## Presenter Profile

Mr. Subrahmanya Somayaji is currently Portfolio Manager for Test Offerings in the North America Quality and Testing Services organization within IBM's Global Business Services. He is responsible for development and delivery of innovative QA solutions to clients across industries. He helps clients realize quality improvement, optimization and efficiency by leveraging analytics and automation.

Mr. Somayaji specializes in solution design and implementation of QA transformation programs integrating tools, best practices and IBM Research assets. He has experience of implementing IBM's leading QA innovations including Combinatorial Test Design (CTD), Defect Analytics, Document Analytics, Code Analysis, Adaptive Automation and Continuous Testing. He has over 15 years of experience at IBM managing global teams in delivering large scale application development and QA testing services.




## Description of the presentation

Many organizations are facing two major testing challenges that are working against their objectives of preventing defects, improving quality and reducing cost.

- Typical defect analysis metrics not providing actionable insights for defect prevention
- Inefficient test design and ever increasing test case volume

This presentation is about quality management best practices that help clients optimize their test suites and accelerate defect prevention. A potential real-time analysis of all defects including non-code and process defects provide more actionable insights, highlights best practices, identifies high-priority cost reduction opportunities associated with software test practices and defect prevention. Test design optimization ensures quality while reducing costs.

# Defect Analytics and Test Design Optimization for Quality Improvement

## Contents

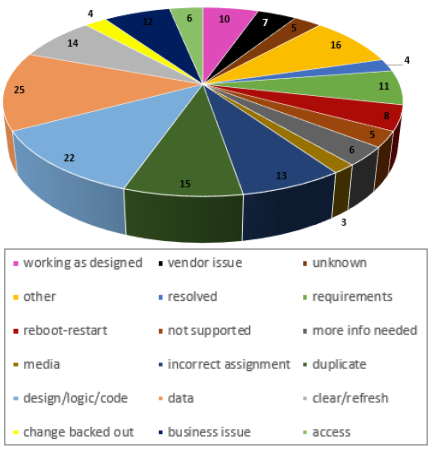
- Traditional defect analysis not telling you the full story
- Defect analytics best practices
- Inefficient test design and “runaway” test suites
- Test design best practices




## Traditional defect analysis is not telling you the full story

- Typical defect classification based on “root cause” classification option produce too many categories
- Lacks clear indication of corrective or preventive actions needs to be taken
- Ambiguities due to mixing of categories (“requirements” vs “more info needed”) and overlapping definitions (“requirements” vs “business issue”)
- Root Cause Analysis (RCA) looks deeper into a subset of defects and attempts to tell why a defect is injected
- RCA results can be subjective and influenced by the analyst's perspective.

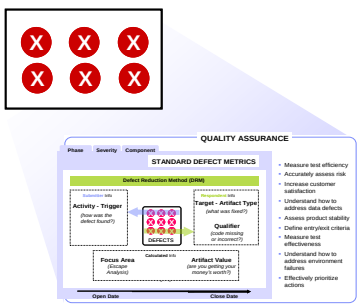
Representative sample of a defect chart







## Collaborative Defect Management (CDM) helps development and test teams quickly identify risks and improve quality




**What is it?**

A real-time defect-based analysis to identify high-priority cost reduction opportunities associated with software test practices and defect prevention.

The Collaborative Defect Management utilizes:

- An IBM approach that encompasses classification and analysis
- Benchmarks based on defect data collected for nearly two decades, mapped to industry, quality level, and test phase/activity
- Metrics to objectively quantify and prioritize risk and opportunity, and to measure quality

The CDM includes:

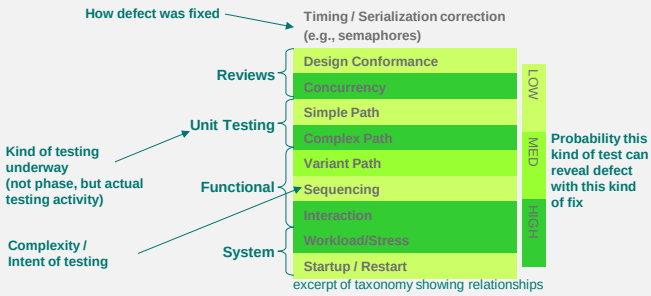
- Classification of defects raised during review and/or test activities
- Analysis and measurement of risks (General Test Effectiveness, Test Design Effectiveness, System Stability and Completeness) and Quality (Evaluation against Quality Benchmarks, Focus Area Identification, and Artifact Value Assessment)
- Recommendations to address the highest priority areas of improvement

**The Collaborative Defect Management provides business value by:**

- Identifying and recommending specific actions to reduce risk and cost associated with developing and testing systems or applications, even within cycle
- Identify actions to detect defects early and prevent defects from occurring
- Acknowledging and reinforcing development and test process strengths
- Guiding organizations to utilize test practices that can increase their ROI while delivering improved quality

## How CDM works?

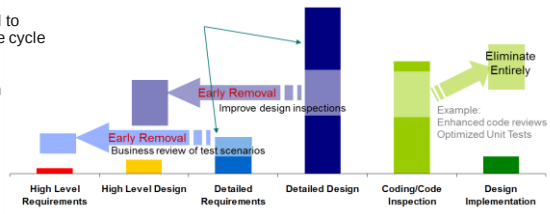
**Key Features**
- Mutually exclusive categories
- Classify how defects are found
- Classify how defects are fixed
- Establish relationships between tests and fixes
- Anomalies reveal opportunities for improvement




Defect fix types are algorithmically mapped to **Focus Areas**, which tell us where in the life cycle we could have found the defect earlier.

Recommendations target Focus Areas with excess defects to defect reduction goals.

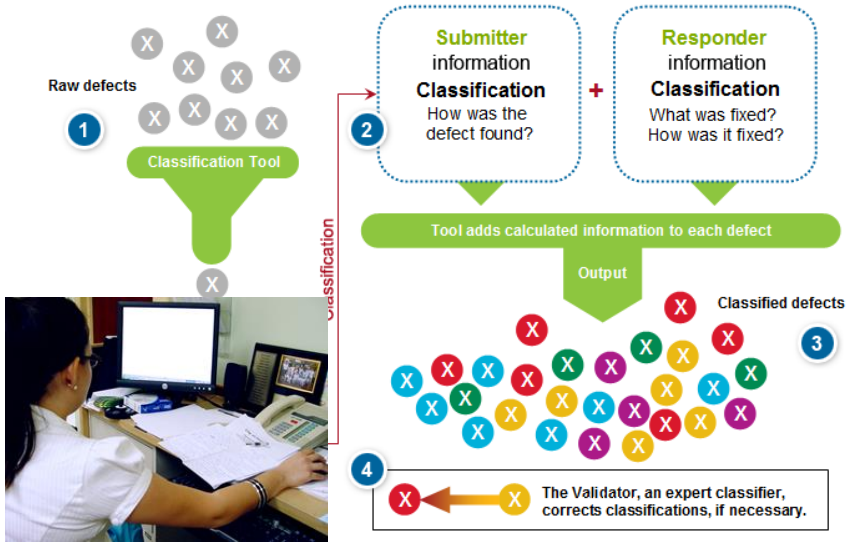
Additional classified features of the defects identify discrete processes within each Focus Area.





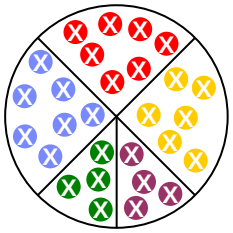


## CDM - Defect Classification

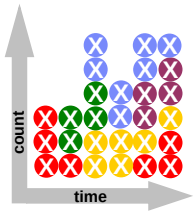


## CDM - Defect Analysis

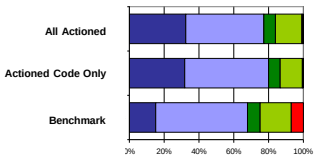
**SMEs** analyze the distributions and trends of the classified defects and compare them to established benchmarks for similar projects (or to baselines for the same project).



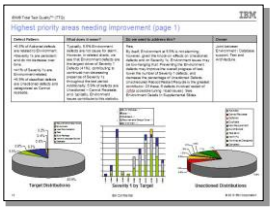
Distribution




Trend




Benchmark comparison



Final Report

**Output**

- Describe test effectiveness and efficiency
- Highlight good trends / practices
- Accurately assess risk
- Identify Focus Areas (Injection Points)
- Explore data improvement
- Assess product stability
- Help define entry/exit criteria
- Measure code quality
- Suggest environment improvements
- Effectively prioritize actions




## Why CDM instead of Root Cause Analysis?

**Traditional analysis methods**
*(e.g. Root Cause)*

- Analyzes a subset of defects at great cost
- *Resources diverted from other efforts*
- *Time required per defect is significant*
- Results are typically subjective and not always accurate

**Collaborative Defect Management (CDM)**

- Analyzes all defects in the target phases, including non-code defects and process defects
- Frees resources to focus on project milestones
- Delivers a full, robust analysis
- Highlights best practices and opportunities for improvement

Non-scalable, historical review

Often requires most well-paid resources

Analysis diverts effort and resources

Not comprehensive and not orthogonal

Focused on code

Analysis limited to project or organization

**Shifting schema, methods, and standards**

Potentially real-time analysis that considers all defects

Classification performed by the project team

Analysis performed by an IBM professional

Identifies good, repeatable practices

Examines Code, Data, Environment, and Process

Analysis compares performance to Benchmarks

**20 Years of IBM's ODC Research and Experience**

## Inefficient test design and “runaway” test suites

Many teams today use an ad hoc design method for test design which results in :

- Too many test cases to create, execute and maintain
- Testing the system behavior in the most basic way not taking into account variety of conditions and circumstances a system will be invoked by end users
- Spending a lot of time and effort on testing while still finding problems late in the cycle
- Lack of ability to identify which tests are missing
- Lack of ability to quantify the risk of testing less or the value of testing more



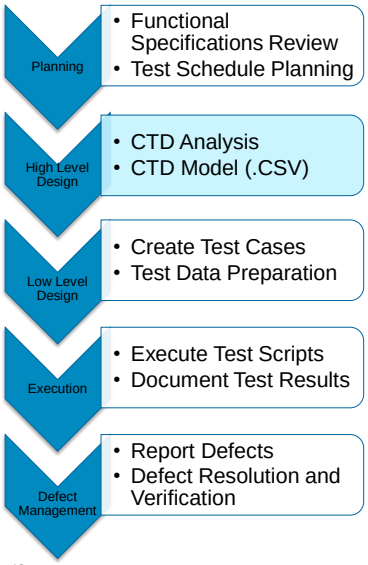

## Best Practices in Test Design

Test design best practices will help overcome some of these challenges

| | |
|---|---|
| **Equivalence partitioning** | Divide input data into partitions of equivalent data that tend to have same behavior and test each partition |
| **Boundary value analysis** | Testing boundaries for range of values |
| **Variation tests** | Negative tests (bad paths)<br>Invoke application under specific condition such as different user profile, alternate browser etc |
| **Sequencing tests** | Test multiple functions by invoking them more than one sequence |
| **Interaction tests** | Test multiple functions simultaneously in ways different than specified in requirements |
| **State transition technique** | Test for all valid states as the application state changes from one to another |



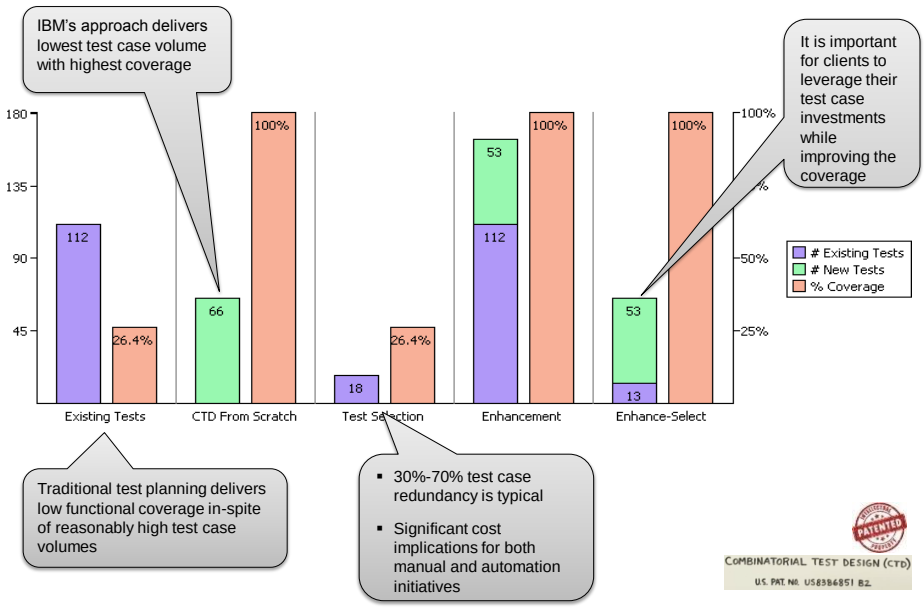

## IBM Combinatorial Test Design optimizes test case and test data coverage




- CTD is a systematic tool based approach to test planning that produces smaller test plans that find more defects than manual approaches to test planning
- A repeatable test design process that doesn't depend solely on the experience of the test team
- Develop new tests as well as optimize existing tests
- Accelerates test design and test automation
- Helps achieve 100% testing coverage of the modeled functionality with up to 40% reduction in number of test cases
  - ✓Fills gaps, improves testing coverage, finds more defects
  - ✓Reduces test cases, reduces effort & cost
- Improved ability to predict quality




## CTD Coverage Analysis Example



## CTD Case Studies

- **Large North American Bank**
  - Applied CTD to optimize 5000+ existing tests achieving **32% test case reduction** and input into automation
  - Identified significant coverage gaps
  - Built base functional models to implement CTD for Agile sprints
- **Canadian Bank**
  - Developed CTD models for Mobile app and reduced test cases from 731 to 436
  - Increased good path coverage from **53% to 100%** and bad path coverage from 75% to 100%
- **Telco**
  - Reverse-engineered model revealed in 117 hand-written test cases that had ~70% coverage of pairwise interaction; could be replaced by 12 test cases with 100% coverage
- **Manufacturing and sales**
  - Large ERP implementation needed to select optimal subset of test cases to meet implementation deadline; for one domain 12% of tests could be omitted without reducing coverage; for another domain, **58% of tests** could be omitted without reducing coverage
- **Health Insurance**
  - Operationalized CTD as part of release test cycle and applied CTD for all critical application releases to achieve average 25% test case reduction