BIO
PRESENTATION

W8

Wednesday, May 17, 2006 1:45PM

S-CURVES AND THE ZERO BUG BOUNCE: PLOTTING THE WAY TO BETTER TESTING

Shaun Bradshaw
Questcon Technologies, A Division of Howard Systems Intl.

International Conference On
Software Testing Analysis and Review
May 15-19, 2006
Orlando, Florida USA

Shaun Bradshaw

Shaun Bradshaw began his career in IT in 1994 with Teleflex Information Systems, a subsidiary of Vanguard Cellular. While with Teleflex he worked as an Application Support Analyst and eventually moved into the Business Analysis group, providing IT support for internal groups such as the Cell Fraud department and Collections. In 1998 Shaun joined Questcon Technologies, an IT firm specializing in Quality Assurance and Software Testing consulting. As a Senior Consultant with Questcon, Shaun assisted dozens of clients in various industries with the establishment, tracking and analysis of test metrics programs. He has managed test efforts using the QuestAssured Metrics Methodology, utilizing both the S-Curve and Zero Bug Bounce as techniques for ensuring the timely completion of those test efforts and he has found that these two graphs are effective at illustrating the state of a test effort to the Project Team.

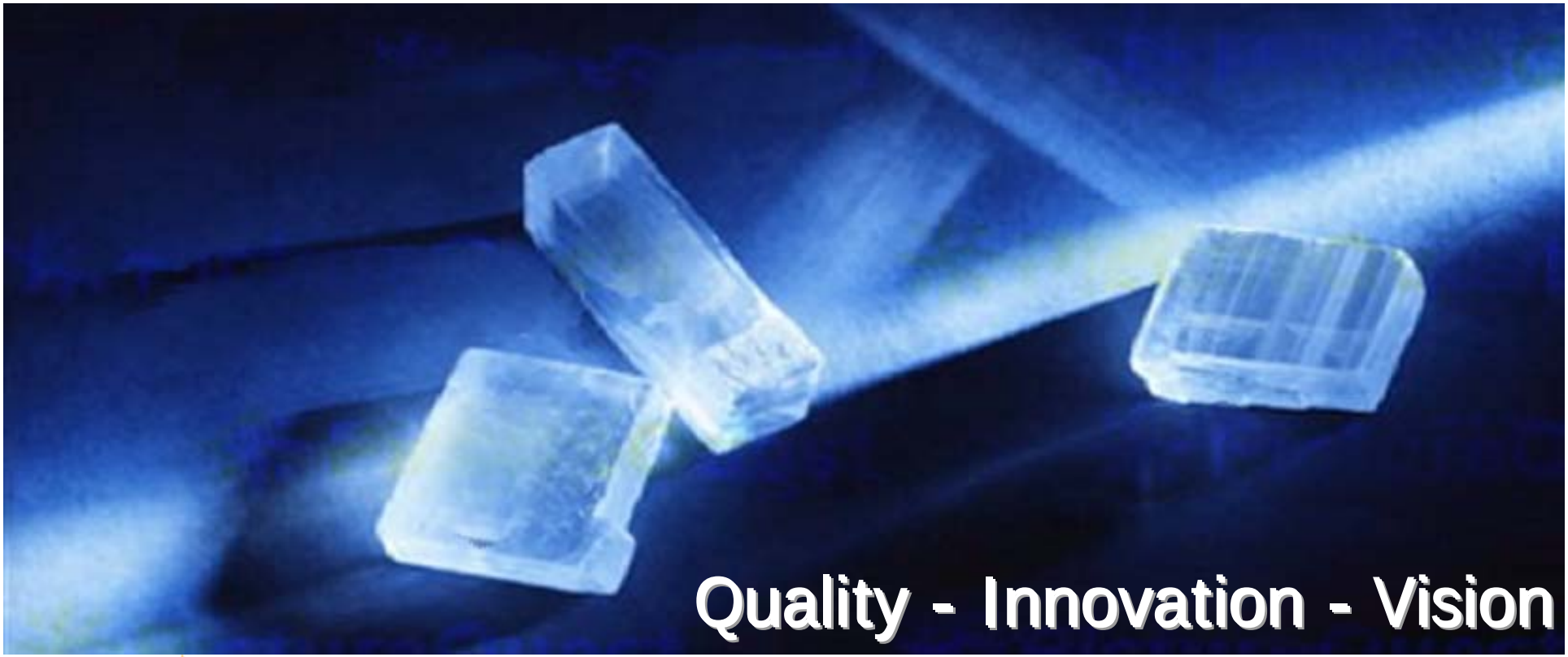
In 2002 he was promoted to the Director of Quality Solutions and now works with and manages the senior consulting staff to improve client Software Testing and QA processes. Shaun is the co-author and editor of the QuestAssured® Service Methodologies, as well as the primary creator of the methodology training classes offered by Questcon. He has been a featured speaker on test metrics and related topics at local and national QA and Testing conferences including the STAREAST 2004 Conference.

Shaun received his B.S. in Information Systems with a minor in Computer Science from the University of North Carolina at Greensboro.

Author Contact

Shaun Bradshaw
Director of Quality Solutions
Questcon Technologies, a division of Howard Systems International
1429 Westover Terrace
Suite A
Greensboro, NC 27408

Phone: 336-273-2428
Fax: 336-273-2413
Email: sbradshaw@questcon.com



S-Curves & the Zero Bug Bounce:

Plotting Your Way to More Effective Test Management

Presented By:
Shaun Bradshaw
Director of Quality Solutions
Questcon Technologies

May 17, 2006

Objectives

The primary objectives of this presentation are to instruct Test Leads & Managers on how to improve their ability to manage and track a test effort utilizing the S-Curve and Zero Bug Bounce, as well as communicate the results of a test effort to other members of the Project Team. To that end, the following concepts will be discussed:

- Test Management Using S-Curves
  - What is an S-Curve
  - Collecting Data
  - Analyzing the Graph
- Defect Management with the Zero Bug Bounce
  - Tracking Defects
  - What is the Zero Bug Bounce
  - Analyzing the Graph

The S-Curve

Test Management Using S-Curves

What is an S-Curve?

What makes it an "S" shape?

How is it used?

Successfully managing a test effort requires the ability to make objective and accurate estimates of the time and resources needed to stay on schedule. The S-Curve is one method for doing this.

What is an S-Curve?

What makes it an "S" shape?

How is it used?

- An S-Curve is a graphical representation of the cumulative work effort, or a subset of the work effort, of a software project.
  - S-Curves can be used to describe projects as a whole, development efforts, and test efforts, as well as defect discovery rates.
  - We will focus on how S-Curves are used to manage test execution and defect discovery rates.

Test Management Using S-Curves

What is an S-Curve?

What makes it an "S" shape?

How is it used?

- Test efforts typically start out slowly as test analysts run into a few major defects that prevent them from moving forward quickly
- As the initial issues are resolved, the test analysts are able to execute more tests covering a larger variety of functionality
- As the test effort nears its end, there are typically a few left over issues that must be resolved thus slowing the process down again

Test Management Using S-Curves

What is an S-Curve?

What makes it an "S" shape?

How is it used?

- Plot the progress of various test metrics to quickly see the effectiveness of the test effort
  - TCs Passed vs. Planned Execution Time
  - Total Failures vs. Planned Execution Time
- Measure test progress by comparing the actual test curve to a theoretical S-Curve
- Use the curve to determine if the application is stable enough to be released

The Theoretical S-Curve

The first step in utilizing an S-Curve for test management involves deriving a theoretical curve, that is, a uniformly distributed curve indicating “optimum” test progress.

- The theoretical S-curve is calculated as follows:

$$\frac{\text{(Day Number / Total Days in Test Effort)}}{\text{(Day Number / Total Days in Test Effort)} + e^{(3-8 * \text{Day Number / Total Days})}}$$

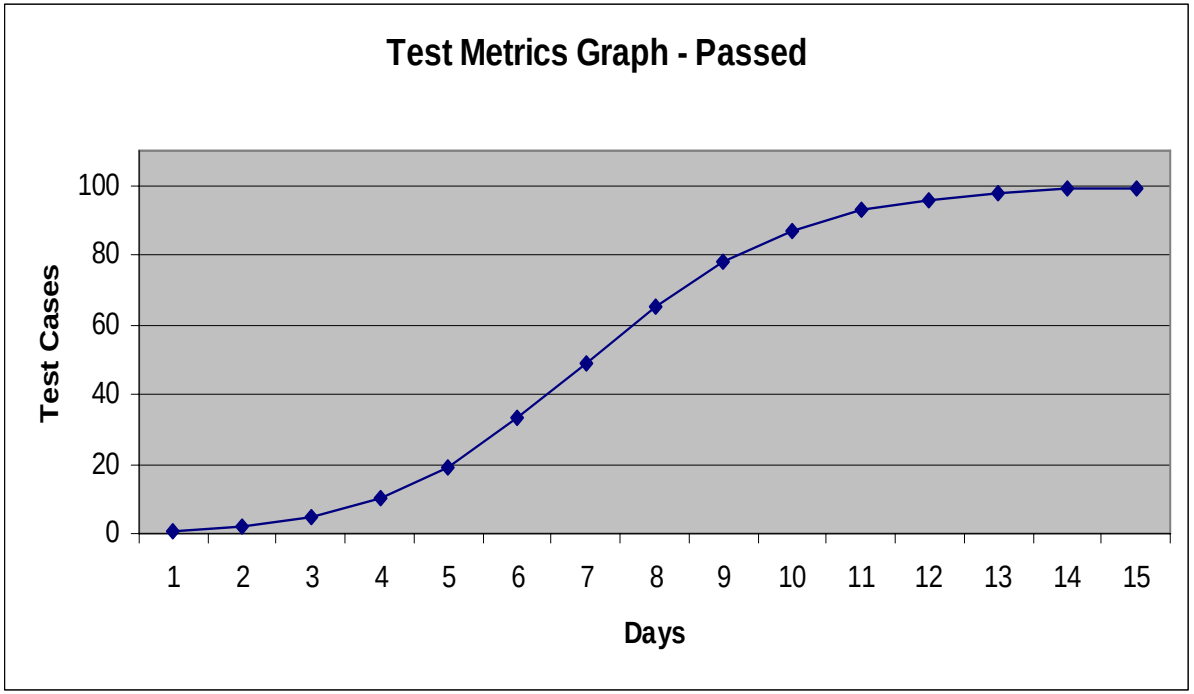
- Using this formula will return the cumulative percentage of tests passed or defects found (depending on the metric being tracked).

Note 1: “*e*” is the base of the natural logarithm (2.71828182845904)
Note 2: the “3” and “8” in the formula set the location of the logarithmic curves

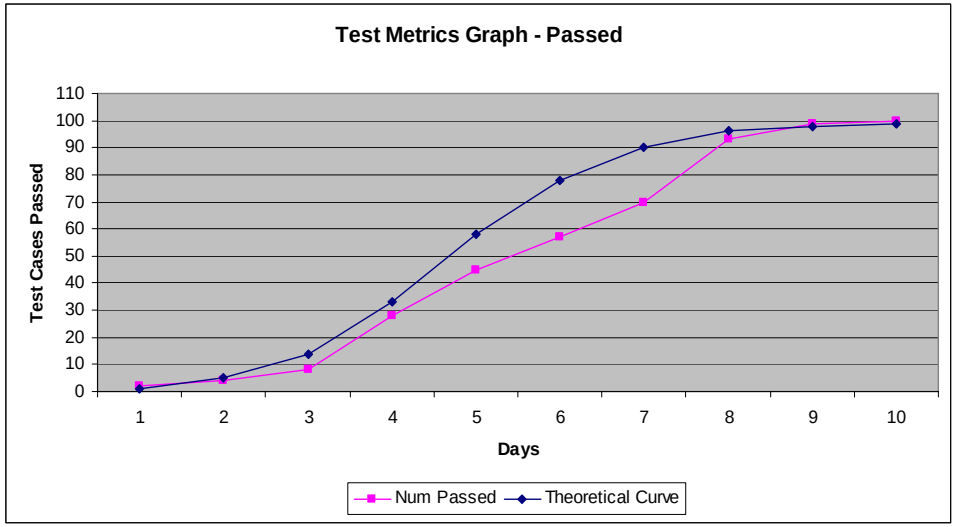
The Theoretical S-Curve

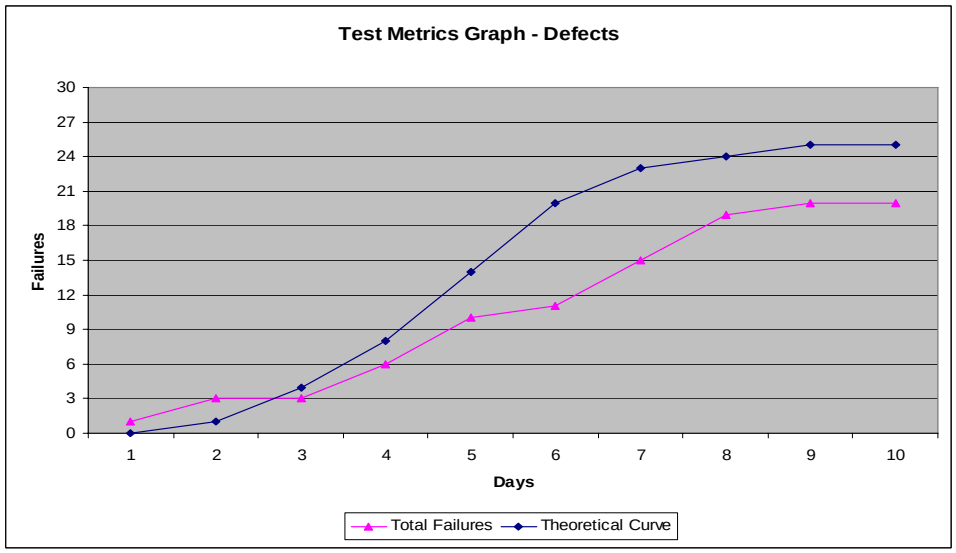
Here is an example of how a theoretical curve will look for a 15 day test effort with 100 test cases to be executed.

S-Curve Calculations - Passed		
# Days	# TCs	
15	100	
	Theoretical Curve	
1	0.56%	1
2	1.89%	2
3	4.70%	5
4	10.08%	10
5	19.28%	19
6	32.82%	33
7	49.28%	49
8	65.43%	65
9	78.40%	78
10	87.30%	87
11	92.80%	93
12	96.00%	96
13	97.79%	98
14	98.78%	99
15	99.33%	99



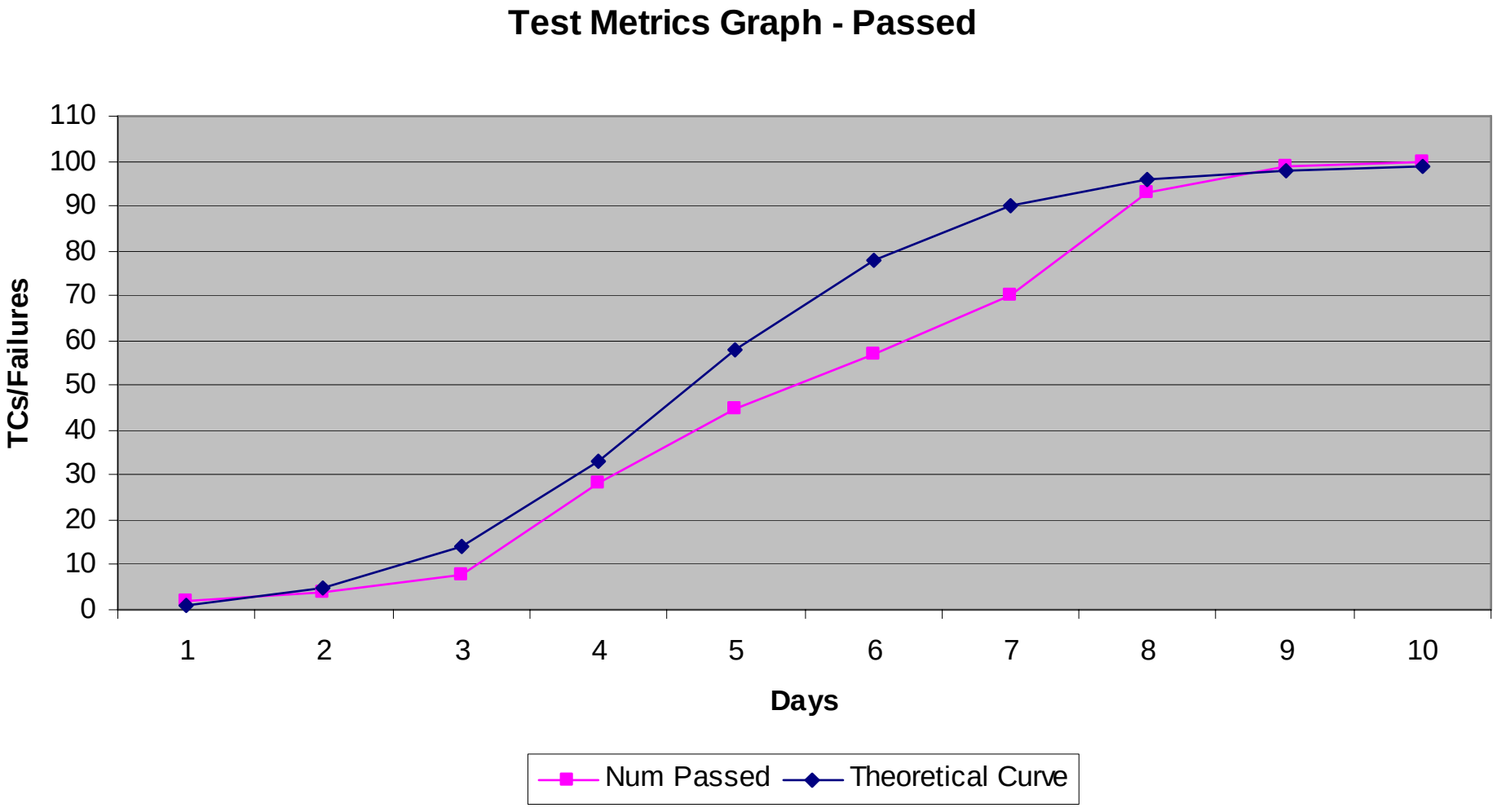
The Actual Test Curve

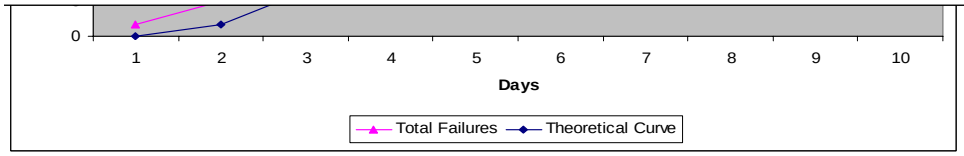




By plotting the actual cumulative number of test cases passed or the cumulative number of defects found during a test effort and comparing the resulting graph to the theoretical curve, we are able to quickly and objectively identify risks and/or issues in the test effort, which will be explained later.

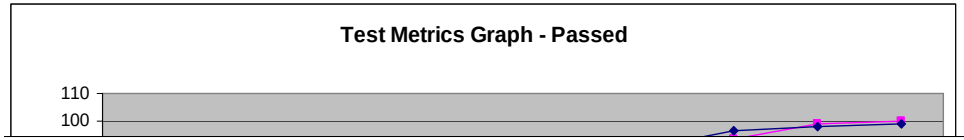
The Actual Test Curve

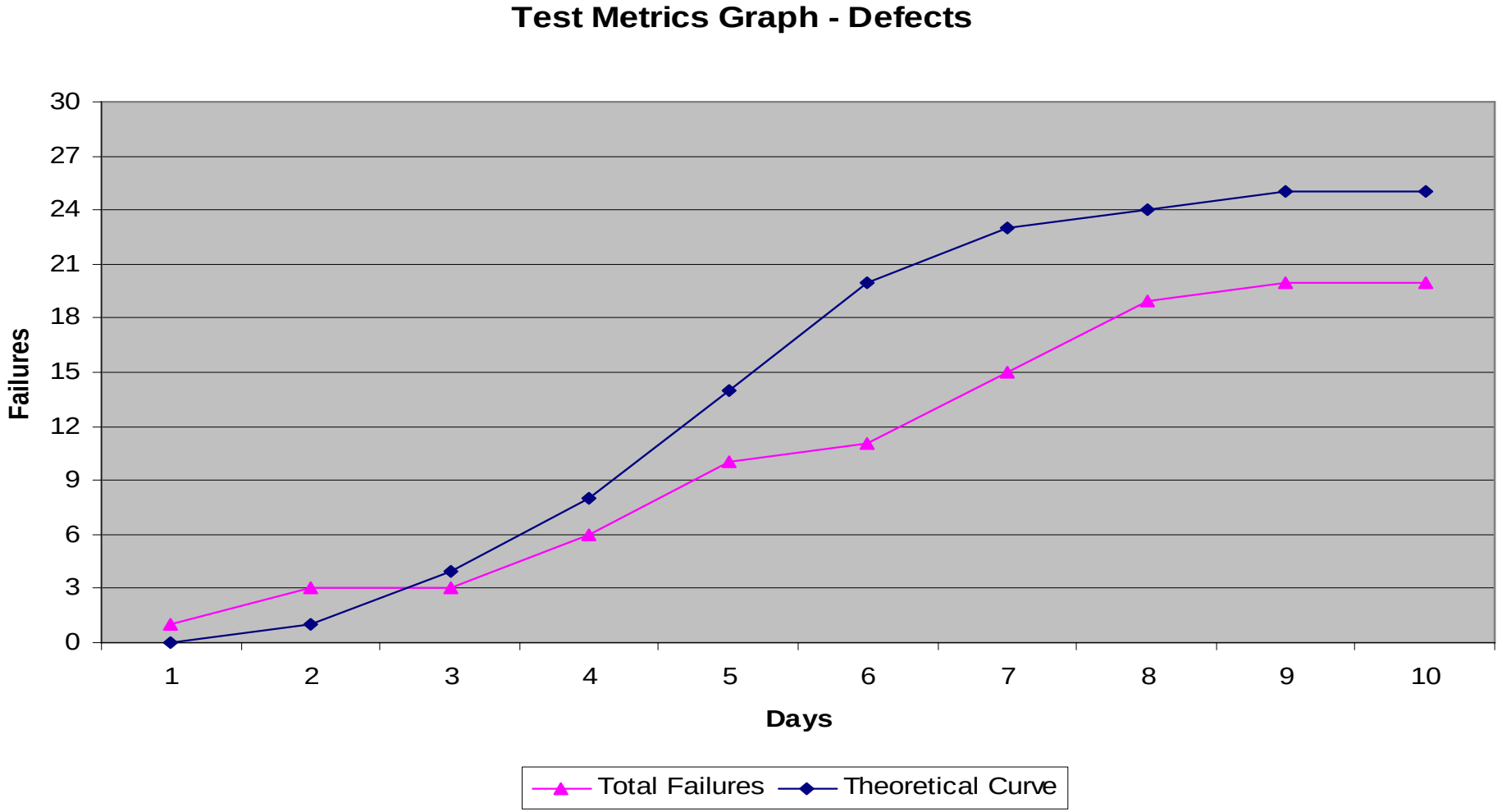




explained later.

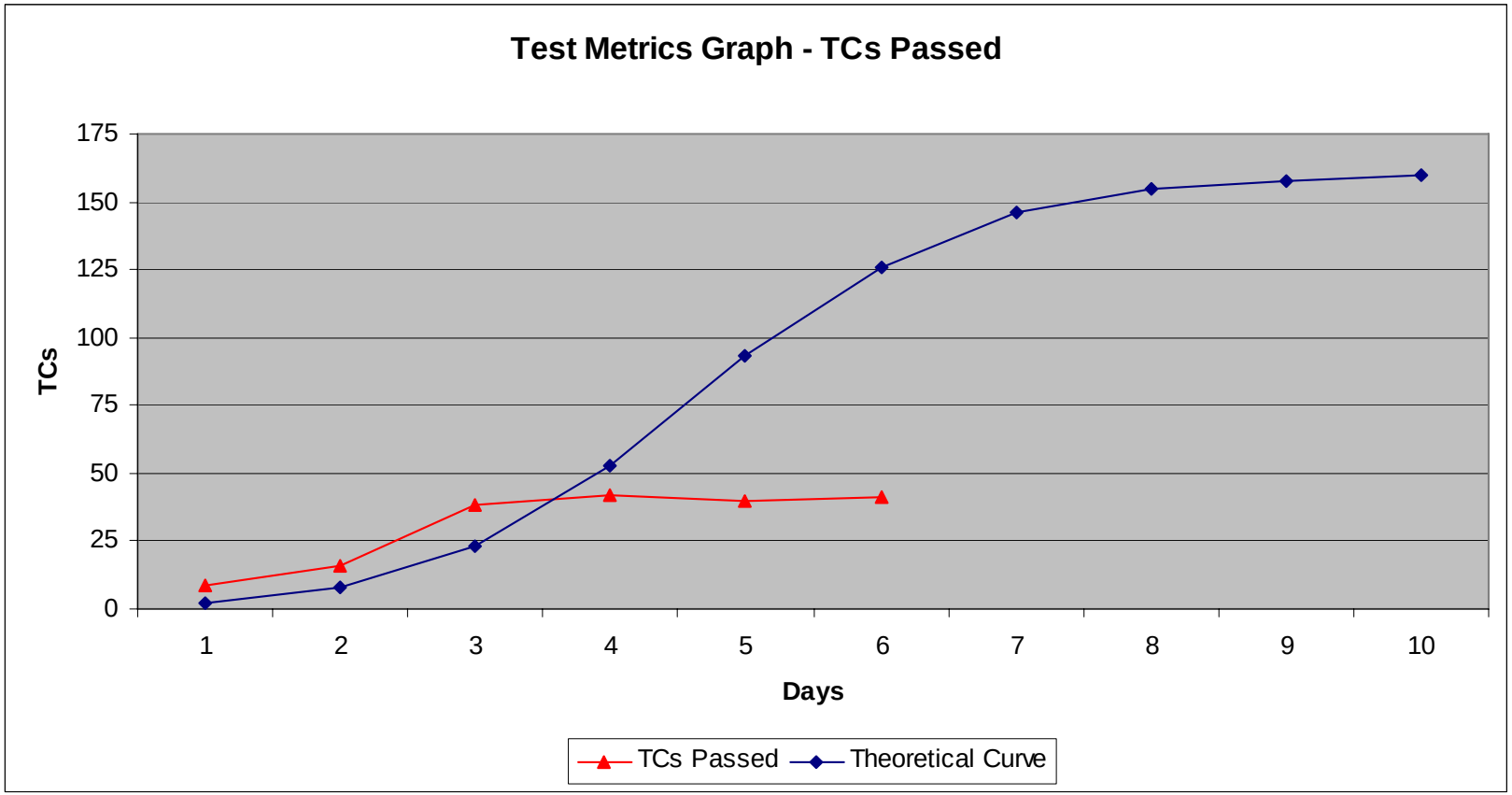
The Actual Test Curve





Analyzing S-Curves

What are some potential causes associated with this S-Curve? How might you correct these issues?



Analyzing S-Curves

Potential Causes

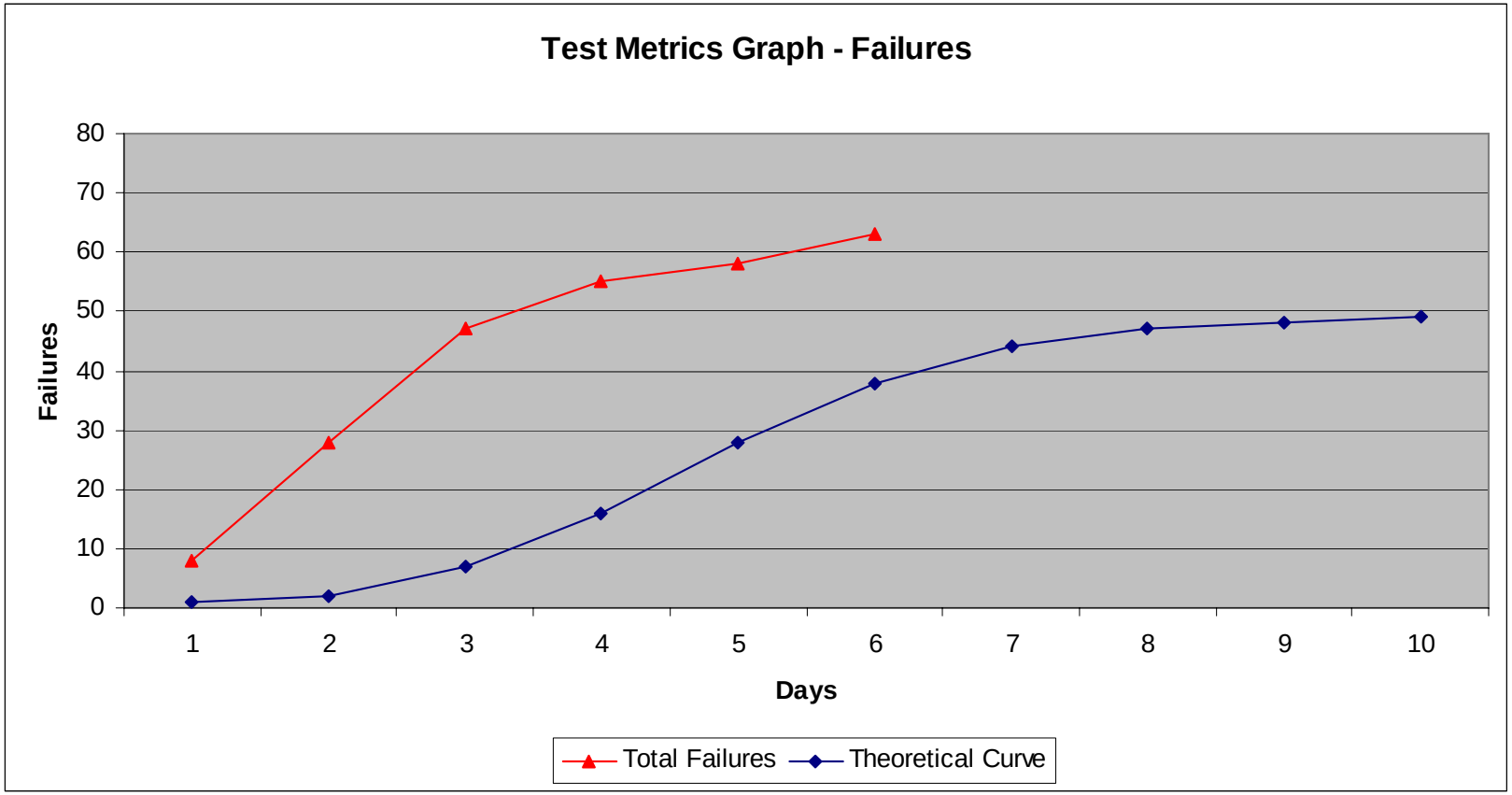
- Defects are causing significant numbers of test cases to be "blocked"
- Test resource re-allocation during the test effort

Corrective Actions

- Request an emergency fix from development team to correct the defect(s) causing tests to be blocked
- Request additional test resources
- Re-evaluate test case execution prioritization to ensure the most critical functionality can be tested prior to release

Analyzing the Graph

What are some potential causes associated with this S-curve? How might you correct these issues?



Analyzing S-Curves

Potential Causes

- Underestimated the number of defects in the release
- Development team releases “bug fixes” with defects still present

Corrective Actions

- Re-evaluate average defect rate related to this type of application or project
- Request the Development Lead to enforce unit testing and/or peer-code reviews before releasing fixes to test

The Zero Bug Bounce

Tracking Defects

Defect tracking is the process of monitoring what happens to a defect when it is found during the test effort.

Without proper control over this process, it can be difficult to ensure that all of the objectives of the test effort have been met and to determine when it is complete.

Tracking Defects

Defect tracking allows us to evaluate our ability to adhere to the schedule based on the number of defects discovered and the amount of time to correct them.

Through this process we can track:

- Which defects must be fixed,
- When defects are corrected, and
- When the system is ready for production.

Defect Management with the Zero Bug Bounce

What is the Zero Bug Bounce?

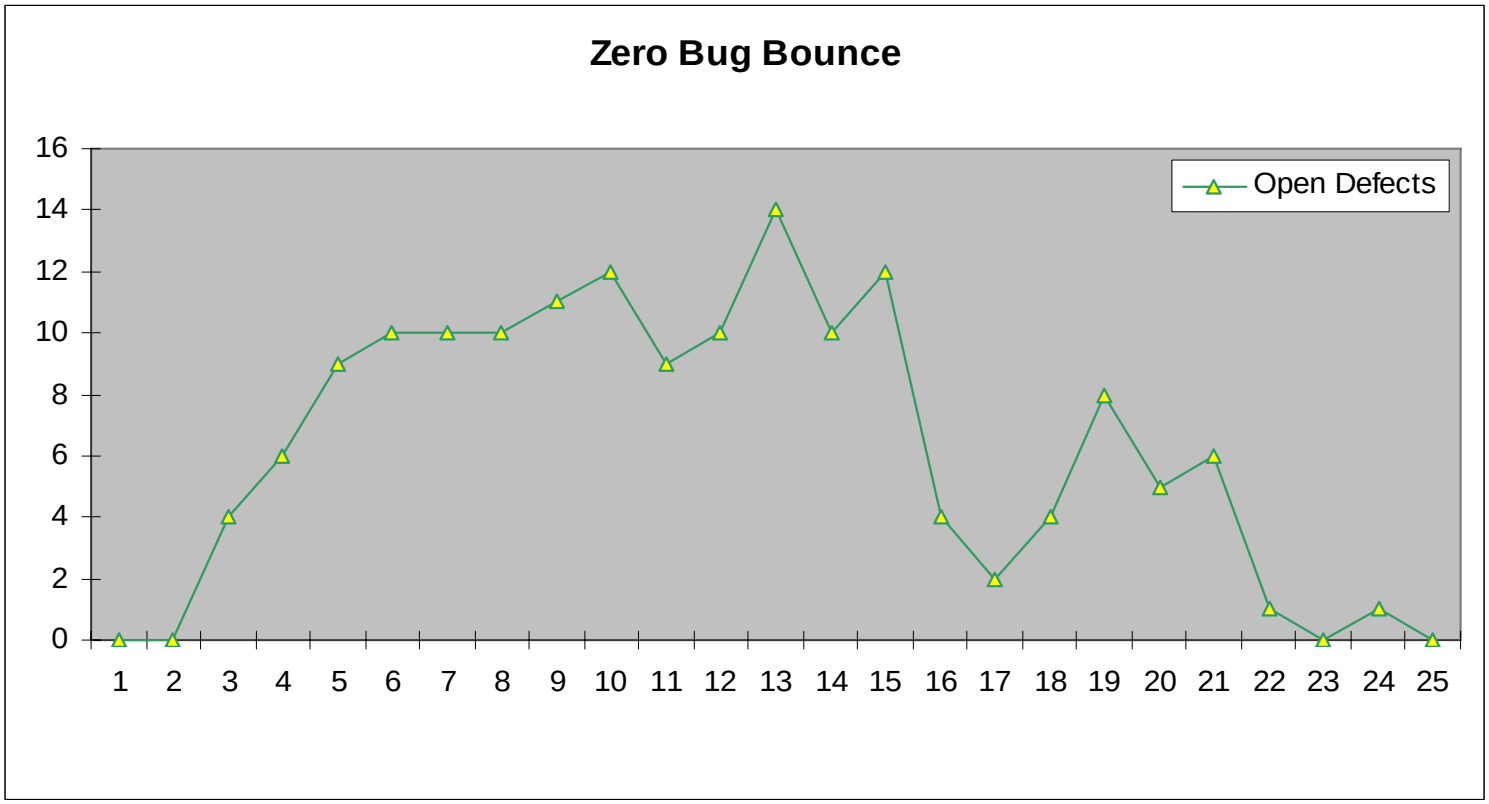
The Zero Bug Bounce (ZBB) is a defect management technique made popular by Microsoft. Strictly speaking, it is the point in the test effort of a project when the developers have corrected ALL open defects and they have essentially "caught up" with the test team's defect discovery rate. The "bounce" occurs when the test team finds additional defects and the development team must again begin defect correction activities.

After the initial bounce occurs, peaks in open defects will become noticeably smaller and should continue to decrease until the application is stable enough to release to production. This is what I call the *ripple effect* of the ZBB.

Defect Management with the Zero Bug Bounce

How do you track the ZBB?

The Zero Bug Bounce is tracked by charting the number of Open defects at the end of each day during test execution.



Defect Management with the Zero Bug Bounce

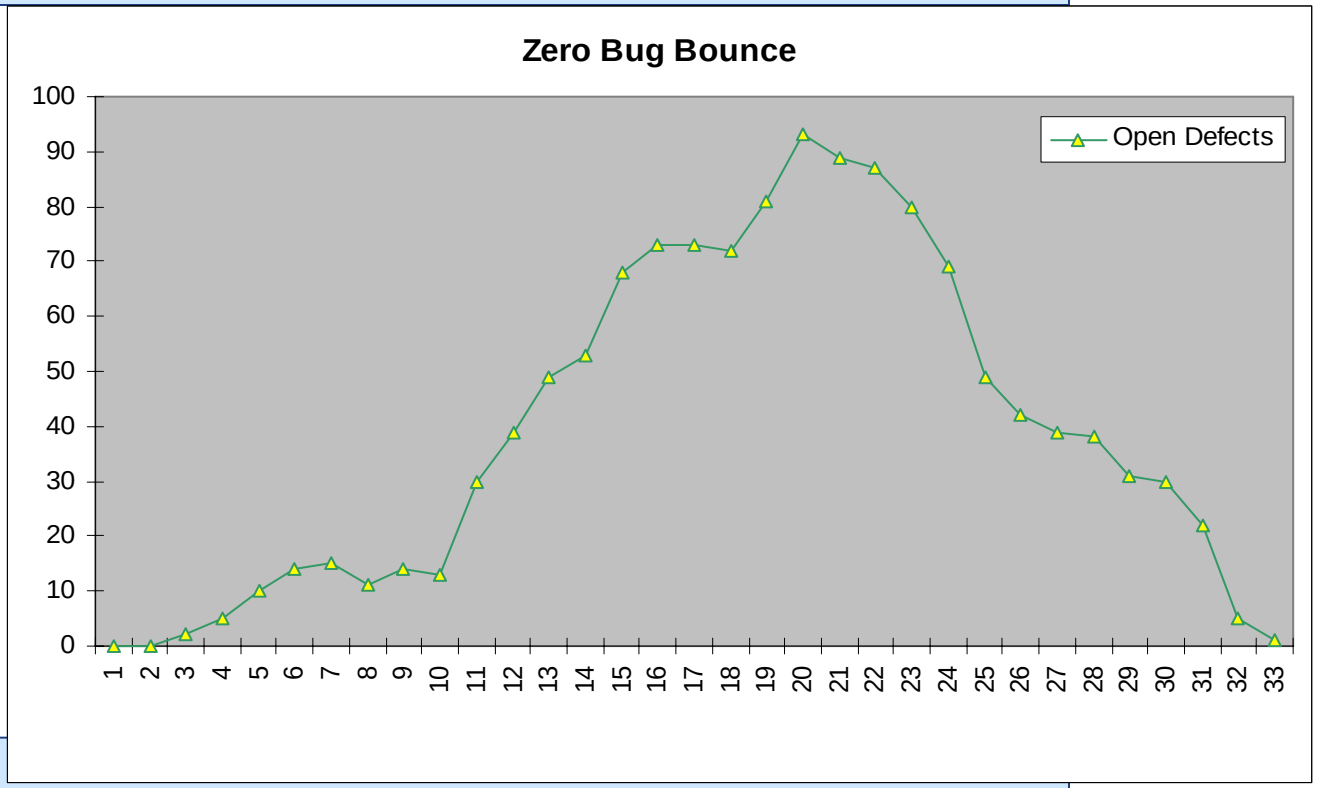
Some Notes On the ZBB

- The “bounce” does not always happen at zero
- The initial “bounce” typically occurs near the end of test execution
- There IS a *ripple effect*
- Use the height and length of the *ripple effect*, in addition to the timing of the initial bounce, to determine if the application is stable enough to be released to production

Analyzing the Graph

Question

- Is the application under test stable enough to release into the production environment?



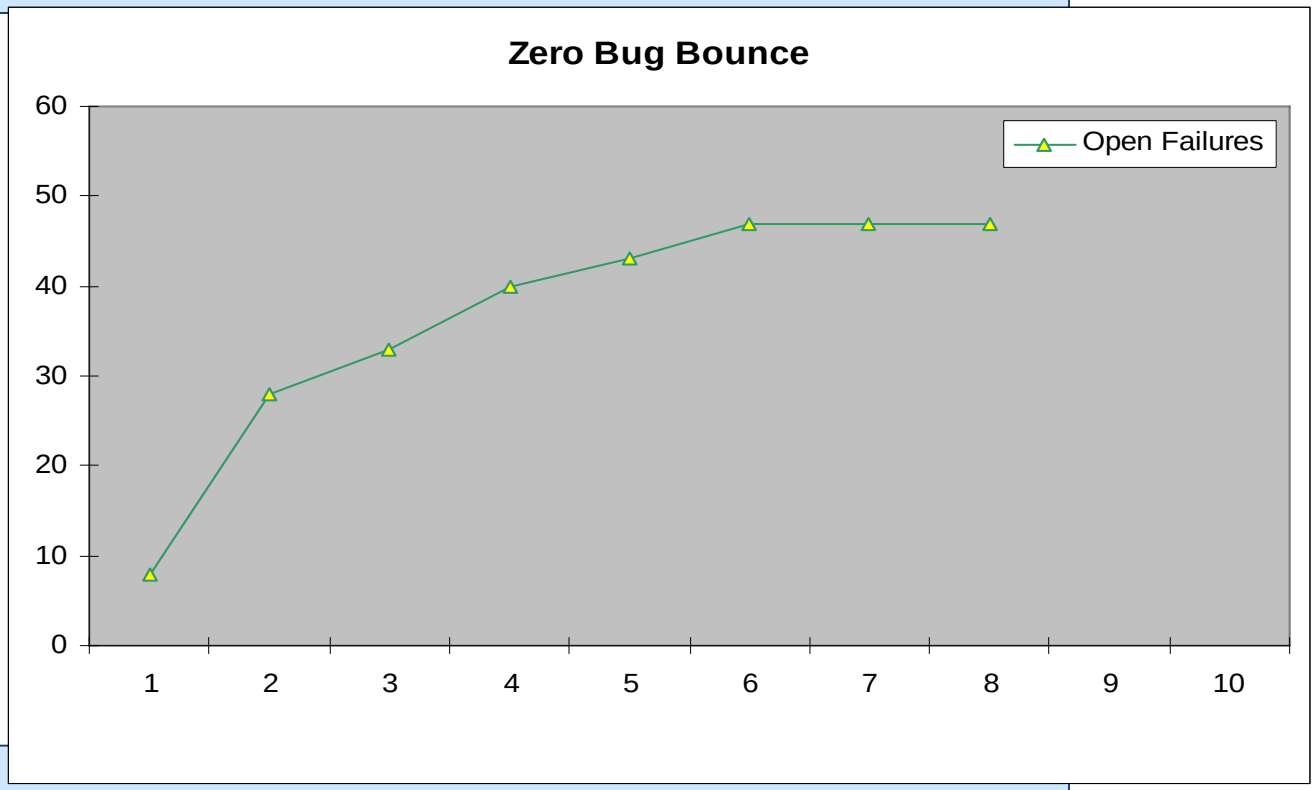
Answer

- Possibly, but not likely. There is a significant chance that a ripple effect will occur.

Analyzing the Graph

Question

- What is wrong with this picture? Can the application be released in 2 days?



Answer

- The developers are not correcting the defects in a timely manner. The application should not be released in 2 days.

Conclusion

The S-Curve and Zero Bug Bounce graphs can improve your ability to manage and track a test effort by providing visual clarity of the issues faced during test execution. Utilizing these graphs to measure and track test progress helps ensure timely and accurate delivery of a high-quality application to the production environment by:

- Helping to determine the resources necessary to complete the test effort in a timely manner
- Report the progress of the test effort through objective test metrics
- Assess the risk of component or application failure prior to release to production

Q & A


the mark of software quality
No one tries to fail.
If true, then why do seven out of ten new software systems fail in some way after they have been released to customers? They fail because many companies do not have processes implemented to catch defects early in the software development life cycle. Additionally, failure occurs because there is no sense of accountability among the employees. Failure is the direct result of lacking a definitive plan to control quality.
Aim to succeed.
Quality Assurance (QA) pinpoints each step in the software development process and assigns responsibility. QA relieves the uncertainty and replaces it with the confidence to catch defects and produce highly reliable software.
Why QA?
Simple. Adopting a QA process ensures your software will do what you promised. Kept promises are a hard currency in the world of business that can build a reputation as well as the bottom line.
Quality Assurance isn't a revolution of ideas, it is peace of mind.
Questions
&
Answers