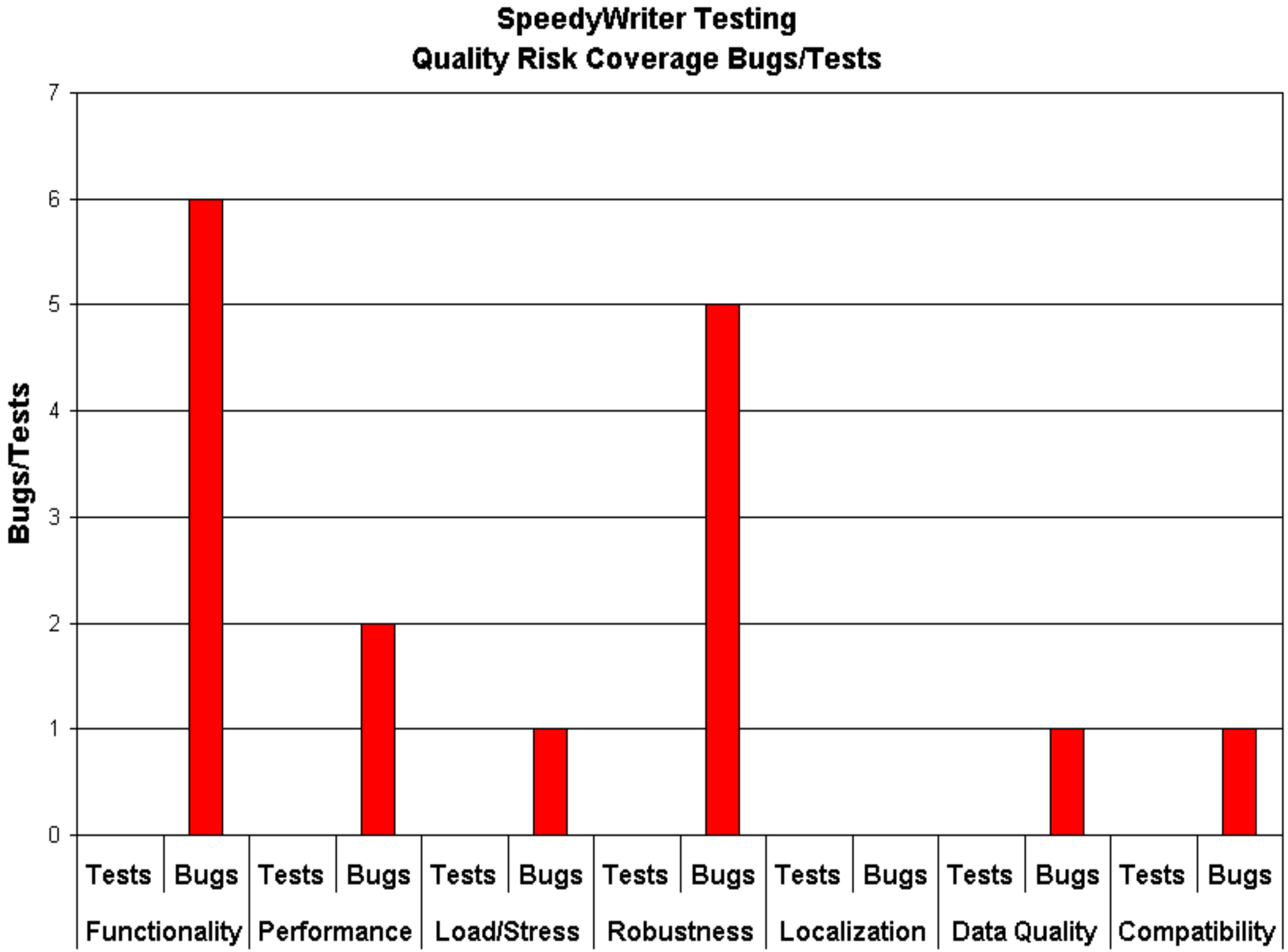


SpeedyWriter System Test
Test Hours Progress



SpeedyWriter System Test
Planned Test Fulfillment




SpeedyWriter Testing
Quality Risk Coverage Bugs/Tests
Bugs/Tests
7
6
5
4
3
2
1
0
Tests
Bugs
Tests
Bugs
Tests
Bugs
Tests
Bugs
Tests
Bugs
Tests
Bugs
Tests
Bugs
Functionality
Performance
Load/Stress
Robustness
Localization
Data Quality
Compatibility

2: No Stable Test Environment

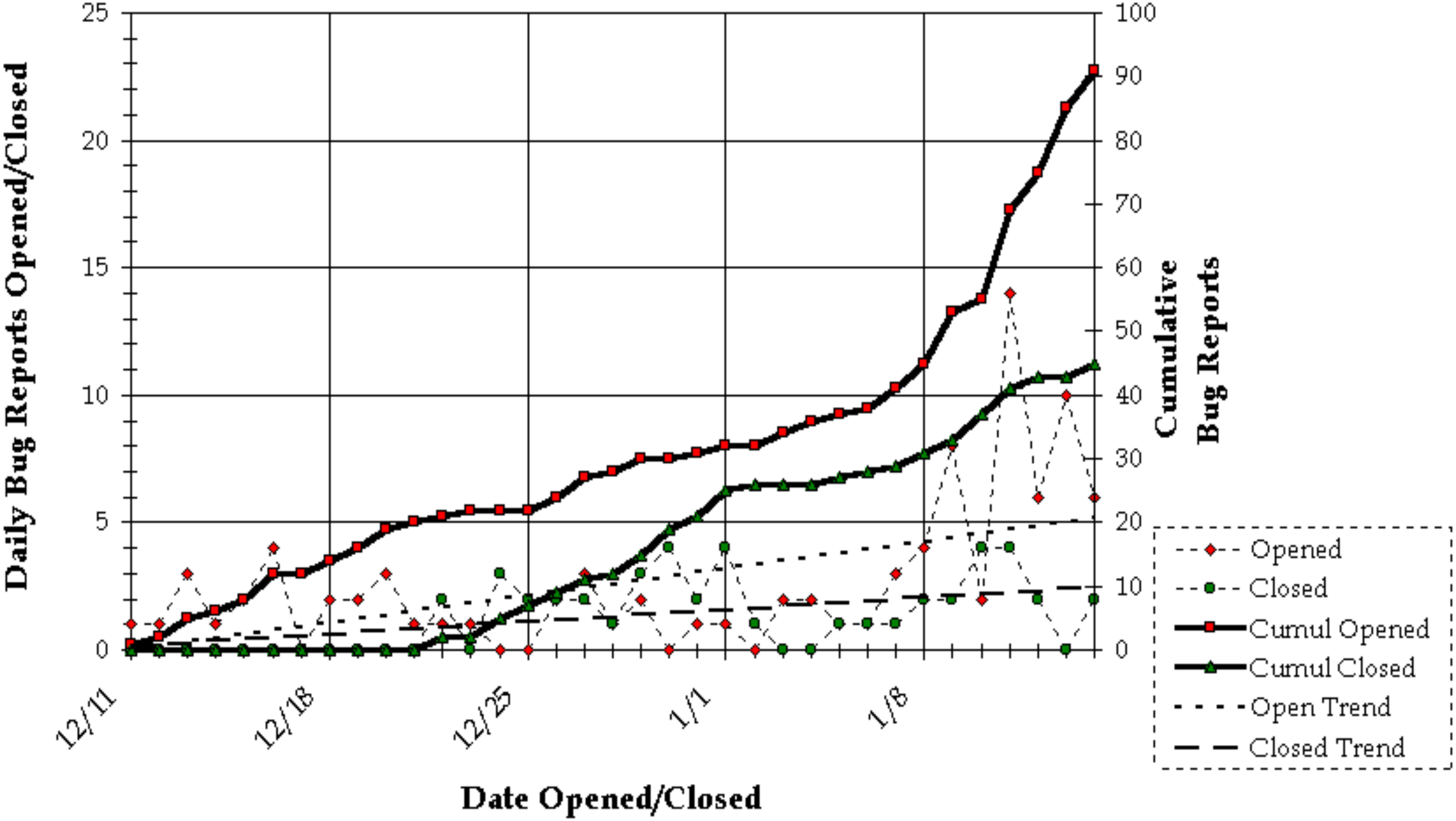
The Test team is totally blocked
The Operations and Development teams were unable to install a working build
***Monday**: Wrong software version on Web server*
***Tuesday**: Operating system reinstall required on Web serverWednesday: Five uninstallable builds delivered*
***Thursday**: Clients couldn't see server on net*
***Friday**: Application crashed server repeatedly on start*
***Saturday and Sunday**: Browser died (GPF) repeatedly. Pages and cell phone calls not returned*
Testing would start but then stop as the "onion peeling" exercises occurred

- **Do not to let such a situation fester for an entire week.**
  - Escalate appropriately but quickly.
  - Get urgent help from development and system administration staff by the middle of the day on the first Monday of testing, and make sure senior management knew of the situation if it persisted as of the end of the day.

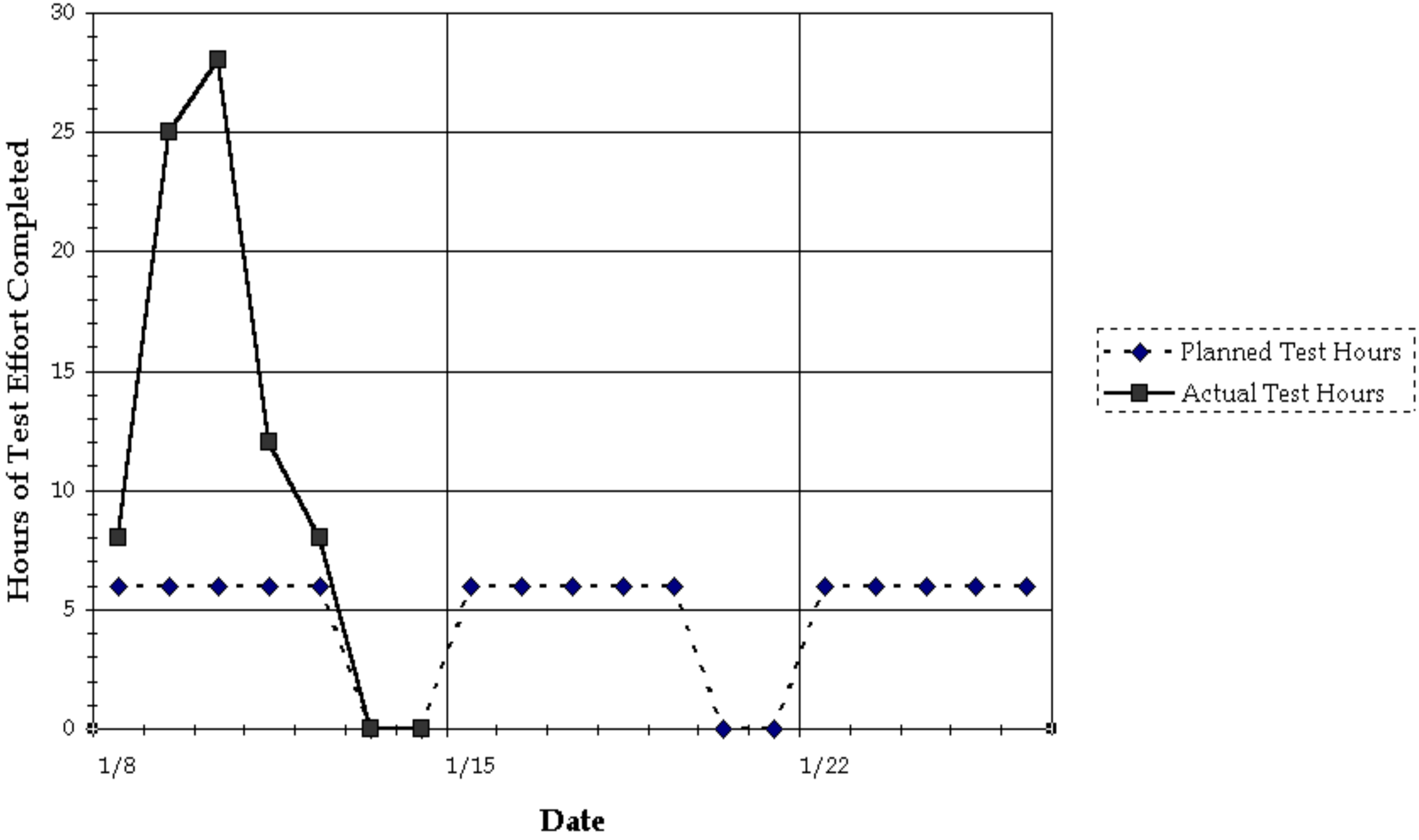
3: Inadequate Component/Integration Testing

- The Test team found many bugs in cycle one
- Extra testers added to meet plan
- Problems are myriad and even in basic areas that simple testing would have caught
  - E.g., can’t save a document longer than one page
  - The number and nature of problems blocks many more sophisticated tests
- The Development team has been warning Test informally that no time was in the schedule for sufficient component and integration testing

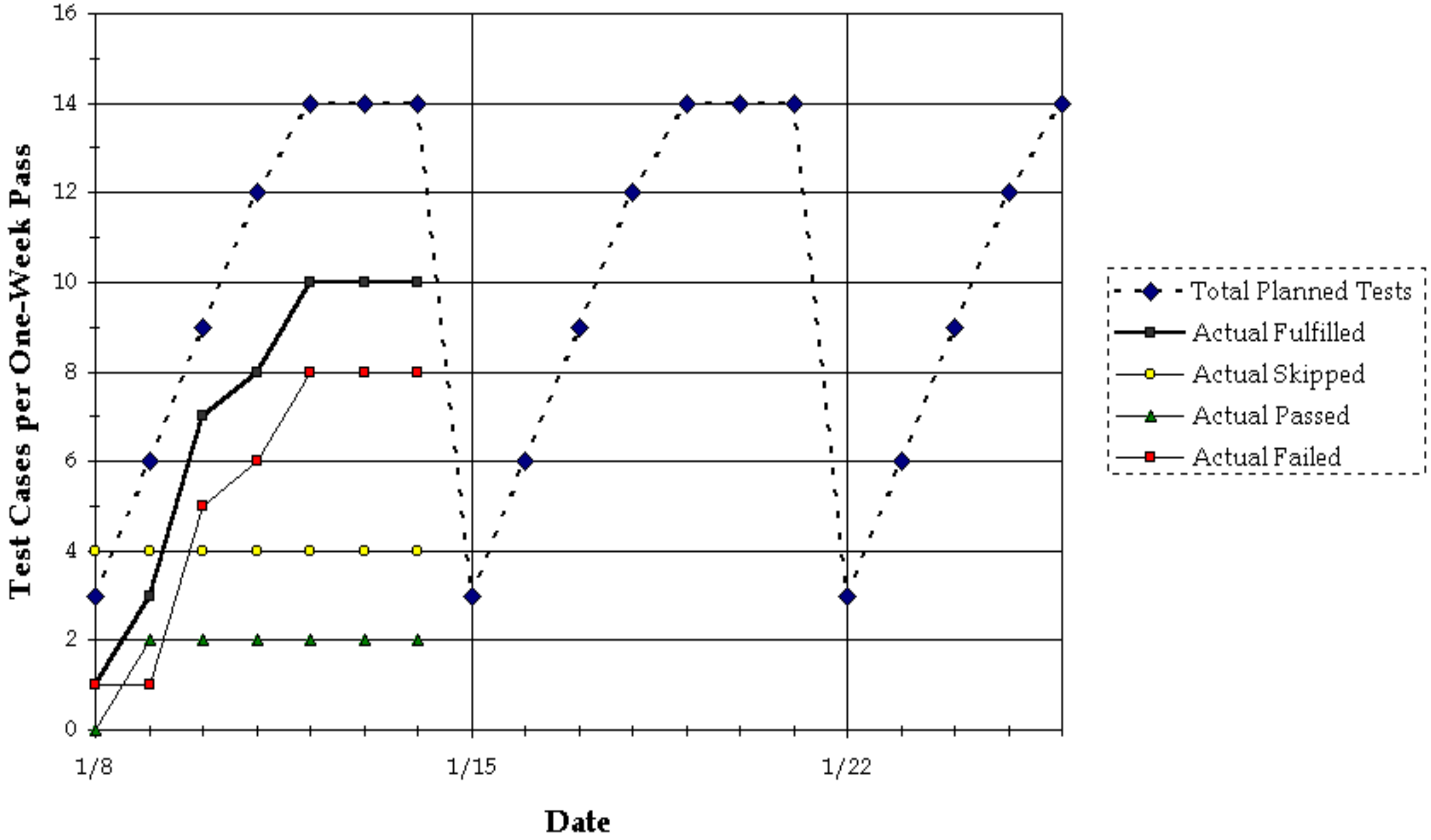
SpeedyWriter System Test
Bug Reports Opened and Closed



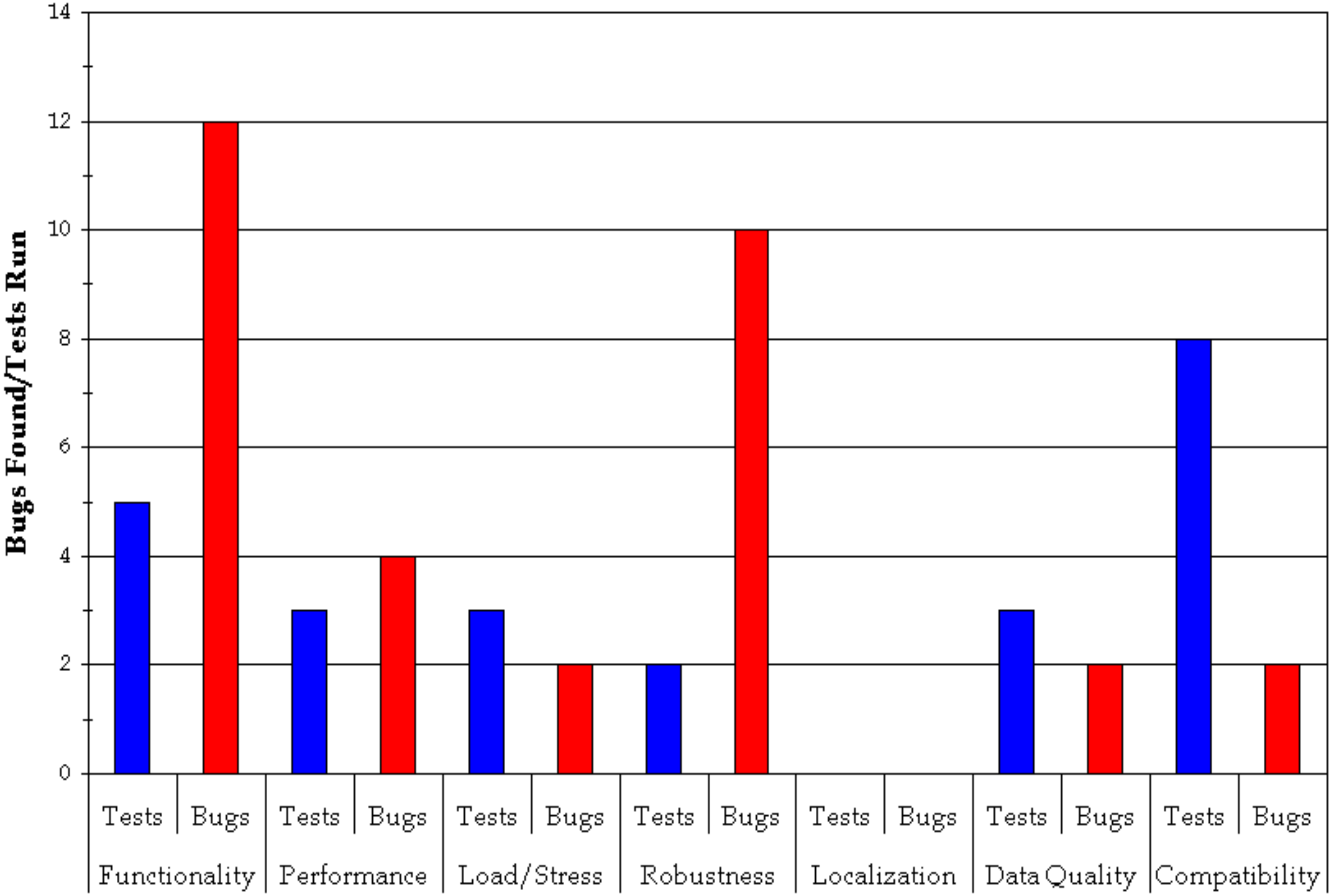
SpeedyWriter System Test
Test Hours Progress



SpeedyWriter System Test
Planned Test Fulfillment



SpeedyWriter Testing
Quality Risk Coverage Bugs/Tests



3: Inadequate Component/Integration Testing

- The Test team found many bugs in cycle one
- Extra testers added to meet plan
- Problems are myriad and even in basic areas that simple testing would have caught
  - E.g., can't save a document longer than one page
  - The number and nature of problems blocks many more sophisticated tests
- The Development team has been warning Test informally that no time was in the schedule for sufficient component and integration testing

- **Threatened by**
- Handling the political issues associated with the inadequate component and integration testing and the hints you've been given about it.

- **Possible Solution?**
- Making a bad complain or going to the development manager early on in the week?
  - putting together an action plan for recovery,
  - and announcing that plan jointly in the project meeting would be the most politically astute plan,

Chapter 5:
Test Management

Section 4:
Configuration management

Configuration Management

- Key concept
  - – How configuration management supports testing
- Terms to remember

Testing and Configuration Management

- Configuration management establishes, maintains the integrity of the items that make up the software or system through the project and product life cycle
- For testing, configuration management:
  - Allows management of testware and results
  - Ensures every item in test can be related back to known system components
  - Supports delivery of a test release into the test lab
- During project and test planning, the configuration management procedures and infrastructure (tools) should be chosen, documented, and implemented, so that no surprises occur at test execution time

Example-

you probably use Linux, Windows, or MacOS on your computer.

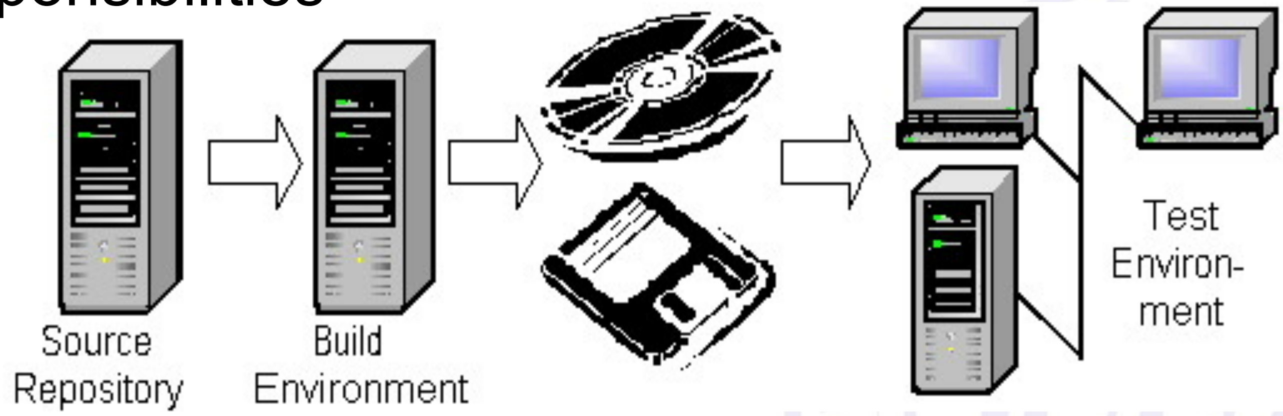
- OS->a family of systems->hundreds individual packages->
- each **package** consists of dozens of ancillary applications like browsers, media players, simple editing programs, networking programs, system management tools, and the like->
- each of which in turn consists of ten or more files. ->
- each of the **files** in all of these programs, including the hundreds of files in the operating system, must be built from dozens or hundreds or even thousands of files. ->
- each of the **source files** used to build the programs and the operating system contains five, ten, or even a hundred discrete **units of code**, **functions or objects depending on the programming language.** ->
- each of those discrete units **is potentially individually modifiable in the course of a project**. ->
- Each modification to one of those units creates **a distinct version of that unit.**
- **Only one version of any unit** may be included in a single test object. Configuration management, done properly, must manage that complexity.

Key Tasks of Configuration Management

- Store and control access to the items that make up the system (a.k.a., source code control, though it goes beyond code)
- Identify and document the items under control
- Allow change to control items through an orderly process (e.g., change control boards)
- Report on changes pending, underway, and complete
- Verify completeness of implementation

Test Release Management

- Release schedule (i.e., weekly? daily? hourly?)
- Update apply (process to install new build)
- Update unapply (process to remove bad build)
- Build naming (revision level); e.g., X.01.017
- Interrogation (process to determine rev. level)
- Synchronizing with databases, other systems, etc.
- Roles and responsibilities for each step



IEEE 829 Test Item Transmittal Report

- A test item transmittal report describes the items being delivered for testing, and includes the following sections
  - Test item transmittal report identifier
  - Transmitted items (include item names and revision numbers)
  - Location (where, what media, labeling, etc.)
  - Status (bugs fixed, changes introduced, etc.)
  - Approvals
- More commonly used are release notes, which include some of this information, usually informally

Chapter 5:
Test Management

Section 4:
Exercise

Exercise: Omninet Configuration Mgmt

- Identify the major items that you think would require configuration management for Omninet.
- Which of these items are important for test release management?
- Discuss.

Chapter 5: Test Management

Section 5: Risk and testing

Risk and Testing

- Key concepts
  - – Hazards and potential risks
  - – Risk as a possible problem threatening the achievement of stakeholders’ objectives
  - – Project as opposed to product risks
- Terms to remember

Project Risks

- A previous chapter introduced the idea of testing as a way of managing product or quality risks
- Testing is also *subject to* risk
- A risk is the possibility of a negative outcome, and that would include events like late test releases, environment problems, etc.
- To discover risks to the testing effort, ask yourself and other stakeholders:
  - What could go wrong on the project that would delay or invalidate your test plan and/or estimate?
  - What kind of unacceptable testing outcomes do you worry about?

Handling Project Risks

- For each project risk, you have four options:
  - Mitigation: Reduce the likelihood or impact through preventive steps
  - Contingency: Have a plan in place to reduce the impact
  - Transfer: Get some other party to accept the consequences
  - Ignore: Do nothing about it (best if both likelihood and impact are low!)
- Another typical risk-management option, buying insurance, is usually not available

Sample Risks, Mitigations and Contingencies

- Logistics or product quality problems block tests
  - ✓ Careful planning, good bug triage, robust test design
- Test deliverables won’t install
  - ✓ Smoke testing, nightly builds, uninstall process
- Excessive change invalidates results, requires test updates
  - ✓ Good change control processes, robust test design, limited test documentation, escalation to management
- Insufficient or unrealistic test environment(s)
  - ✓ Explain risks to management, outsource performance testing
- Test environment support unreliable
  - ✓ Good escalation process, sys. admin. skill in team
- Gaps in test coverage revealed during test execution
  - ✓ Exploratory testing, continuous test improvement
- Slips in test start dates and/or deliverables to test
  - ✓ Risk priority to drop tests, escalation to management
- Budget and/or staffing cuts
  - ✓ Risk priority to drop tests, well-rounded team, outsource testing
- Debugging in the test environment
  - ✓ Escalation to management, risk priority to drop/reschedule tests

Other Project Risks to Consider

- Organizational factors
  - Skill and staff shortages
  - Personal and training issues
  - Communication problems
  - Lack of follow up on test results, including failure to use such results for process improvement
  - Improper attitude toward or expectations of testing, whether by the testers or by others on the project
  - Complex organization (e.g., distributed)
- Supplier issues
  - Failure of a third party, including a test tool vendor
  - Contractual issues
- Technical issues
  - Problems in defining the right requirements
  - Unachievable requirements (leading to blocked/ unrunnable tests)
  - The quality of the design, code, tests and test data
  - Highly complex system

Chapter 5:
Test Management

Section 5:
Exercise

Exercise: Omninet Testing Risks

- Identify three to five project risks for Omninet testing.
- Assess the likelihood and impact of each risk on a three-point scale (High, Medium, or Low).
- Out of your four options, pick one or two appropriate steps to take to manage the risk.
- Discuss.

Chapter 5:
Test Management

Section 6:
Bug or incident management

Bug or Incident Management

- Key concepts
  - The content of a bug or incident report
  - Writing a bug or incident report
- Terms to remember

Goals and Audience

- Incident or bug reports often have the following goals:
  - To provide detailed information about the incident or bug to those who need it
  - To be part of the aggregate data to analyze (e.g., bug charts)
  - To lead to development and test process improvement
- Typical readers of bug reports include:
  - Developers: who need to fix the problem reported
  - Managers: who need to make resource-allocation/ prioritization decisions about the problem
  - Technical support staff: who need to be aware of deferred/ unfixed issues at ship time
  - Testers: who need to know the current state of the system

Ten Steps to a Good Bug Report

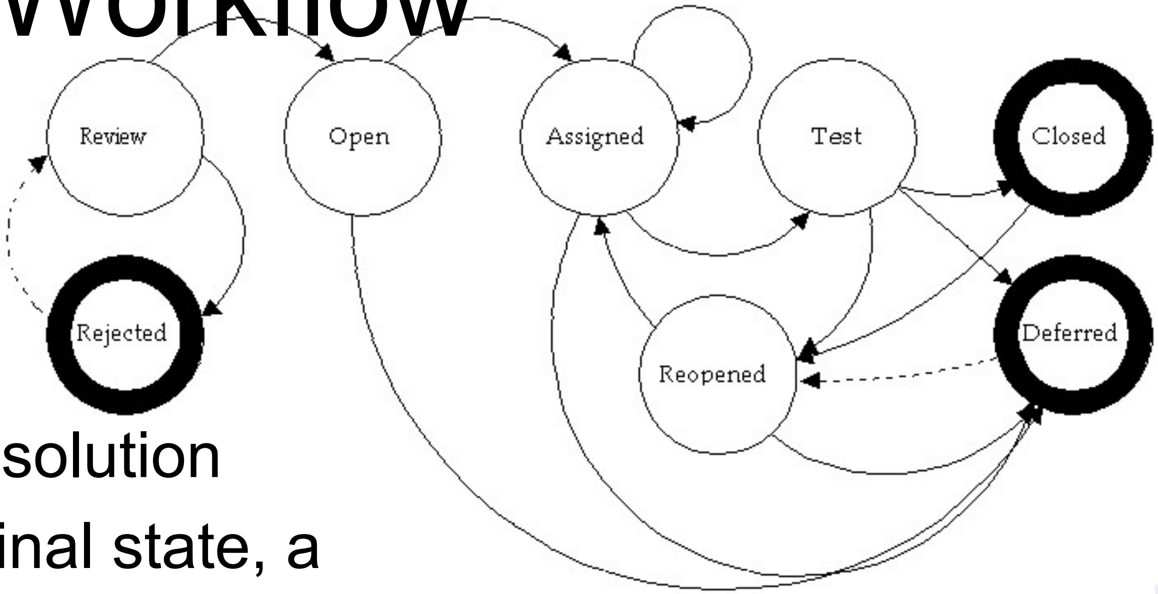
1. *Structure: test carefully*
2. *Reproduce: test it again*
3. *Isolate: test it differently*
4. *Generalize: test it elsewhere*
5. *Compare: review similar test results*
6. *Summarize: relate test to customers*
7. *Condense: trim unnecessary information*
8. *Disambiguate: use clear words*
9. *Neutralize: express problem impartially*
10. *Review: be sure*

Good bug reports on a problem can differ in style and content

IEEE 829 Test Incident Report

- A test incident (or bug) report describes a test event needing further investigation, especially a bug, and includes the following sections
  - Test incident report identifier
  - Summary (one line, especially impact on stakeholders)
  - Incident description (inputs, expected results, actual results, anomalies, date and time seen, environment, test in progress, reproducibility, reported by, and other details)
  - Impact (on testing, project, product, stakeholders, etc.)
- Strictly, incident reports describe any questionable behavior, while bug reports describe behavior due to bugs (failures) rather than bad tests or test data, tester errors, test environment problems, and the like

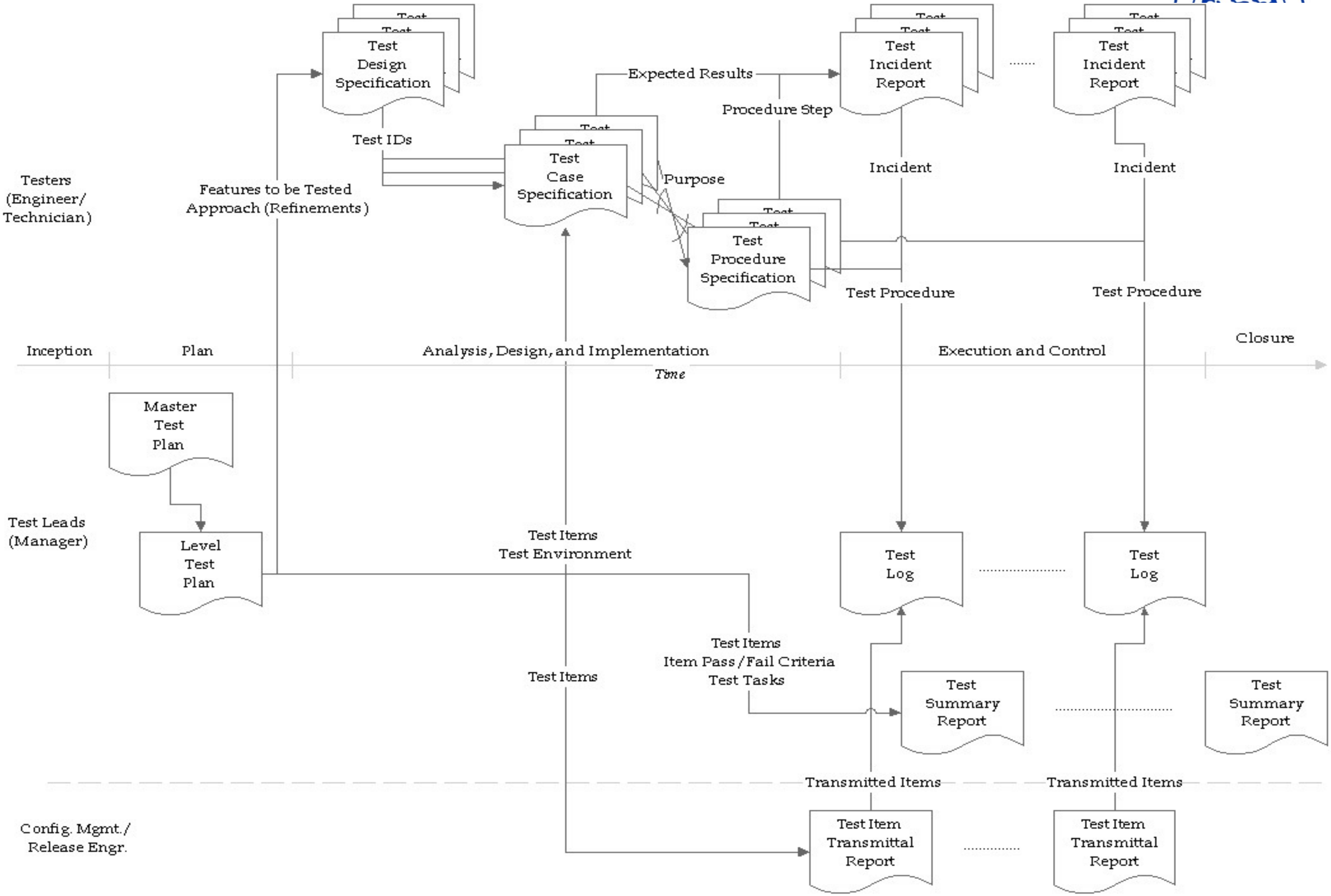
Bug Report Lifecycle or Workflow



- Bug reports move through a series of states (lifecycle or workflow) to resolution
- In each non-terminal state, a manager or the bug triage committee specifies an owner who is to move the bug to the next state
- Bug tracking systems can and should implement and automate these workflows, but project team and management support make the workflow work!

Other Information to Include

- The configuration of the software or system
- The phase of introduction, detection, and removal of the bug
- Urgency/priority to fix
- Conclusions and recommendations
- Risks, costs, opportunities, and benefits of fixing/not fixing
- Change history, especially for each state change
- Date of report, reporting organization, and author
- Expected and actual results
- Identification of the test item and environment
- Life cycle process in which the incident was observed
- Scope or degree of impact on stakeholder(s) interests
- Severity of the impact on the system



Testers (Engineer/ Technician)

Test Leads (Manager)

Config. Mgmt./ Release Engr.

Inception

Plan

Analysis, Design, and Implementation

Execution and Control

Closure

Time

Test Design Specification

Test Case Specification

Test Procedure Specification

Test Incident Report

Test Incident Report

Master Test Plan

Level Test Plan

Test Log

Test Log

Test Summary Report

Test Summary Report

Test Item Transmittal Report

Test Item Transmittal Report

Features to be Tested

Approach (Refinements)

Test IDs

Purpose

Expected Results

Procedure Step

Incident

Incident

Test Procedure

Test Procedure

Test Items

Test Environment

Test Items

Item Pass/Fail Criteria

Test Tasks

Test Items

Transmitted Items

Transmitted Items

Chapter 5:
Test Management

Section 6:
Exercise

Exercise: Omninet Bug Report

- The next page shows a poorly-written bug report for the Omninet project.
- Try to improve it, making some reasonable assumptions about the testing done. Try to cover all points addressed in the IEEE 829 incident report standard.
- You may assume the instructor is the tester who wrote the bug report and ask questions to help fill in any missing data.
- Discuss.

Summary: Session lasts too long.

Steps to Reproduce:

1. *Swipe credit card in payment subsystem.*
2. *Buy some blocks of time.*
3. *Start a stopwatch when surfing starts.*
4. *Session will last between one and five minutes too long.*

上海市嘉定区曹安公路4800号，同济大学嘉定校区软件学院