Quantifying Software Reliability and Readiness

Jack Olivieri
Lead Multi-Discipline System Eng
MITRE, Dept. E53D
V2.1

MITRE

Summary

- While rigorous, quantitative product readiness criteria are often used for hardware business cases and related areas, software reliability and readiness is often assessed more subjectively & qualitatively

- This presentation:
  - Defines motivation and requirements for quantitative software readiness criteria for product release decisions
  - Proposes method for organizing and streamlining existing quality and reliability data into a simple metric and visualization which is applicable across products and releases

Outline

- Business Problem
  - Problem Definition
  - Current Best Practices & Software Readiness Methods
  - Quantitative vs. Qualitative Techniques
- Software Readiness Criteria
- Deploying Readiness Criteria
- Summary

Problem Definition

- **Why specify and measure software readiness?**
- **The fundamental reason for measuring software readiness is to support technical and business decisions while achieving increased customer satisfaction:**
  - **Full rate production/ general availability release readiness**
  - **Setting customer expectations**
  - **Better control the schedule, cost, and quality of software releases**
  - **Estimating risk of liquidated damages**
  - **Most organizations do this already for hardware and this is an analogous activity**
    - **Identification of contribution of SW vs HW**
    - **SW could be 90% of failure contribution; may be much easier to correct**

Goals for a Quantitative Technique

- Create an easy-to-understand, quantitative software readiness index that:

  1. Enables quantitative “pass” criteria to be set from product requirements
  2. Is easy to calculate from existing data
  3. Offers a meaningful visualization
  4. Enables release-to-release comparisons
  5. Enables product-to-product comparisons
  6. Complements software modeling and defensive programming

- Non-goals (out-of-scope)

  1. Business metrics such as cost, resources, manufacturing, materials
  2. Schedule performance metrics

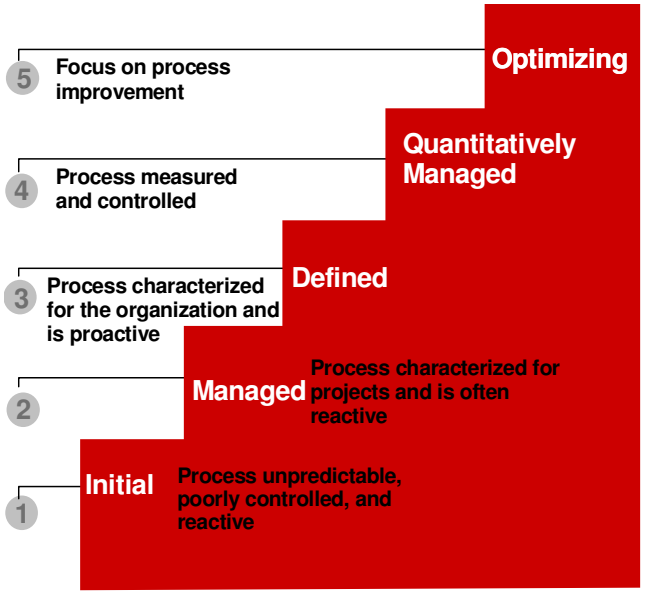
Best Practice – CMMI

Background - Quantitative v. Qualitative Criteria

- Capability Maturity Model® Integration (CMMI),
- Version 1.2 CMMISM for Development
  - From Carnegie Mellon University, Software Engineering Institute –
  - http://www.sei.cmu.edu/cmmi/
- Research on defining software measurement data
  - Establish and maintain the description of a defined measurement and analysis process
  - Establish and maintain quantitative objectives for the measurement and analysis process, which address quality and process performance, based on customer needs and business objectives
  - Objectively evaluate adherence of the measurement and analysis process against its process description, standards, and procedures, and address noncompliance
- Plan-do-check-act process
- SEI Core Measures:
  - Size, effort, problem count, schedule
- Software Measurement for DoD Systems based on CMMI
  - Tech report Recommendations for Initial Core Measures
  - Keene Model (based on CMMI & KLOC count)

Best practice is to evolve to quantitatively managed processes, including software readiness criteria.

SEI Research Focuses on Growing CMMI Maturity from Qualitative process (CMMI levels 1-3) to a more Qualitative process (CMMI levels 4-5)



Other Current Standards and Practices

- **IEEE/ EIA 12207.0, "Standard for Information Technology-Software Life Cycle Processes", is a standard that establishes a common framework for software life cycle process. This standard officially replaced MIL-STD-498 for the development of DoD software systems in August 1998.**
  - **The purpose of this International Standard is to provide a defined set of processes to facilitate communication among acquirers, suppliers and other stakeholders in the life cycle of a software product. "This International Standard does not detail the life cycle processes in terms of methods or procedures required to meet the requirements and outcomes of a process"**
- **Telcordia GR 282 "Software Reliability and Quality Acceptance Criteria" Dec 1999 has qualitative requirements**
- ***AIR FORCE INSTRUCTION 10-602 - 18 MARCH 2005***

A8.2. (Appendix 8 Software Maturity)

Table A8.1. Software Severity Levels and Weights.

Priority/Severity Level	Impact	Description	Severity Weight (Points)
1	System Abort	A software or firmware problem that results in a system abort or loss.	30
2	System degraded No Work Around	A software or firmware problem that severely degrades the system and no alternative work around exists,	15
3	System Degraded Work Around	A software or firmware problem that severely degrades the system and an alternative work around exists (e.g., system rerouting through operator actions).	5
4	Software Problem	An indicated software or firmware problem that doesn't severely degrade the system or any essential system function.	2
5	Minor Fault	All other minor deficiencies or nonfunctional faults.	1

Software severity levels and weights based on a scale

Technical Report

CMU/SEI-92-TR-019

ESC-TR-92-019

Software Measurement for DoD Systems:

Recommendations for Initial Core Measures

Mostly Process Based

(aligned with CMMI) but 4 "units of measure" recommended

Unit of measure	Characteristics addressed
Counts of physical source lines of code	Size, progress, reuse
Counts of staff-hours expended	Effort, cost, resource allocations
Calendar dates	Schedule
Counts of software problems and defects	Quality, readiness for delivery, improvement trends

Figure 3-1 Measures Recommended for Initial DoD Implementation

Outline

- Business Problem
- Software Readiness Criteria
  - Requirements For a Quantitative Technique
  - Primary Criteria Vectors
  - Fishbone Diagram
  - Per-Vector Details
  - Constructing an Index
- Deploying Readiness Criteria
- Summary

Requirements for Quantitative Technique

- **An improved, quantitative software readiness criteria should meet the following requirements:**
  1. **Quantitatively characterizes product's software quality and reliability at General Availability (Production and Deployment)**
  2. **Support quantitative "green (go)/yellow (go w/ condition)/red (no go)" criteria, derived from product requirements**
  3. **Be computable from SQA & second-pass system test interval thru system test complete**
  4. **Apply to most (or all) software-centric products**
  5. **Support easy release-to-release comparison**
  6. **Support meaningful and easy product-to-product comparison**
  7. **Ability to validate criteria with field data, and hence close the loop**
- **Implement a tool to track the SW Readiness against targets set at the start of the Execute phase.**

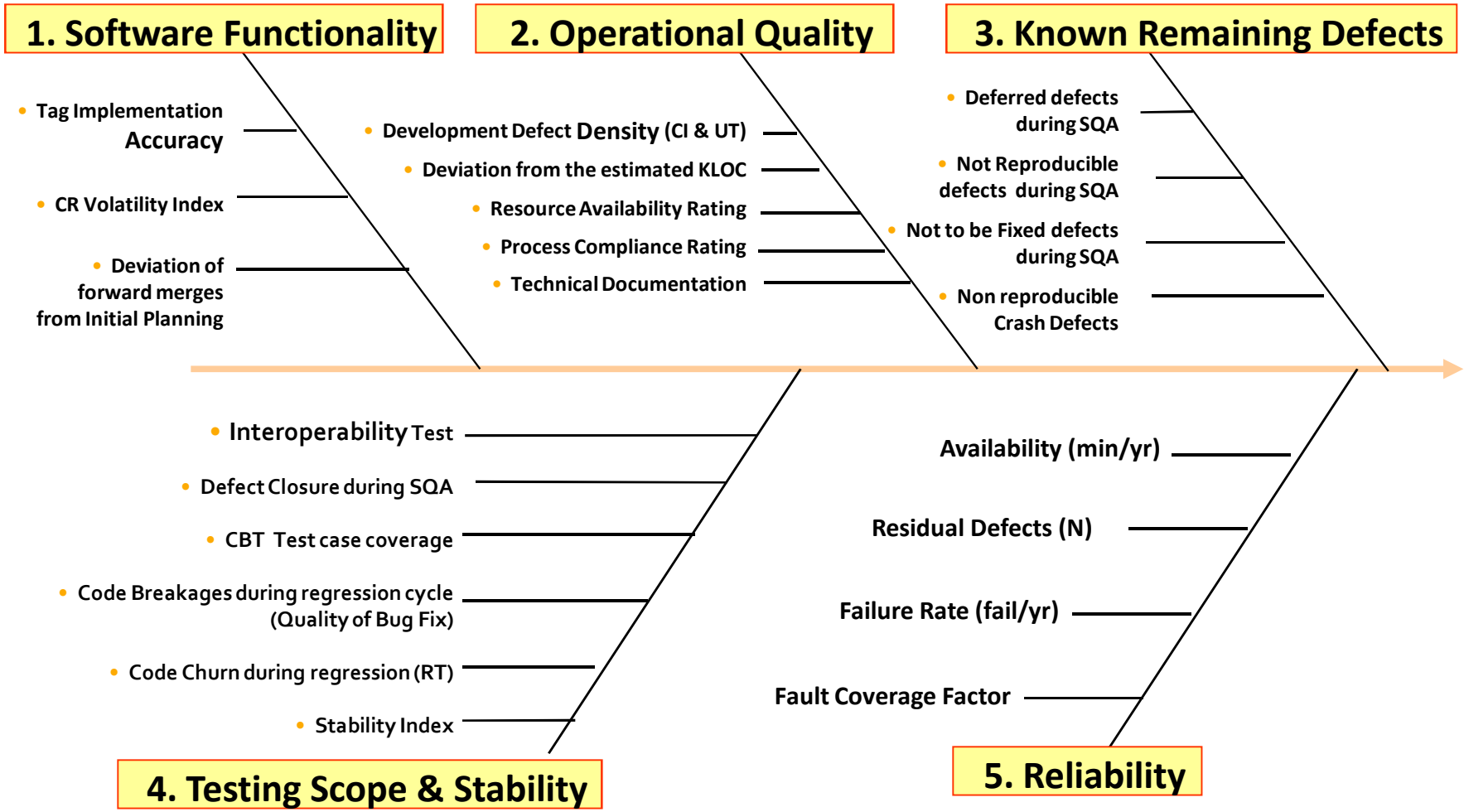
These requirements enable indicators and features useful to "business leaders" or "decision makers", project managers, program managers and system testers

Software Readiness Criteria Vectors

- The Software Readiness Index defines and measures the growth of the software system along the following five dimensions:
  - Software Functionality
  - Operational Quality
  - Known Remaining Defects
  - Testing Scope and Stability
  - Reliability
- This principal set of measurables are relatively orthogonal attributes for making the measurement descriptions exact and unambiguous.
- The framework relates discovery, reporting, and measurement of problems and defects
- Rules for creating unambiguous and explicit definitions or specifications of software problem and defect measurements are defined.

Since we concentrate on problems and defects, we shall define their meaning and discuss the relationship they have to other terms and entities.

Software Readiness Index Fishbone - Interrelationships



Evaluate test plan against the readiness index early part of execute/implement phase

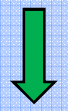
Software Readiness Index
Software Functionality Vector

What does it measure?

- Did we implement all the features (tags) correctly?
- How many major changes did we introduce?
- How many unplanned forward feature merges did we have?

How does it measure?

1. Feature Tag Implementation Accuracy
   - Example: percentage of Tags Correctly Implemented/Planned (e.g., 95%)



2. CR (Change Requests) Volatility Index
   - Example: percentage of confirmed major CRs accepted of Total number of CRs (e.g., 2%)

3. Deviation of forward merges from initial planning
   - Example: percentage of (Total actual forward merges - Initial planned forward merge) / Initial planned forward merge (e.g., 6%)

Specific set of attributes and metrics can be tailored for product lines

Software Readiness Index

Operational Quality Vector

What does it measure?

- How many defects did we remove prior to SQA?
- How good was our initial defect estimates based on KLOC?
- Were the planned resources adequate for the program (e.g. testing)?
- How well did we comply with the SQA process?
- What is the quality of customer documentation?

How does it measure?

1. Development Defect Density (Code Inspection & UT)
   - CUT Defects / (Total FT and FAT defects + CUT defects) (e.g. 60%)

2. Deviation from the estimated KLOC to actual KLOC (for new code)
   - Abs(Actual KLOC - Estimated KLOC) / Estimated KLOC (e.g. 30%)

3. Resource Availability Rating
   - No. of Staff Weeks with resources < 20% against plan / Total Staff Weeks (e.g. 50%)

4. Process Compliance Rating
   - Major NCs / Total NCs (e.g. 10%)

5. Technical Documentation
   - No. of defects identified and corrected for Technical Documentation / Total No of Pages (e.g. 0.6)

MITRE

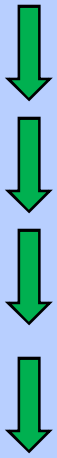
Software Readiness Index

Known Remaining Defects Vector

What does it measure?

- How many defects got deferred during SQA cycle?
- How many defects (out of planned test cases) were not reproducible?
- How many defects were judged not-to-be-fixed, hence, deferred?
- How many system crash defects that were not reproducible?

How does it measure?



1. Deferred defects during SQA cycle (FAT, FT and RT)
   - Percent sev 1 & sev 2 defects deferred / Total Defects Found During SQA Cycle (e.g. 3%)
2. Not Reproducible defects SQA cycle (FT, FAT and RT)
   - Percent not reproducible defects / Total Defects Found During SQA Cycle (e.g. 12%)
3. Not to be Fixed defects SQA cycle (FT, FAT and RT)
   - Percent not to be fixed defects / Total Defects Found During SQA Cycle (e.g. 3%)
4. Non reproducible Crash Defects (screen hangs, system crashes)
   - No. of residual crash defects (not reproducible) / Total number of crashes defects (e.g. 2%)

Software Readiness Index
Testing Scope & Stability Vector

What does it measure?

- How stable is the software system?
- How well did we do with interoperability and fault insertion tests?
- How many defects have been removed during system test?
- How much code churn and breakage during system test?

How does it measure?

1. Interoperability & Fault Insertion Test (Including Protocol Conformance Suite)
   - Test Cases passed / Total Interop Tests (e.g. 90%)
2. Defect Closure during SQA (FT and FAT)
   - No. of defects fixed / Total found defects (e.g. 59%)
3. Customer Based Trial Test case coverage
   - Test cases provided by PM & CT / Total Regression Cycle test cases (e.g. 50%)
4. Code Breakages during regression cycle (Quality of Bug Fix)
   - No. of Breakages / Total No. of Fix Available (e.g. 20%)
5. Code Churn during regression (RT)
   - (Code size after RT - Code size before RT) / Code size before RT (e.g. 5%)
6. Stability & Stress Index
   - As per Stability &Stress Index Test Plan (e.g. 97%)

Stability with normal-overload (monitor CPU/memory usage levels);

Robustness with stress (controls);

Recovery with fault Insertion

Software Readiness Index
Reliability Vector

What does it measure?

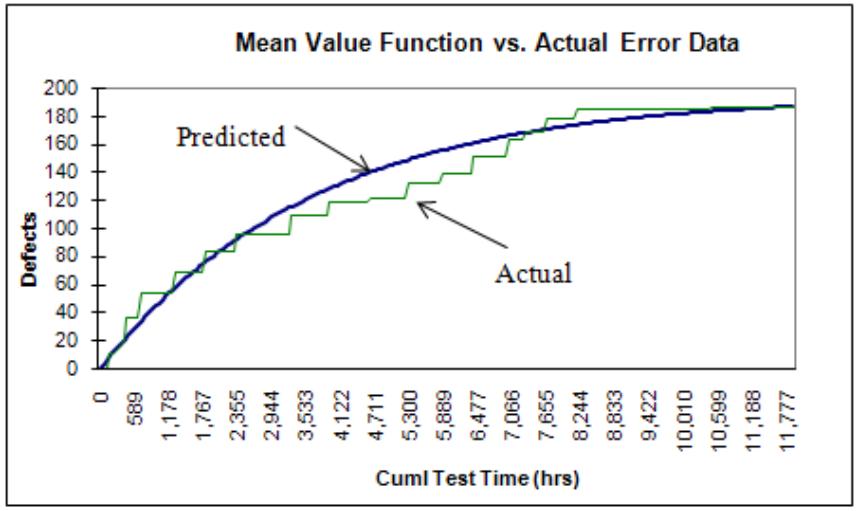
- How many latent defects may be remaining in the system?
- What is the predicted software failure rate in the field?
- What is the predicted service (unplanned) downtime due to software failures?

How does it measure?

1. Predicted Residual Defects (SRGM)[1]
   - Number of Predicted Residual Defects (e.g. 7)
2. Predicted Software Downtime
   - Min/yr/Sys (e.g. 6.4)
3. Predicted Software Failure Rate
   - Fail/yr/Sys (e.g. 0.25)
4. Fault Coverage Factor
   - % Negative Tests Passed/ Total negative Tests (weighted avg. of pass 1 & 2) (e.g. 97%)

[1]Note: SRGM is an *example* of how to predict residual defects; there are other ways to predict this (e.g., defect propagation modeling). Predictions are based purely on system test data. Also available are predictions based on Markov Models (HW + SW)

Input Data for SRG Model		
Hours	CRs/MRs	Week (W)
Cumulative Testing (Exposure) Time	Cumulative CRs/Defects (sev 1&2 only)	
131	0	W1
283	11	W2
330	14	W3
373	16	W4
458	20	W5
680	36	W6
1278	54	W7
1782	69	W8
2332	84	W9
3248	96	W10
3928	109	W11
4641	118	W12
5272	121	W13
5845	132	W14
6370	139	W15
6963	151	W16
6964	156	W17
7250	163	W18
7633	169	W19
8130	178	W20
8729	184	W21
9089	184	W22
10433	185	W23
11777	186	W24



Output of SRG Model		Goals
a (Estimated Defects/Faults)	193.6104995	
b (Per fault failure rate)	0.000275	
Predicted Residual Defects	7.610499527	<= 16
per fault fails/hr	0.00027477	
fails/hr	0.002091139	
fails/yr	18.3183751	
Calibration Factor	60	
Adjusted Failure Rate (fails/yr)	0.305306252	
Coverage Factor	94%	
Outage Rate (/yr)	0.019697178	<=0.10
Mean Outage Duration	20	
Downtime (min/yr)	0.39	<=2.5
Availability	99.9999%	

Note: Gray highlighted cells are historical estimates based on previous releases, similar code or actual data

Constructing an Index

☞ **While individual vectors serve as an indicator of software goodness (& risk) in each dimension, a sense of the overall software goodness (and risk) is needed.**

- **Factors to consider are:**
  - Each vector is computed as a weighted sum of the constituent variables; magnitude of each vector is normalized to 1.0
  - The weights for each variable is decided by the relative significance of that variable to the vector (indirectly to the product & program)
  - The vectors are similarly weighted and combined into a composite index
  - The equation for the composite index should be simple, insightful and easy to understand, i.e. a linear algebraic model
- **Additive model –** each vector contributes a fraction to the overall SRI and can affect the SRI up to its allocation worth.
- **Multiplicative model –** each vector is independent and affects the entire range of SRI directly; SRI very sensitive to each vector. A geometric weighted average using integer weights may be used.
- **Hybrid model –** vectors combined into normalized additive groups that are then multiplied to form the SRI; the significant variables affect the SRI more than the variables that are deemed less important; all variables contribute to the SRI.
- **The green, red, yellow ranges for the SRI scale with the weights for the various vectors; similarly the green, red, yellow thresholds for each individual vector also scale with the weights and the ranges for the individual variables.**

Software Readiness Index

Combining Vectors to a Single Index Value

Additive Model

SW Readiness Index =
w1 * SW Functionality + w2 * Operational Quality +
w3 * Known Remaining Defects +
w4 * Test Scope & Stability + w5 * Reliability

Multiplicative Model

SW Readiness Index =
SW Functionality * Operational Quality * Known Remaining Defects *
Test Scope & Stability * Reliability
Geometric average with integer weights could be used.

Recommended

Hybrid Model

SW Readiness Index =
(w1 * SW Functionality + w2 * Operational Quality +
w3* Known Remaining Defects) x
(w4 * Test Scope & Stability + w5 * Reliability)

Outline

- Business Problem
- Software Readiness Criteria
- Deploying Readiness Criteria
  - Index Table Structure
  - Visualizing Readiness Criteria- Example
  - Trending
- Summary

Index Table Structure
Measurement and Metrics

- Definition of the parameters;
- Who is responsible for measuring;
- How often to measure;
- The source of the data;
- When to start measurement

Product Quality Parameters	Acronym	Details	Metrics
Software Functionality			
Tag Implementation Accuracy	TIA	Definition: Percentage of essential tags correctly implemented & tested Responsibility: SQA Lead for the program Frequency: Monthly (end of the month) during SQA cycle Source: Tag requirement document and FT plan Trigger: FAT Entry stage (first time) or UT Completion	Tags Correctly Implemented/Planned

Vector

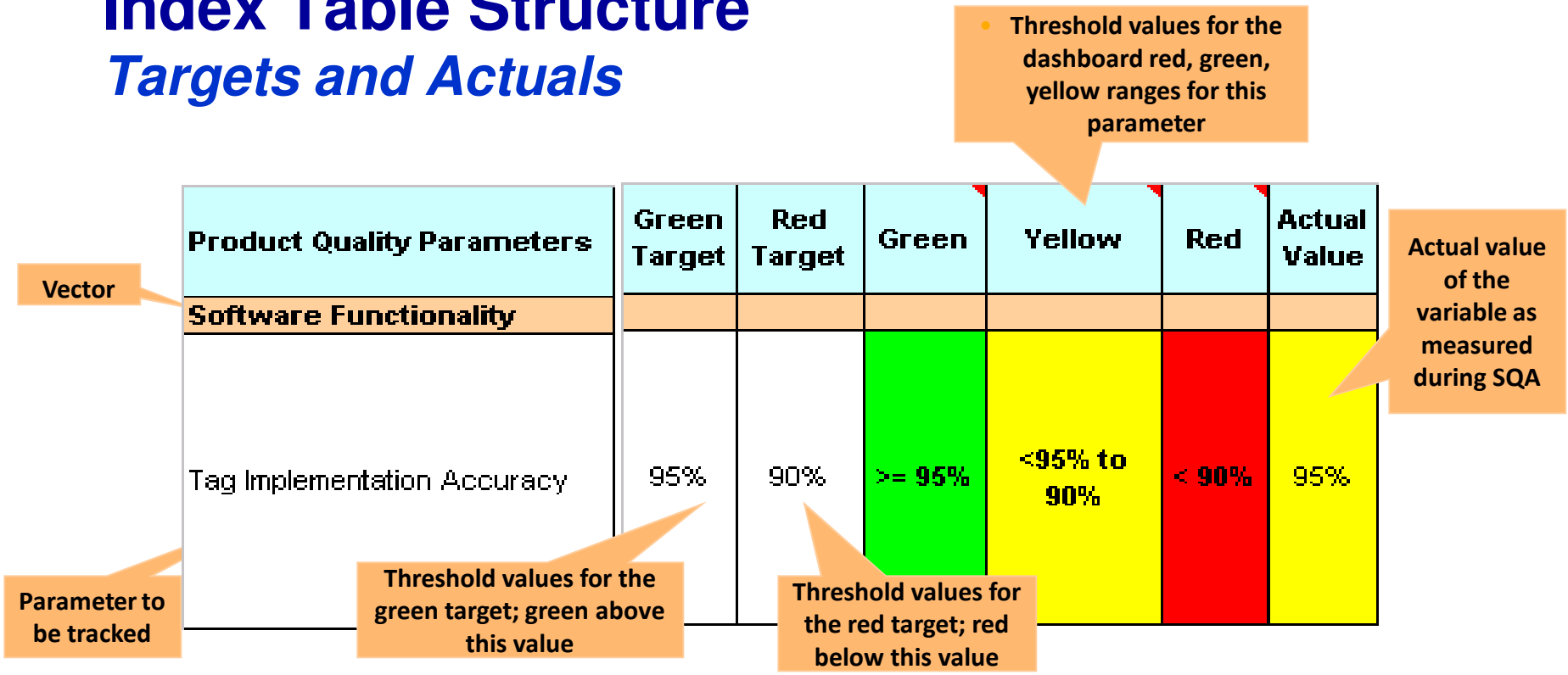
Parameter to be tracked

Metric used to compute the variable

- Each vector (e.g. software functionality) has several variables that contribute to it
- Each variable is clearly defined, assigned a owner, tracked at a specified frequency, has a well defined data source and a defined trigger

Index Table Structure

Targets and Actuals

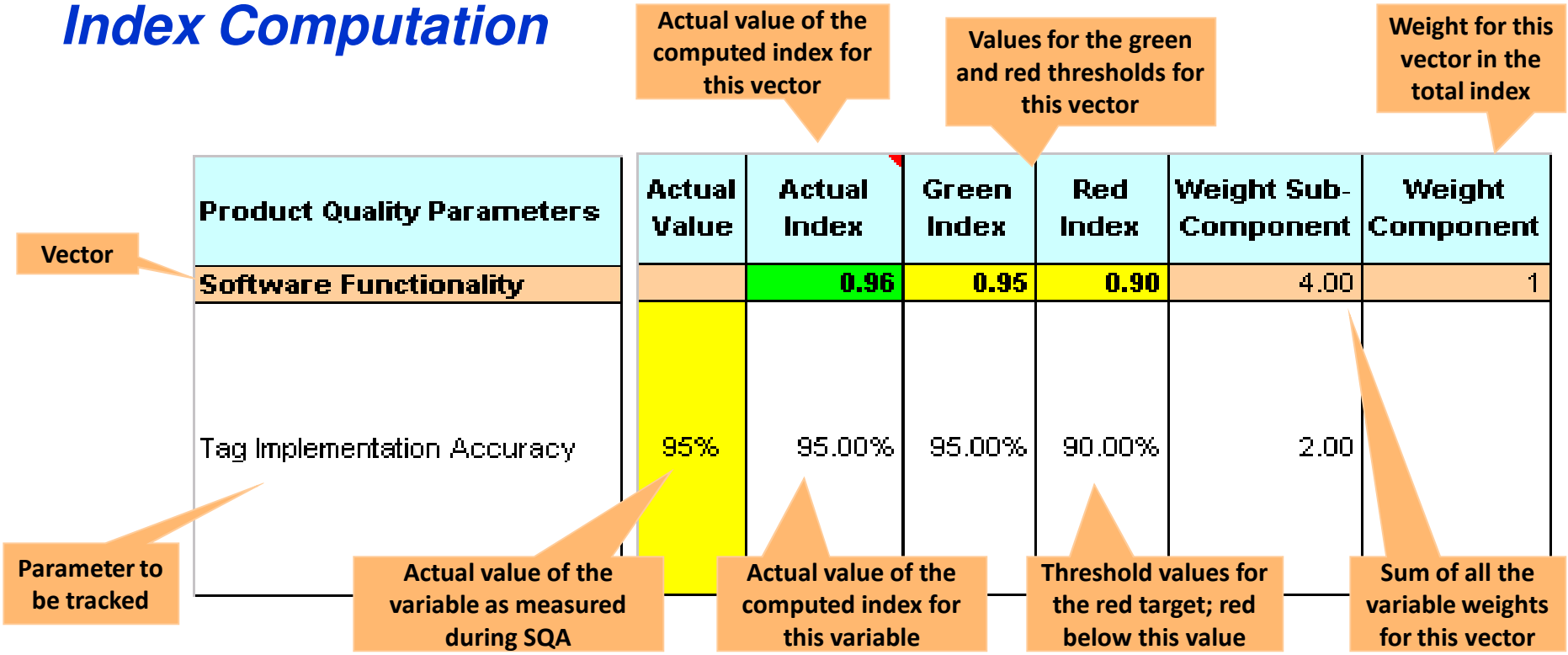


Product Quality Parameters	Green Target	Red Target	Green	Yellow	Red	Actual Value
Software Functionality						
Tag Implementation Accuracy	95%	90%	>= 95%	<95% to 90%	< 90%	95%

- The target values for red and green thresholds are established
- The actual value for each variable is tracked as defined by its metric

Index Table Structure

Index Computation

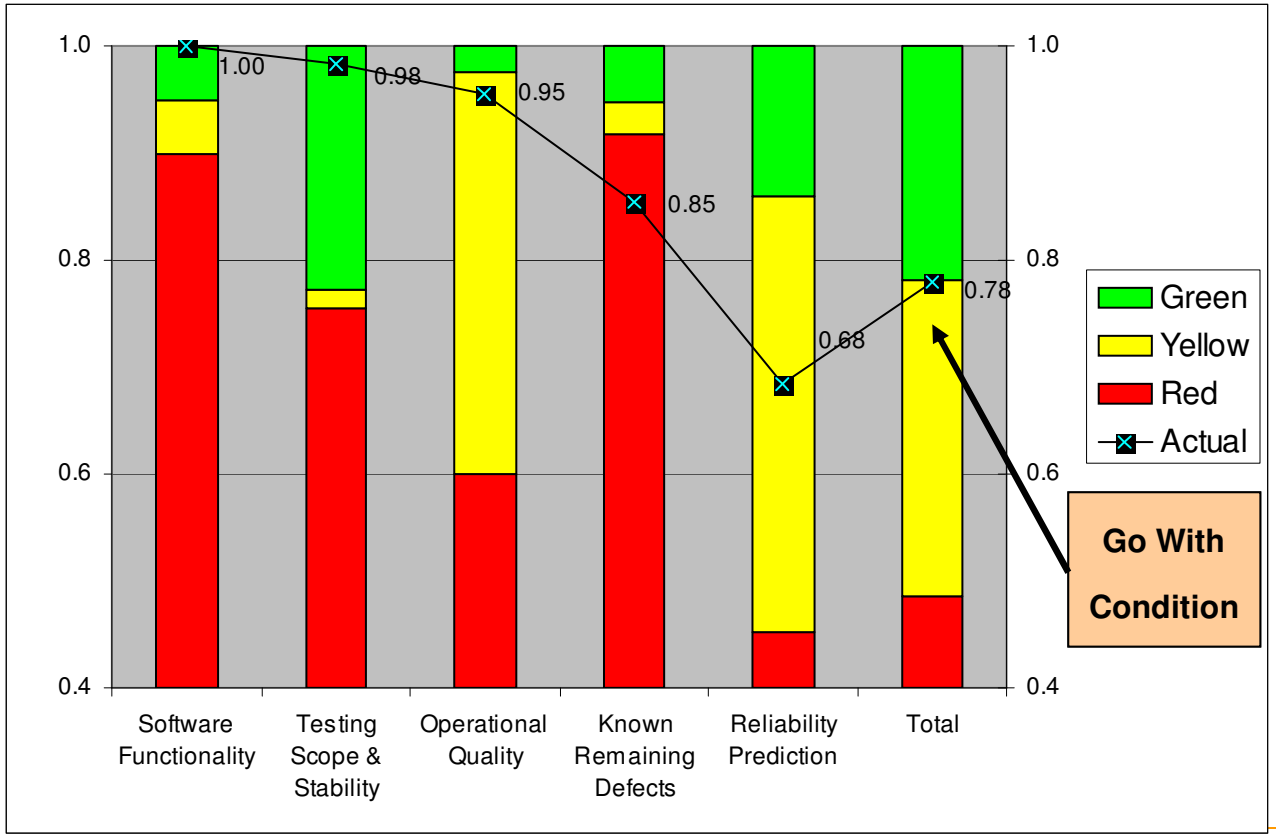


Product Quality Parameters	Actual Value	Actual Index	Green Index	Red Index	Weight Sub-Component	Weight Component
Software Functionality		**0.96**	**0.95**	**0.90**	4.00	1
Tag Implementation Accuracy	95%	95.00%	95.00%	90.00%	2.00	

- The actual index for each variable is computed from the actual value
- The actual index for each vector is computed as the weighted sum of each variable that constitutes that vector
- The red and green threshold indexes are computed from the red and green targets

Visualization Example

Vector	Actual	Red	Yellow	Green
Software Functionality	1.00	0.90	0.05	0.05
Testing Scope & Stability	0.98	0.76	0.02	0.23
Operational Quality	0.95	0.60	0.38	0.03
Known Remaining Defects	0.85	0.92	0.03	0.05
Reliability Prediction	0.68	0.45	0.41	0.14
Total	0.78	0.49	0.29	0.22



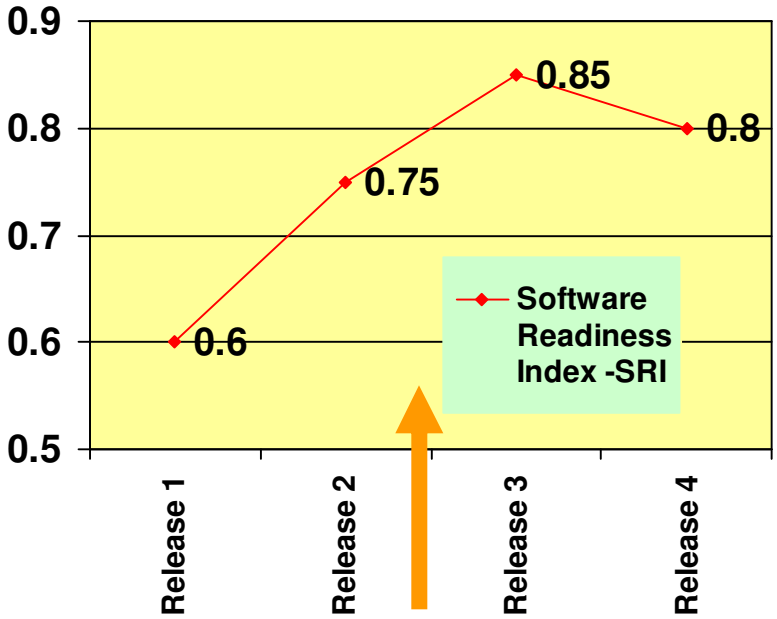
- The charts shows the red, yellow, green ranges for each vector and the actual value of the index for that vector
- The total index bar shows the target ranges for the composite index and the actual value for the composite index.
- Red = NO GO
- Green = GO
- Yellow = GO w/ CONDITIONS

MITRE

Trending

- Single Release
  - Readiness evolution of a single release
  - Parameters and targets that may get adjusted due to business decisions
- History Across Releases (same product)
  - Readiness comparison with previous releases
  - Parameters and targets that may differ from release to release
- Trend Across Product Family
  - Readiness comparison within a product family
  - Parameters and targets common to the product family (e.g. division)
- Trend Across Company
  - Readiness comparison across all products
  - Common parameters and their targets

Trending by Release and Product



Track SRI for a Product by Release to Monitor Improvement

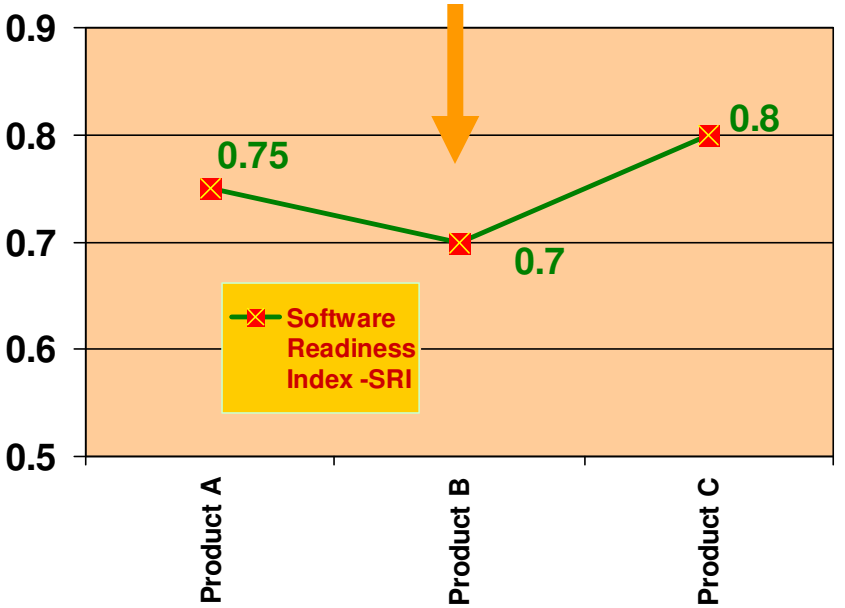
Assumes readiness measures, weights and criteria to remain constant over a product's set of releases (e.g., R1, R1.1, through Rx.y)

If the criteria change as product matures, renormalization is required.

Comparison assumes that criteria is identical across product lines (e.g., division) or across entire product areas (company)

Normalization required, if criteria changes.

Track SRI across Products to Monitor Consistency



Summary

- **Systematic approach to quantifying software goodness**
  - Setting measurable targets for each metric along a vector and tracking the status of each metric and the overall index in a dashboard.
  - During the implement phase, the dashboard provides a summary of the state of the software, the risk areas, the size of the gap in each area to aid the prioritization process.
  - Drives the Go/No Go decision.
  - Reduces subjectivity in the decision process by quantifying the individual areas in a single composite index.
- **Organizes & streamlines existing quality and reliability data and processes into a simple tool and a methodology: Agree – Track – Verify.**
  - The index and tool simplify and systematize the process of probing, monitoring and managing the software development and SQA/testing. Each metric is defined, its source is identified, and the method for computing the metric is clearly spelled out.
  - Complements and supplement other tools/techniques (e.g., DfS, DfT, DfM) since it is focused primarily on software prior to Controlled Introduction and General Availability.
- **Creates a framework that is applicable across products and across releases; the calibration improves with each release within this framework.**

Advanced/Future Topics

- Customizing SRI templates for:
  - CI vs GA
  - Major releases (including NPI)
  - Minor releases
- Trending analysis
  - Release
  - Product family
  - Cross Product; Solutions
- Validate with field data as the framework gets used across releases and products
- Integration with CMMI Lev 4/Lev 5

QUESTIONS ?