



Driving Product Quality  
Towards Release  
Bhushan Gupta  
Nike Inc.

October 19, 2010

# Agenda



- Introduction
- GQM
- Release Criteria – a measuring stick for release readiness
- Measuring Scope Coverage
- Measuring Quality
- Code Volatility and Defect Aging
- Proactive approach towards Quality
- Conclusion
- Q&A

# Are We Ready to Release?



- Voting – Thumbs UP OR Thumbs Down
- I feel its good quality – gut feel, hunch
- Product meets business goals –
  - Providing value to the customer
  - Achieving revenue and profit goals
  - Enhancing brand image

# Applying GQM To Release Quality



- GOAL – Deliver a quality product on schedule
- Question –
  - Would we be able to deliver the intended functionality on time?
  - Would the functionality have intended quality?
- Metrics
  - Scope Coverage
  - Quality Measurements

# What is a Release Criteria?



- Definition – a documented set of conditions that has been agreed upon by the stakeholders and must be met before releasing the product.
- Areas to Set the Conditions Around
  - **Requirements Coverage**
  - **Test Coverage**
  - **Defect trends by Severity**
  - **Highest Severity Open Defects not Planned to be Fixed**
  - **Number of Defects Fixed but NOT Verified**
  - **Code Volatility Measures**
  - **Other Release Readiness aspects:**
    - **Training and Support Requirements**
    - **Invention Disclosures**

# Release Criteria - Example



Criterion	Description	Owner
Functionality/ Requirements	100 Percent of the planned requirements completed	Development Manager
<b>Test Coverage</b>	<b>100 percent test cases executed</b>	<b>QA Manager</b>
Defects	<ul style="list-style-type: none"> <li>100 percent “high severity” defects addressed as either:                             <ol style="list-style-type: none"> <li>Fixed, verified, and closed</li> <li>Workarounds available where possible</li> <li>Support cost estimated and agreed upon</li> </ol> </li> <li>100 Percent customer impacting “medium severity” defects understood</li> <li><b>100 percent defect fixes verified</b></li> <li>Find rate declining for 3 consecutive weeks</li> </ul>	Program Manager + <b>QA Manager</b>
Support Readiness	<ul style="list-style-type: none"> <li>Release notes up to date with workarounds where available</li> <li>Documentation/Training material ready</li> </ul>	Support Manager
Marketing Readiness	<ul style="list-style-type: none"> <li>Rollout plans ready and communicated</li> </ul>	Marketing Manager
Development	<ul style="list-style-type: none"> <li>Intellectual property activities completed</li> <li>Open defects moved to next release</li> </ul>	Program Manager







- **New Functionality Testing**
  - Following Test Strategy/Plan
  - Create Scope Coverage Matrix
  - Creation of Test Cases
  - Test Case Execution
  - Enhancing Test Cases for Efficiency and Effectiveness
  - Defect Logging
  - Defect Reproduction and Resolution Support
- **Regression Phase**
  - Test Case Re-execution for Defect Fix Verification
  - Reopen Defects
  - Re-execution of a Subset of Test Cases
  - Finding New Defects

# Release Criteria – % Planned Functionality Complete

## A Snap Shot



Functional Area	Test Cases Planned	Planned for execution till date	Executed till date	% of Total planned Executed	% Test Cases Passed	Test cases blocked
User Interface	45	35	32	71	98	3
Database	195	45	12	27	75	8
Output Display	25	20	15	60	99	2
Reporting	40	10	10	25	50	1
<b>Total</b>	<b>305</b>	<b>110</b>	<b>69</b>	<b>22</b>	<b>87</b>	<b>14</b>

Functional Area	User Interface	Database	Output Display	Reporting
<b>Status</b>				



## QA Proactive Approach

- Raise awareness
- Identify Risks
- Develop and Implement Mitigation with Stakeholders
- Monitor the situation with Stakeholders

## Example: Reporting Functionality

- Testing is on schedule
- A large # of test cases have failed
- Risk – Release criteria will not be met
- Mitigation – Reviews code and design, Improve Development Approach

# Other Quality Attributes



- Ease of Installation
- Performance
- Security
- Localization





## “Should 100% of defects be fixed?”

Defect Resolution Implies – a defect that is no longer an issue for this release

- Customer Impact – Severity and Frequency (basis of prioritization)
- Support Impact – Cost Incurred in Resolving Customer Problem
  - Release Notes Including workarounds
  - Call Support
- Release Plan
  - Hot fixes
  - Plan for next release
- High Risk of Fixing – too late in the development cycle
- Customer Loyalty and Perception

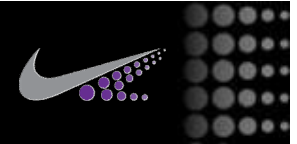


## Release Criteria - % Defect Resolved – Resolution (Fix)

### Modes of Defect Resolution:

- Code Fix
- Low Customer Impact and Frequency – No Fix Needed
- Relatively Low Support Cost – Easy Workaround
- Planned for a Hot Fix
- Include in Future Release
- High Risk – Impacts Multiple Functional Areas, Future Release

# Testing is Time Bound - Find and Fix as many defects as possible by Release Date



Regression

## Weekly Defect Find Rate by Severity

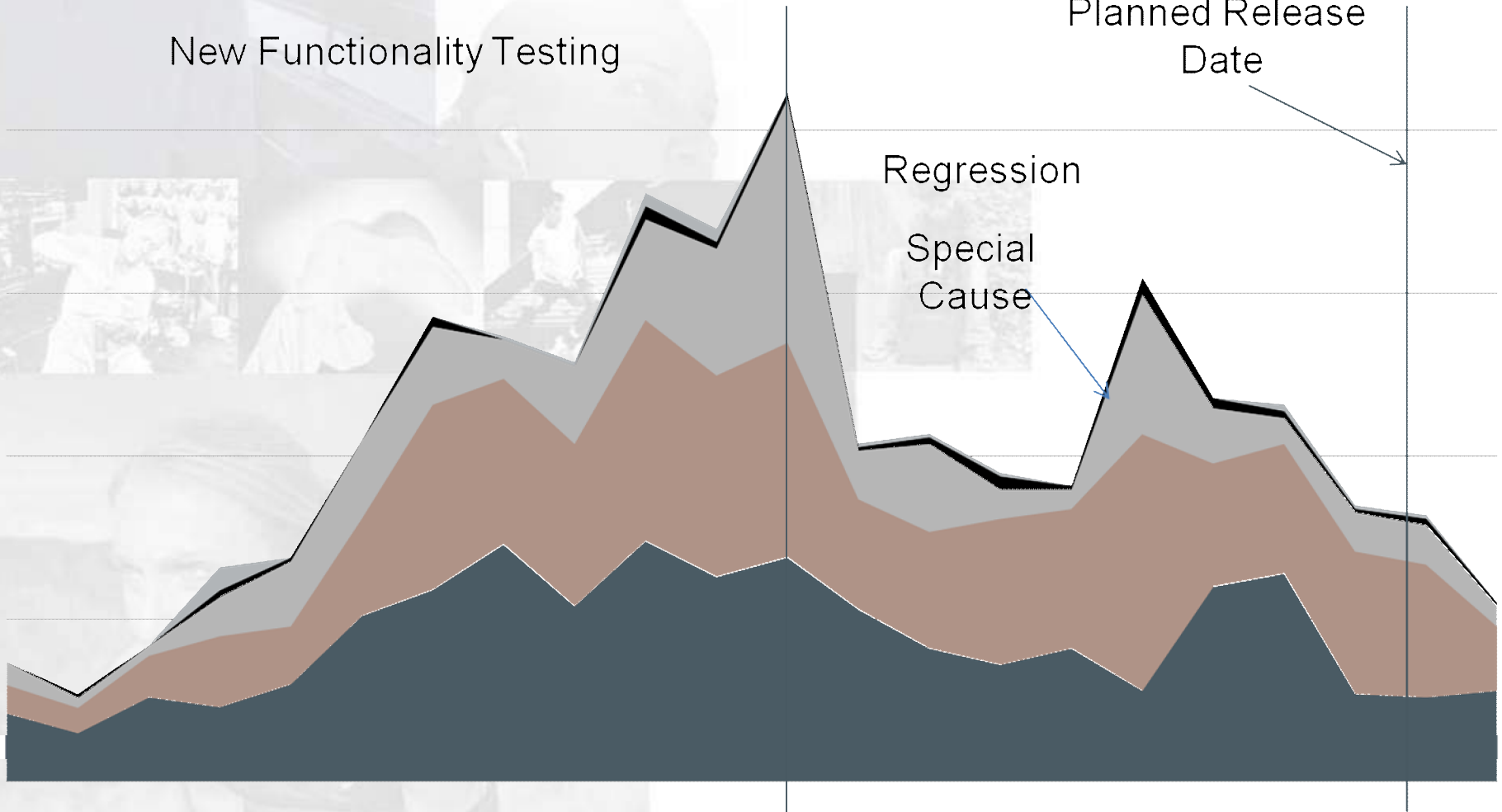
■ Urgent ■ High ■ Medium ■ Low ■ None

New Functionality Testing

Planned Release Date

Regression

Special Cause





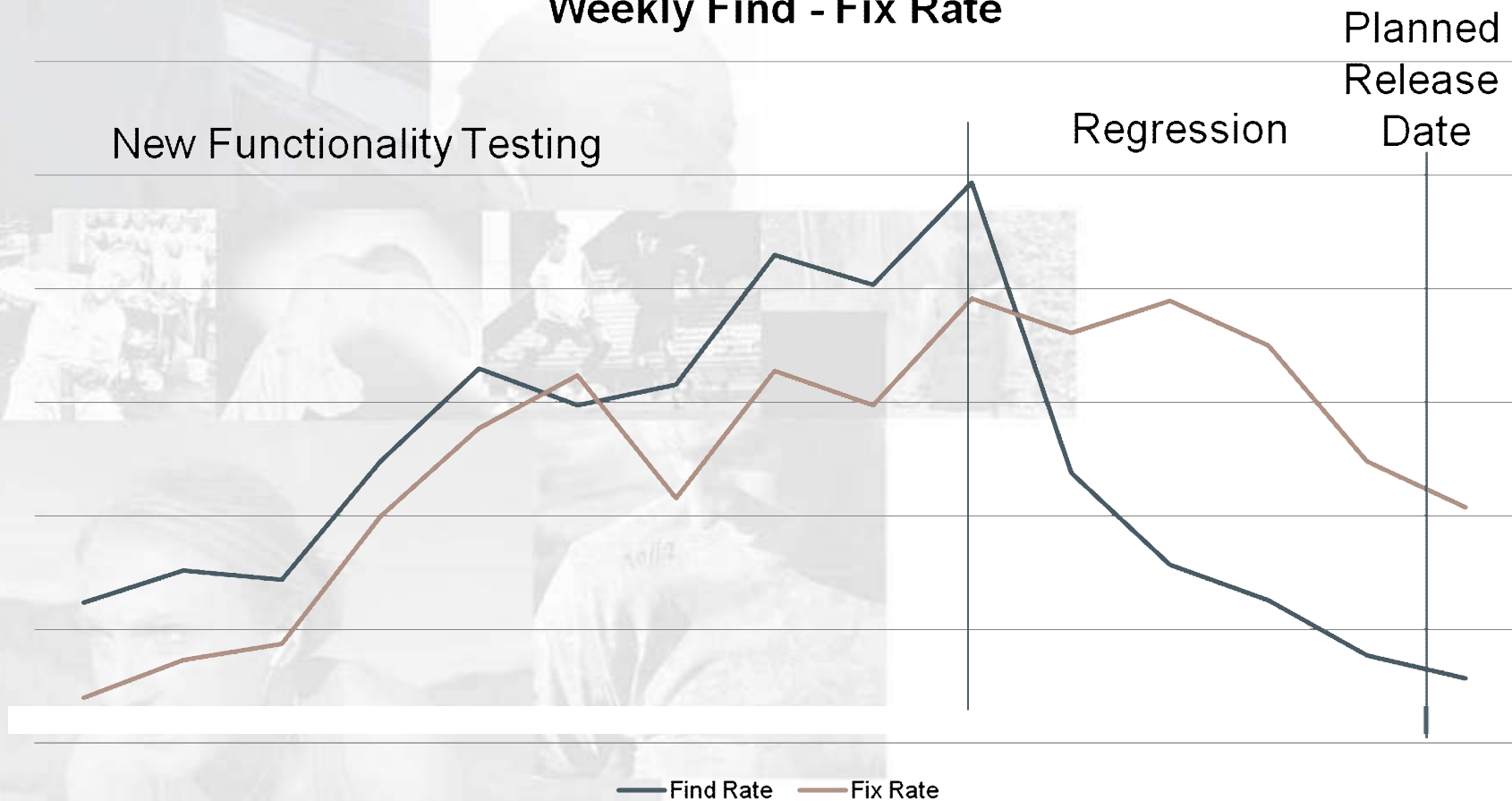
- Initial Ramp-Up
  - Understanding functionality
  - Planned Functionality not Yet Complete OR Blocked
  - End-to-End Workflow not Ready
  - Test Case Repair and Development
- Study Incline and Peak
- Tapering Off
  - Complex Functionality Testing Complete
  - Majority of Functionality Tested
  - Regression In progress
- Special Causes
  - Delayed Functionality
  - Feature Creep
  - Other Aspects – Localization

# Release Criteria - % Defect Resolved



## Resolution Rate Must keep Up with the Find Rate

Weekly Find - Fix Rate

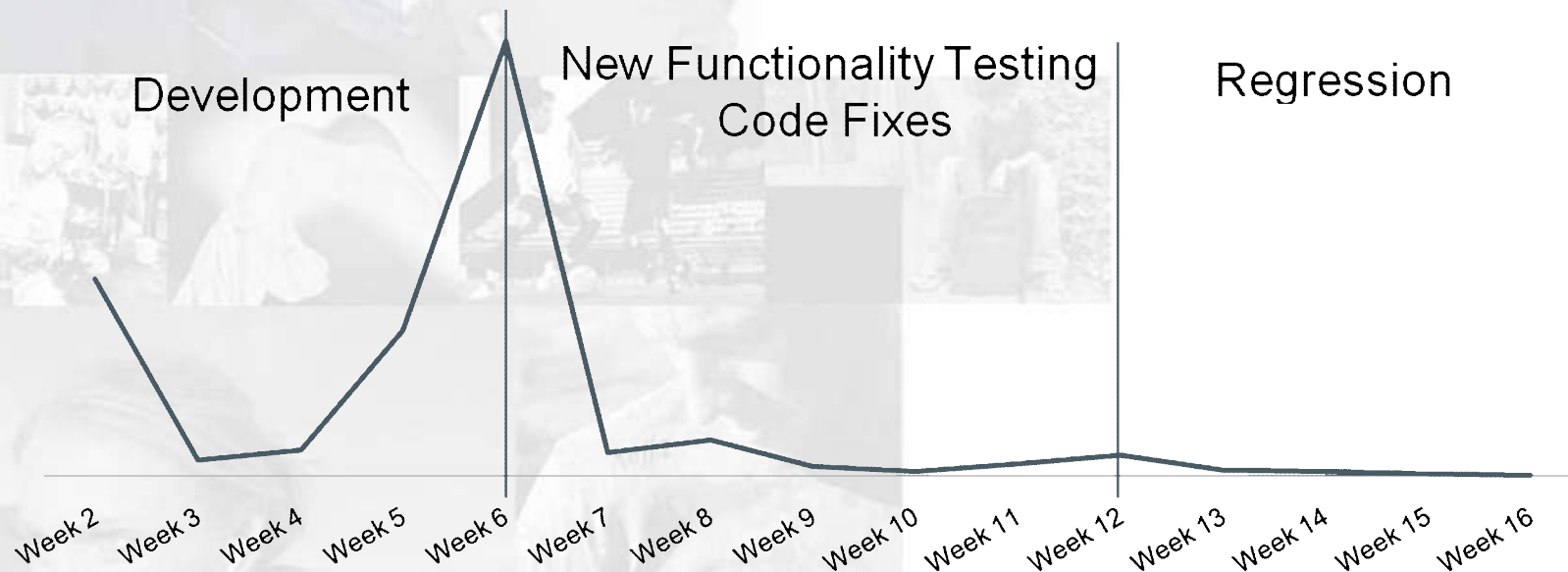


# Code Volatility – An Indicator of Defects Introduced



Definition – Lines of Code Delta between Two Release

Weekly % Lines Touched



# Code Volatility & Defects Relationship



- Defect Density – Number of Defects per 1K Lines of Code
- Code Volatility – Number of Lines Changed Between TWO Release

Defects Introduced Between TWO Releases  
= Defect Density \* Code Volatility/1K

## Considerations:

- Commented vs. Non-Commented
- Deleted Lines
- Code Reuse (Counted as New lines)
- Change in Environment

# Code Volatility Usage



- Number of Defects Anticipated in a Code Drop
- Resources Required
- Test Effectiveness
- Release Readiness – Controlled Code Volatility



# Defect Resolution Rate – Defect Aging



Defect Aging – period of time a defect stays in a particular state  
Prominent States and Aging Time:

- New – ASAP
  - Immediate Reproduction
  - Need for Logs and other Data for RCA
  - Replication of Test Environment
- Open – Necessary Time to Resolve the Defect
- Fixed – In a Reasonable Time
- Accuracy of Fixed Rate to Meaningfully predict Release Date
- Verified – Don't Care
- Closed – Final State

Ultimately Defect Aging Time is a Business Decision

# An Example of Defect Aging

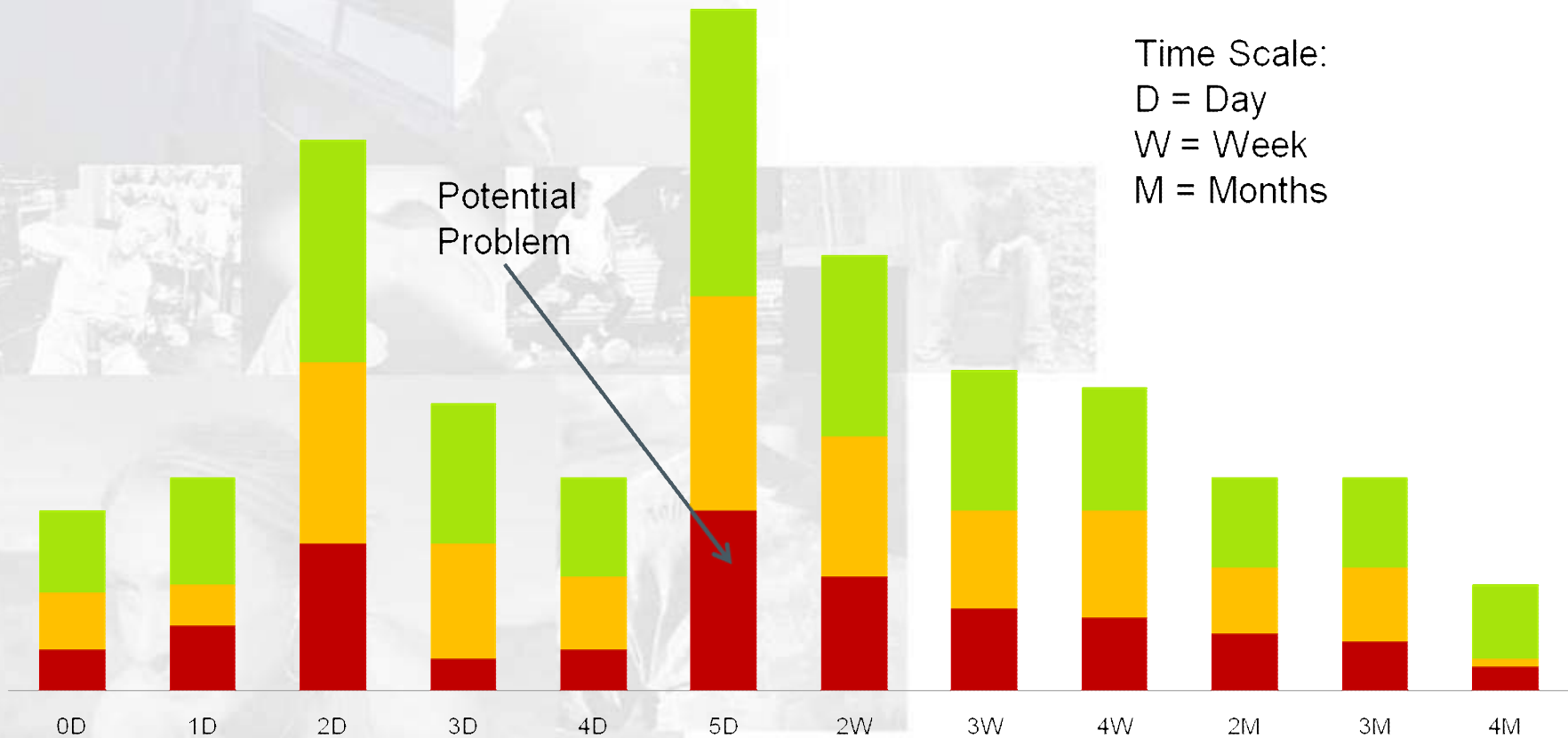


## Defect Aging by Severity

■ High ■ Medium ■ Low

Time Scale:  
D = Day  
W = Week  
M = Months

Potential Problem





- We Have Meet the Release Criteria – Kudos to Everyone
  - Functional Coverage
  - Quality Coverage
  - Other Conditions
  
- We Are Unable to Meet the Release Criteria  
Business Decision to move forward
  - Delay the release
  - Pay the cost of poor quality
    - = Support Cost + Customer Loyalty

# Proactive QA Approaches



- Prevention is better than cure
- Get to Know the Product Early
- Understand High Risk Functional Areas
- Understand Root Causes for Testing Delays
- Understand Testing Capacity and Resource Needs
- Provide Complete Support for Defect Resolution
- Be Innovative – Use Effective Alternative Testing Approaches
- Remember “we are all in it together”

# Conclusion



- Release criteria minimizes the polarity tension
- A release should be bonded by a release criteria
- At any time during testing the QA team can evaluate the progress towards release and act upon the data
- Prevention is better than cure
- A QA organization can proactively improve release quality and help maintain schedule

Thank You.



Questions ??