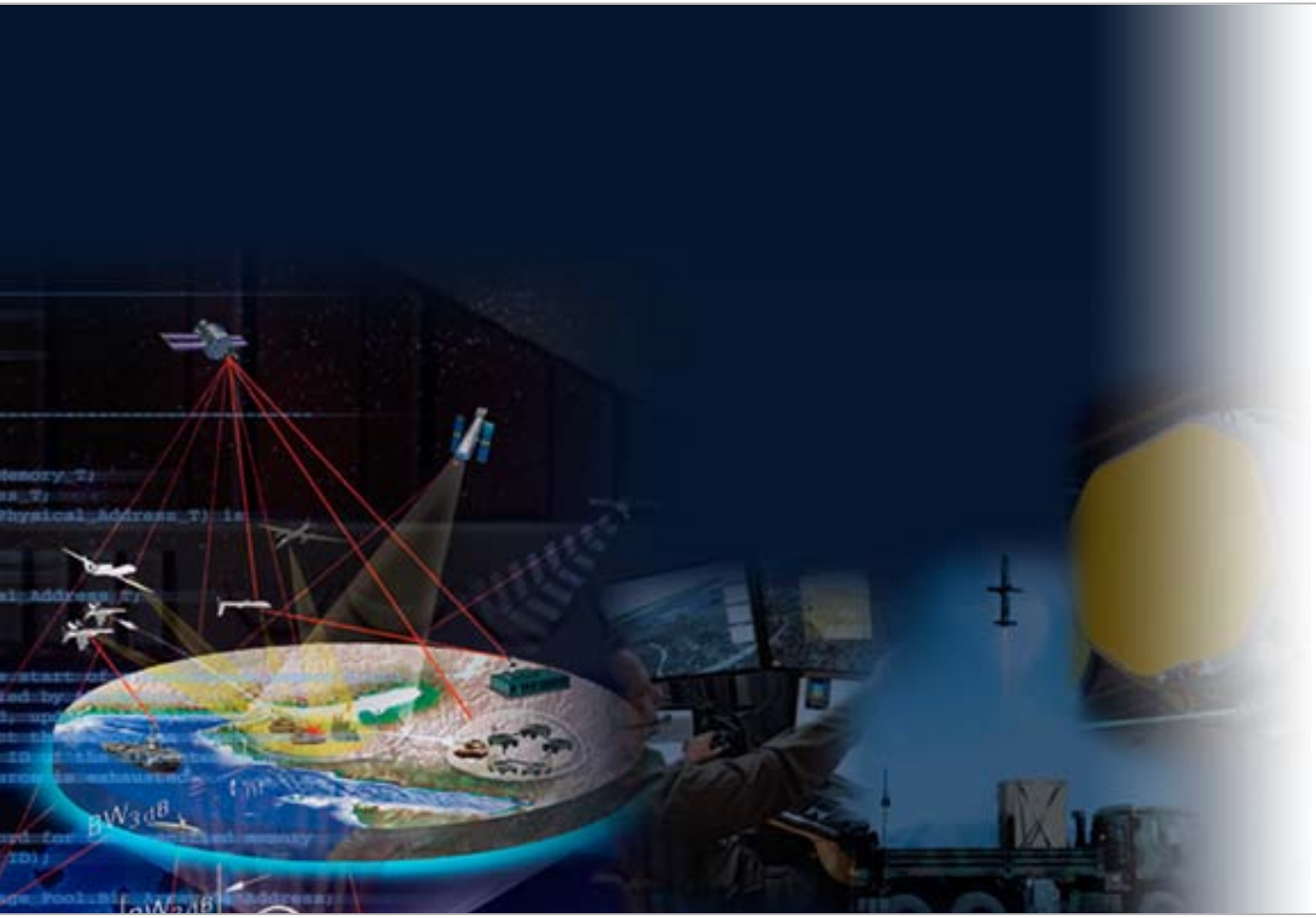
Raytheon

Customer Success Is Our Mission



Software Reliability Growth Approach

Lou Gullo

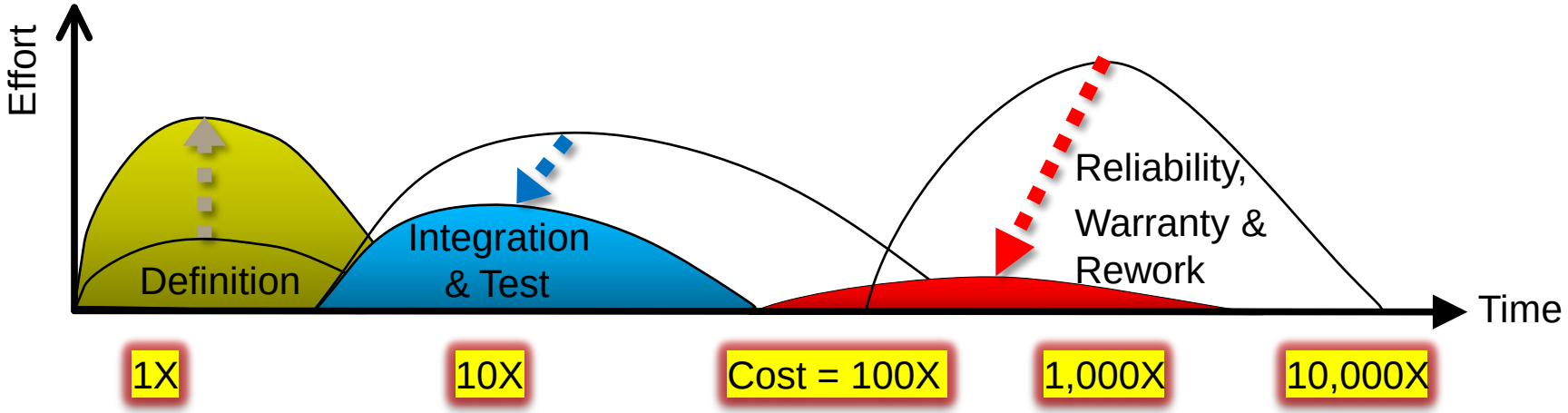
October 28, 2010

Overview

1. Value Added Benefits
2. Software Reliability Definitions
3. How SW Reliability Affects System Reliability
4. Software Reliability Prediction & Measurement Process
5. Tiered Software Development/Test Approach
6. IEEE 1633 (Steps 1, 2 and 3)
7. Sample CASRE Data
8. IOS and Ao (Steps 4 and 5)
9. Sample Results to Demonstrate Growth
10. Software Reliability Innovation – Path Forward
11. SW Reliability Reference Books

Value Added Benefits

- Reduce cost of failures later in the software development process
- Track failure trends of probabilistic conditions (e.g. race conditions) and systemic process-related issues
- Drive software design corrective actions to improve reliability results in a lower customer Total Cost of Ownership (TCO)



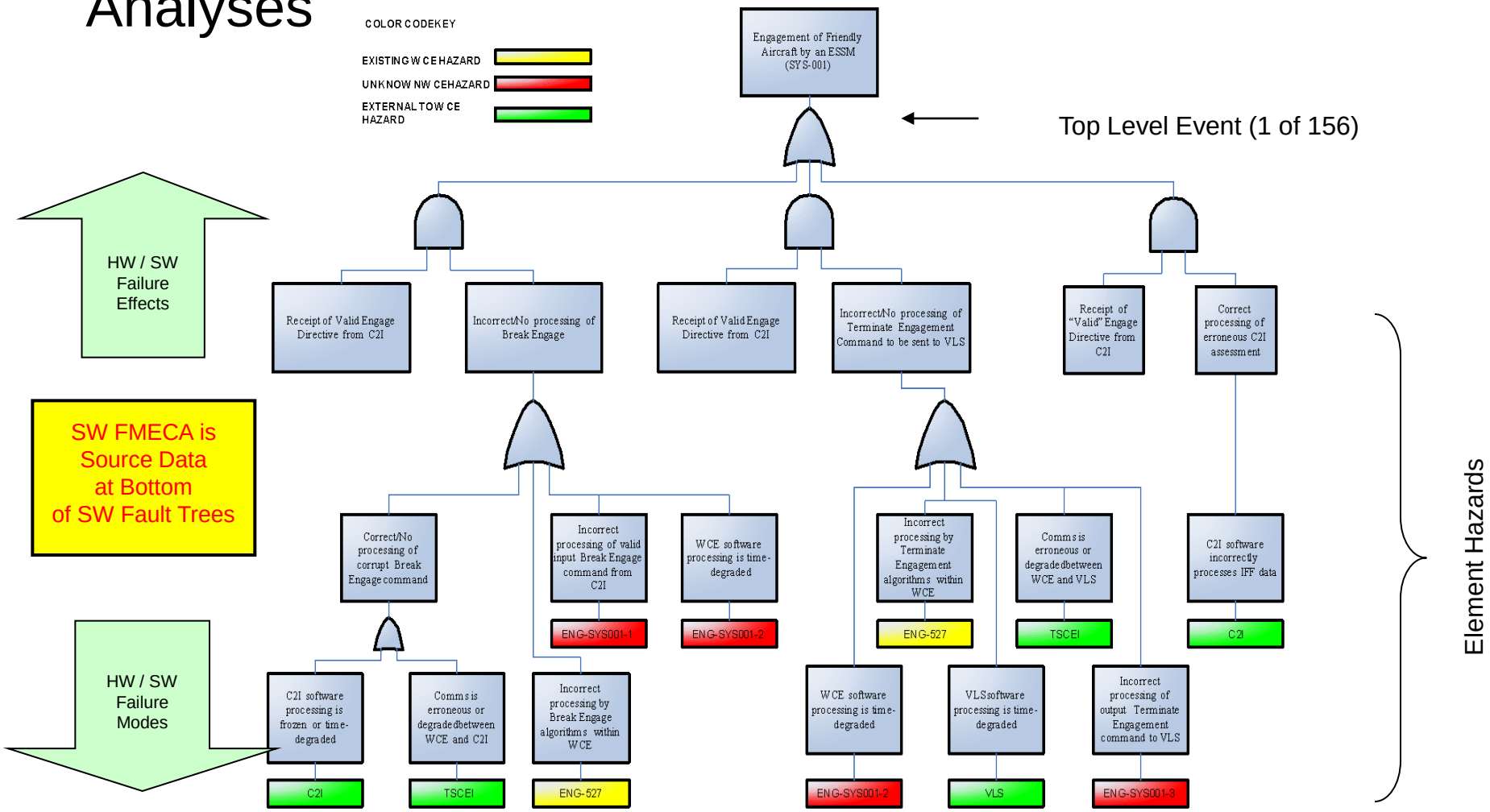
Software Reliability Definitions

- The American Institute of Aeronautics and Astronautics (AIAA) - "the application of statistical techniques to data collected during system development and operation to specify, predict, estimate, and assess the reliability of software-based systems."
- IEEE 1633:
  - **(A)** The probability that software **will not cause the failure of a system** for a specified **time** under specified **conditions**.
  - **(B)** The ability of a program to perform a required **function** under stated **conditions** for a stated period of **time**.
- IEC 62628:
  - Software Dependability - ability of the software to perform as and when required when integrated in system operation

NOTE: Software Dependability includes Software Reliability as well as other measures of software performance and capability

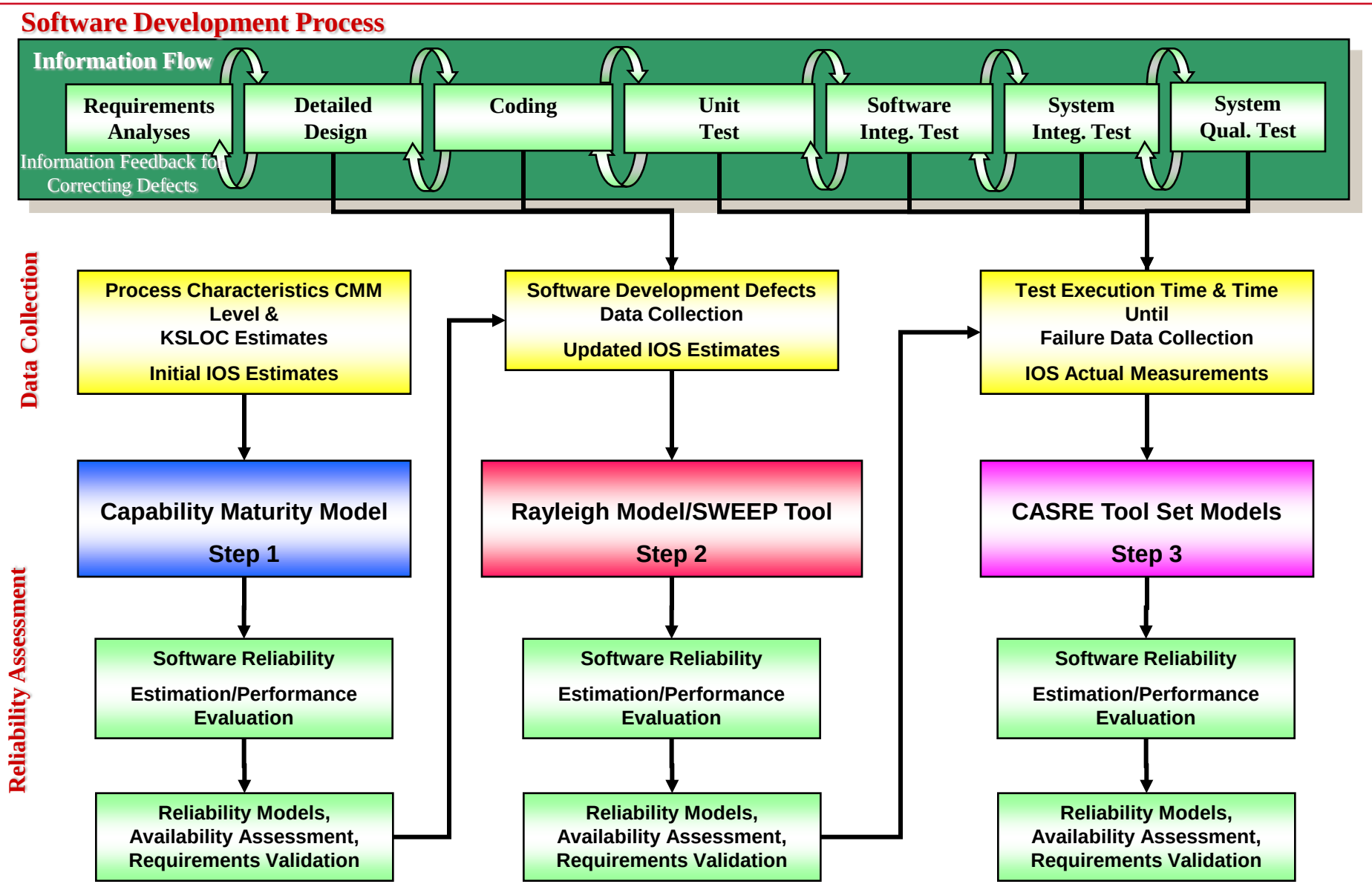
How SW Reliability Affects System Reliability Top-Level Event (System View)

- Fault Trees generated for TLE's in System Safety Analyses

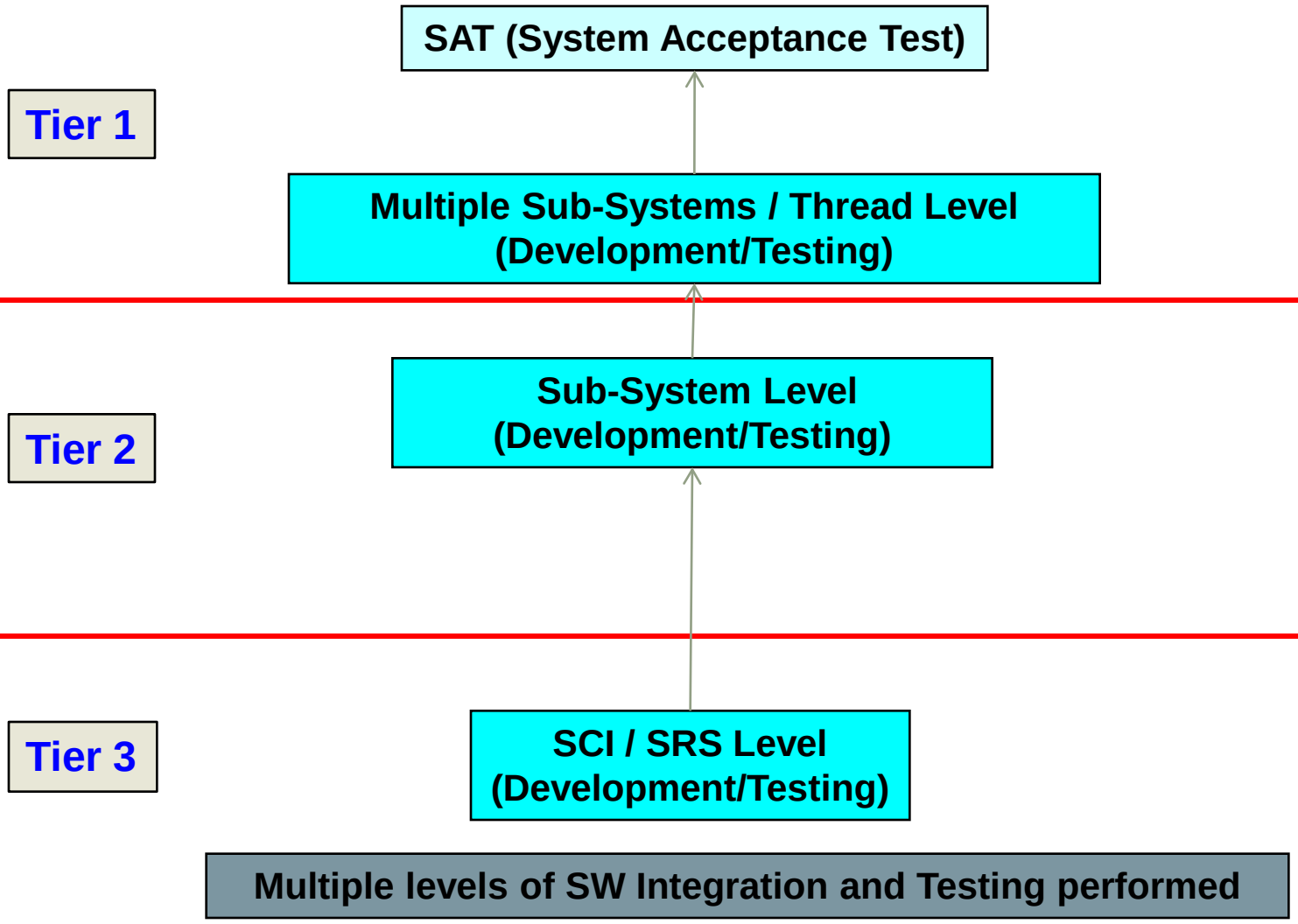


Integration of the Software Reliability into System Development Process

Raytheon



Tiered System/Software Development/Test Approach



IEEE 1633

IEEE 1633 – IEEE Recommended Practice on Software Reliability (SR)

- Developed by the IEEE Reliability Society in 2008
- Purpose of IEEE 1633
  - Promotes a systems approach to SR predictions
  - Although there are some distinctive characteristics of aerospace software, the principles of reliability are generic, and the results can be beneficial to practitioners in any industry.

How IEEE 1633 Aligns with SW Development Process

Raytheon

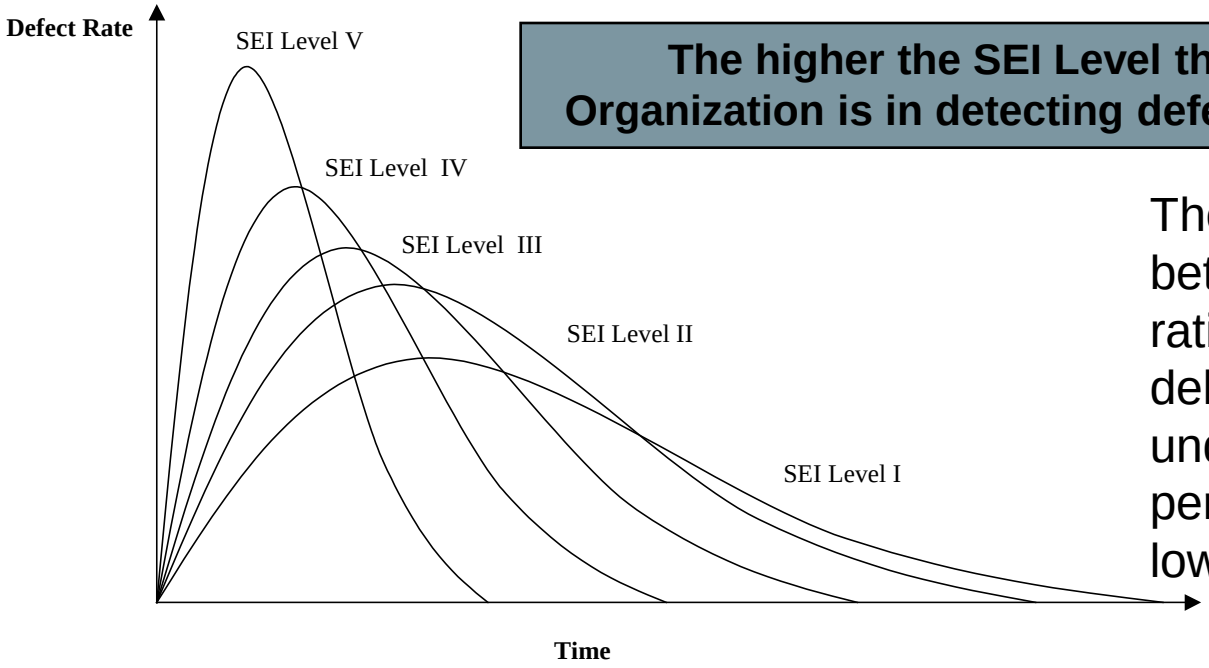
3 step process leveraging IEEE 1633:

- Step 1 – Keene Model for early software predictions
  - Weighs SEI CMMI Process Capability (e.g. CMMI Level 5 achieved by IDS) to Software Size (e.g. 10KSLOCs)
- Step 2 – SWEEP Tool for tracking growth of Software Trouble Reports (STRs) and Design Change Orders
- Step 3 – CASRE Tool for tracking failures in test

Capability Maturity Model (Keene Model) Step 1

Raytheon

- The Capability Maturity Model provides a preliminary prediction based on:
  - Estimated size of the code in KSLOC
  - Software Engineering Institute's (SEI) Capability Maturity Model (CMM) rating of the software developer
  - The assertion is that the software process capability is a predictor of the latent faults shipped with the code.



The higher the SEI Level the more efficient and Organization is in detecting defects early in development

The better the process, the better the process capability ratings and the better the delivered code, developed under that process, will perform….defects will be lower.

Keene Process-Based (a priori) SW Reliability Model (CMM Model) Inputs

Raytheon

- This model provides MTBF and Ao predictions for each Ensemble. These were used to confirm that the Ao requirements were reachable.

- These predictions are somewhat approximate, and so further refinement is needed in the later stages of the process.

- Process Capability (SEI Level)
  - Development Organization
  - Maintaining Organization
- Code Extent (SLOC)
- Exponential growth to a plateau level
- Historical Factors
  - R growth profile
  - Usage level
  - Fault latency
  - % Severity 1 and 2 failures
  - Fault activation rate

PROCESS INPUT PARAMETERS			
Data Required	**Inputs**	**Range**	**Input Instructions:**
KSLOCs	**441.7**	>0	**Number of 1,000 lines of source code (KSLOCs).**
SEI Level - Develp	**3**	1-5	**SEI Level factor (1-5).**
SEI Level - Maint.	**3**	1-5	**SEI Level factor (1-5).**
Months to maturity	**20**	<=48	**Number of months to maturity or failure rate plateau.**
Use hrs/week	**168**	<=168	**Number of operational hours per week.**
% Fault Activation	**100**	<=100	**Average %population exhibiting fault activation.**
Fault Latency	**2**	>=1	**Ave. # of fault reoccurrences/failing-site until corrected.**
% Sev 1&2 Fail	**10**	<=100	**Average % severity 1 and 2 or % countable failures.**
MTTR	**10**	>0	**Average time to restore system (minutes)**

Useful to Flowdown or Decompose Requirements to Lower Tiers

SWEEP (Software Error Estimation Program) Capabilities - Step 2

Raytheon

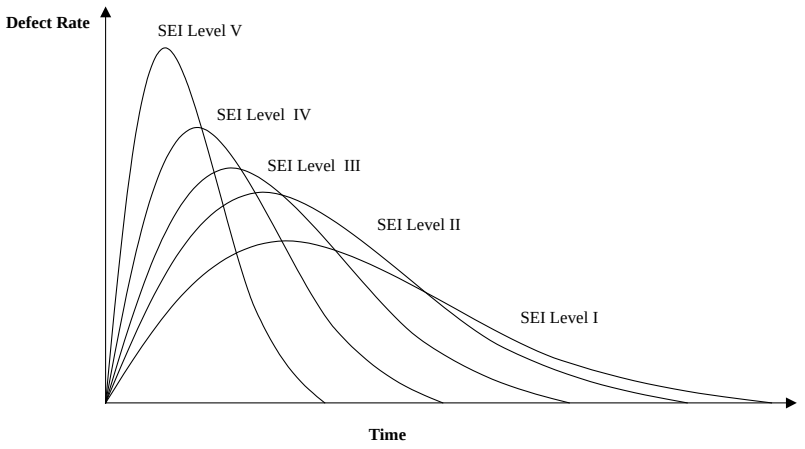
- **The SWEEP tool enables you to:**
  - **Predict and track the rate at which defects will be found**
  - **Predict the latent defect content of software products.**
  - **Analyze estimated errors injected in each phase of the software development cycle**
  - **Determine the detection effectiveness and leakage of errors to subsequent phases.**
  - **Measure percentage of critical failures that feedback into the Keene model**

- **SWEEP Data Collection**
  - **Data is typically collected using Software Trouble Reports (STR)**
  - **Data can be organized by development phase or time increments.**

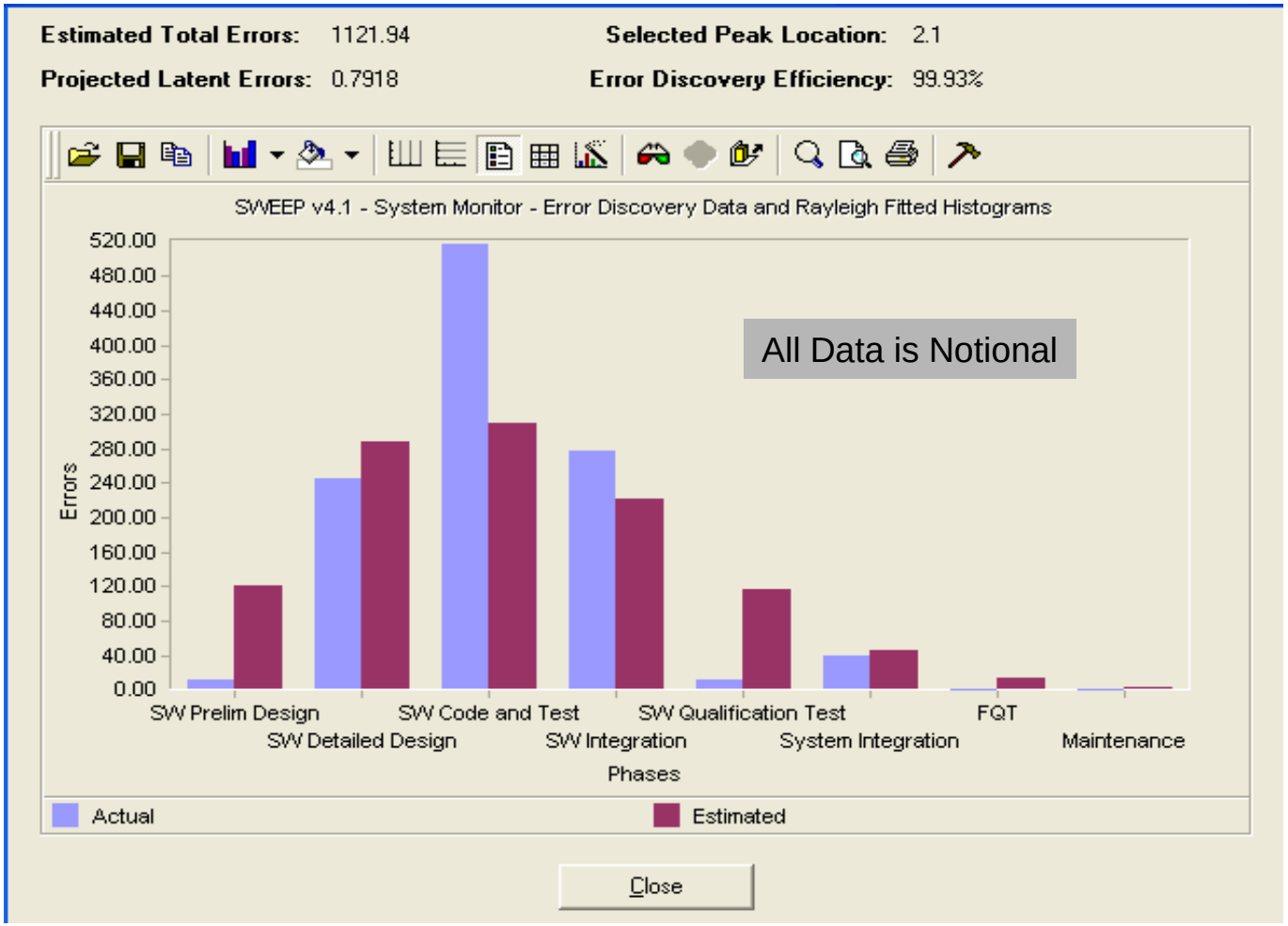
SWEEP Model Theory

- The SWEEP Tool uses the Rayleigh Model based on the Rayleigh Distribution
- The Rayleigh Distribution is a special case of the Weibull Distribution
- Model Assumptions
  - The defect rate observed during the development process is positively correlated with the defect rate in the field (The more area under the curve, the higher the field defect rate).
  - Given the same error injection rate, if more defects are discovered and removed earlier, fewer will remain in later stages.
- Reference Reading Metrics and Models in Software Quality Engineering, Addison Wesley Publishing

  (Author: Stephen Kan)



SWEEP Tool Output (Sample)



CASRE (Computer Aided Software Reliability Estimation) Step 3

Raytheon

- CASRE (Computer Aided Software Reliability Estimation) is a software reliability measurement tool that runs in the Microsoft Windows environment…developed by Allen Nikora at JPL.
- The modeling and analysis capabilities of CASRE are provided by the public-domain software reliability package SMERFS (Statistical Modeling and Estimation of Reliability Functions for Software).
- In implementing CASRE, the original SMERFS user interface has been discarded, and the SMERFS modeling libraries have been linked into the user interface developed for CASRE.
- CASRE is typically applied starting after unit test and continuing through system test, acceptance test, and operations.
- You should only apply CASRE to modules for which you expect to see at least 40 or 50 failures. If you expect to see fewer failures, you may reduce the accuracy of your estimates.
- Experience shows that at the start of software test, modules having more than about 2000 source lines of executable code will tend to have enough faults to produce at least 40 to 50 failures.

CASRE Data Input

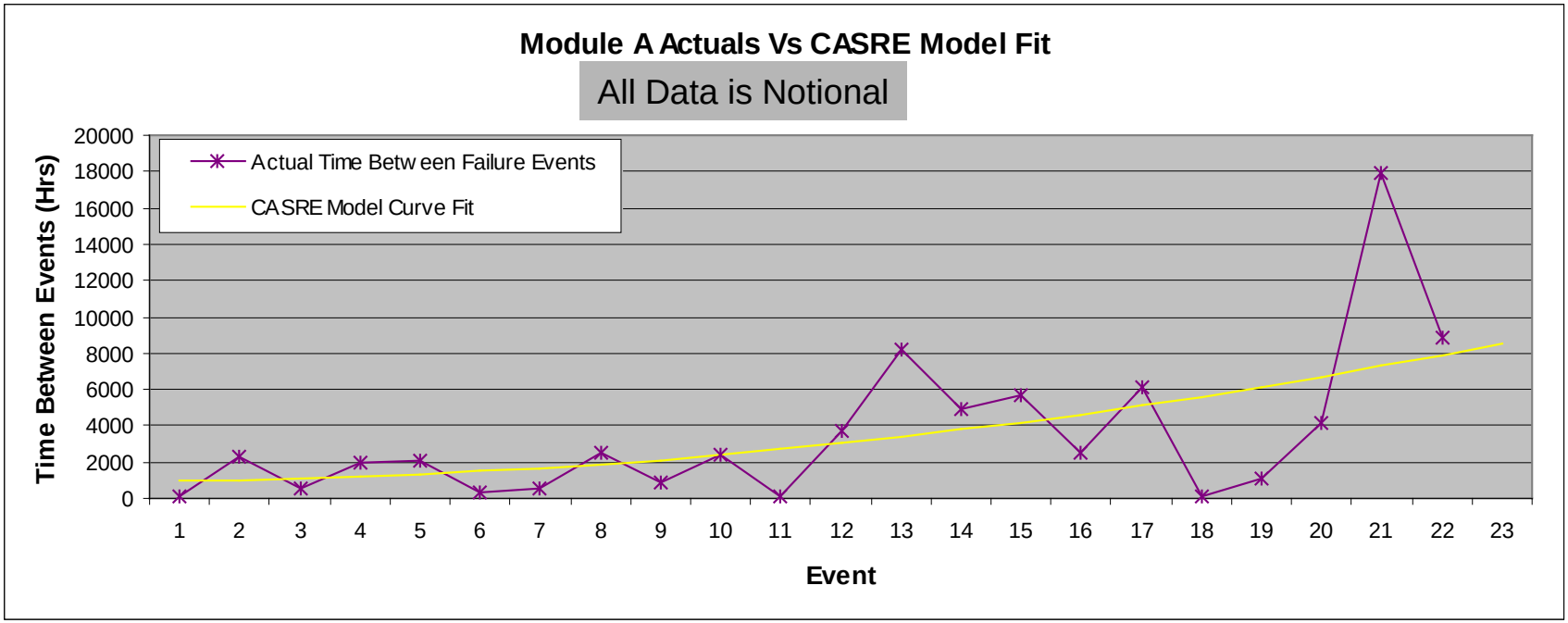
Two types of data files CASRE can accept

1. Times between successive failures.
   - Error Number (integer)
   - Time since last failure (floating point)
   - Error Severity (integer)
2. Failure counts per test interval and test interval length.
   - Interval Number
   - Number of Errors
   - Interval Length
   - Error Severity

Information to enhance the accuracy of model predictions:

- Date and time at which each failure was found, and the test interval during which the software was run that produced that failure.
- Date and time at which the testing method changed. The reason for this is that the perceived reliability of the system depends on how it is executed.
- Date and time at which the test environment changed. The reason for collecting this information is to more accurately characterize the length of a test interval.
- Date and time at which the software being tested changes significantly.
- Severity of each failure.

Sample CASRE Data Models and Output

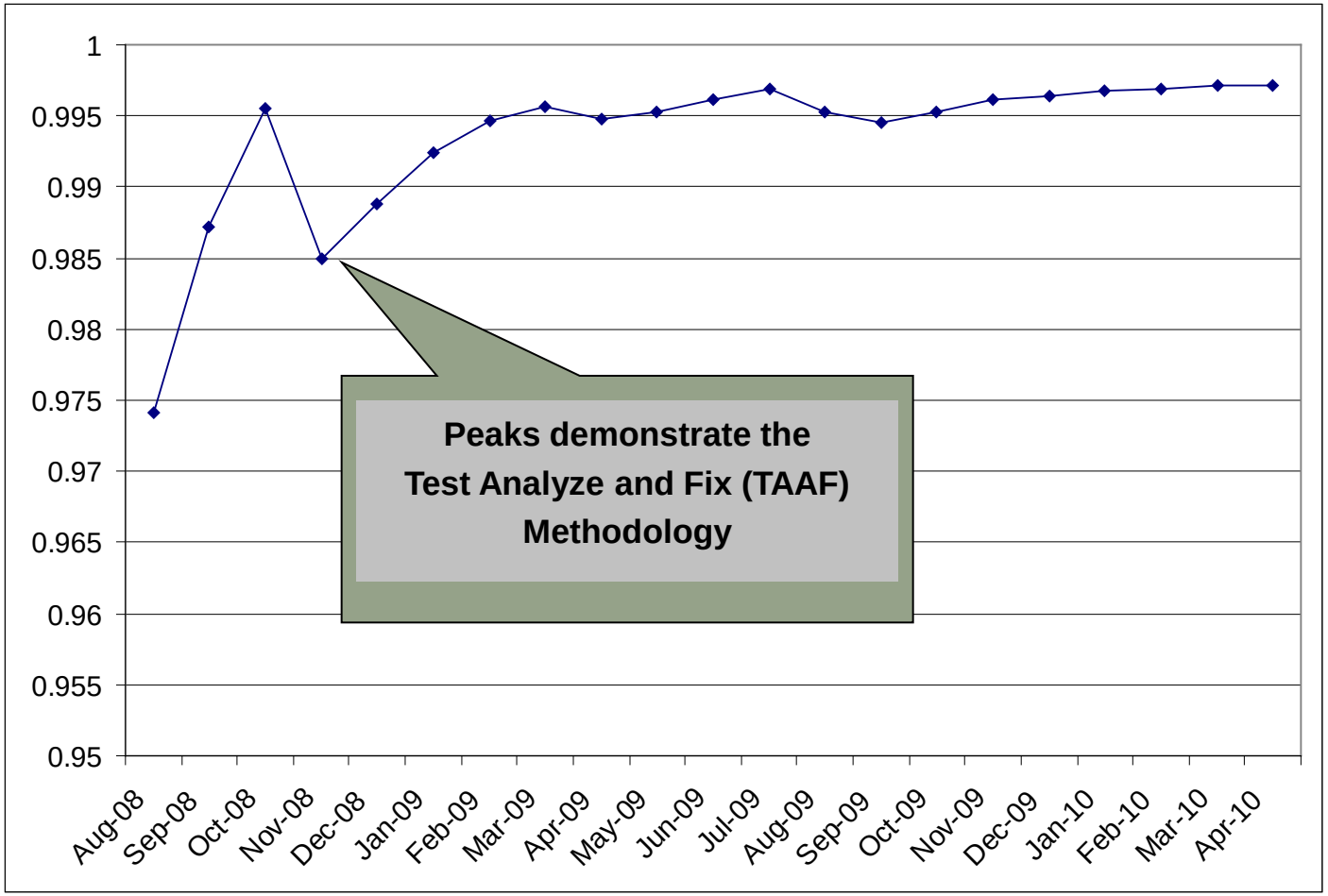


Times Between Failures Models
Geometric
Jelinski-Moranda
Littlewood-Verrall Linear
Littlewood-Verrall Quadratic*
Musa Basic
Musa-Okumoto *

Failure Counts Models
Generalized Poisson
Schneidewind
Shick-Wolverton
Yamada S-shaped

Combination Models
Dynamically-Weighted Linear Combination (DLC/S/4)
Equally-Weighted Linear Combination (ELC)
Median-Weighted Linear Combination (MLC)
Unequally-Weighted Linear Combination (ULC)

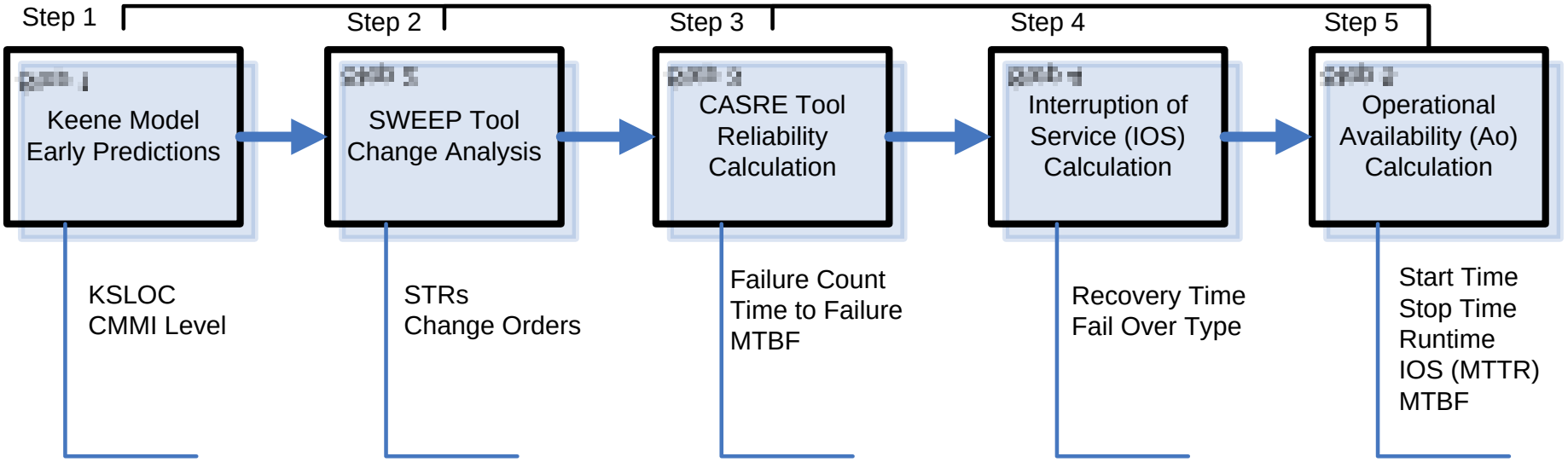
Sample Results to Demonstrate Growth



Demonstrates Traditional Growth Curve

IOS and Ao Calculations
Added Steps 4 and 5

Raytheon



Closed Loop System with Step 5 Feedback to Steps 1 -3

Software Reliability Innovation – Path Forward

Initiatives to Increase SW Reliability Growth and Accelerate Deliveries of Mature / Dependable SW to the Warfighter:

- Continue to develop capabilities to detect software stress points earlier in the software life cycle
- Continue decreasing SW fault density significantly during SW production, prior to testing
- Continue improvement of software reliability growth testing processes and tools
- Continue to develop new standards or sustain existing standards (e.g. IEEE 1633 and IEC 62628)
- Develop more rigorous software development processes

Raytheon Approach Accommodates Increased SW Complexity & Reliability

SW Reliability Reference Books and Standards

Raytheon

- Metrics and Models in Software Quality Engineering, Stephen Kan, Addison Wesley Publishing
- Handbook of Software Reliability Engineering, Michael Lyu, McGraw Hill Publishing
- Software Reliability: Measurement, Prediction, Application, John D. Musa, Anthony Iannino, and Kazuhira Okumoto, McGraw-Hill Book Company
- IEEE 1633: Recommended Practice on Software Reliability (SR)
- IEC 62628: Guidance on Software Aspects of Dependability

Biography

– Lou Gullo, Raytheon, Integrated Defense Systems, Whole Life Engineering Directorate. Leader on several Enterprise-wide Engineering Council-sponsored special projects including software reliability methods and the automation of electrical stress analysis methods. 30 years experience in military, space and commercial programs. Retired US Army Lieutenant Colonel. Senior Member of the IEEE. IEEE Reliability Society Standards Committee Chair. Member of the Reliability and Maintainability Symposium (RAMS) Management Committee.

Louis J Gullo
Sr Principal Systems Engineer
Lou.Gullo@Raytheon.com
401-842-4139