

Strategy for Applying Software Reliability Growth Models (SRGM) to DOD Systems

Andy Long

Office of the Secretary of Defense (OSD),

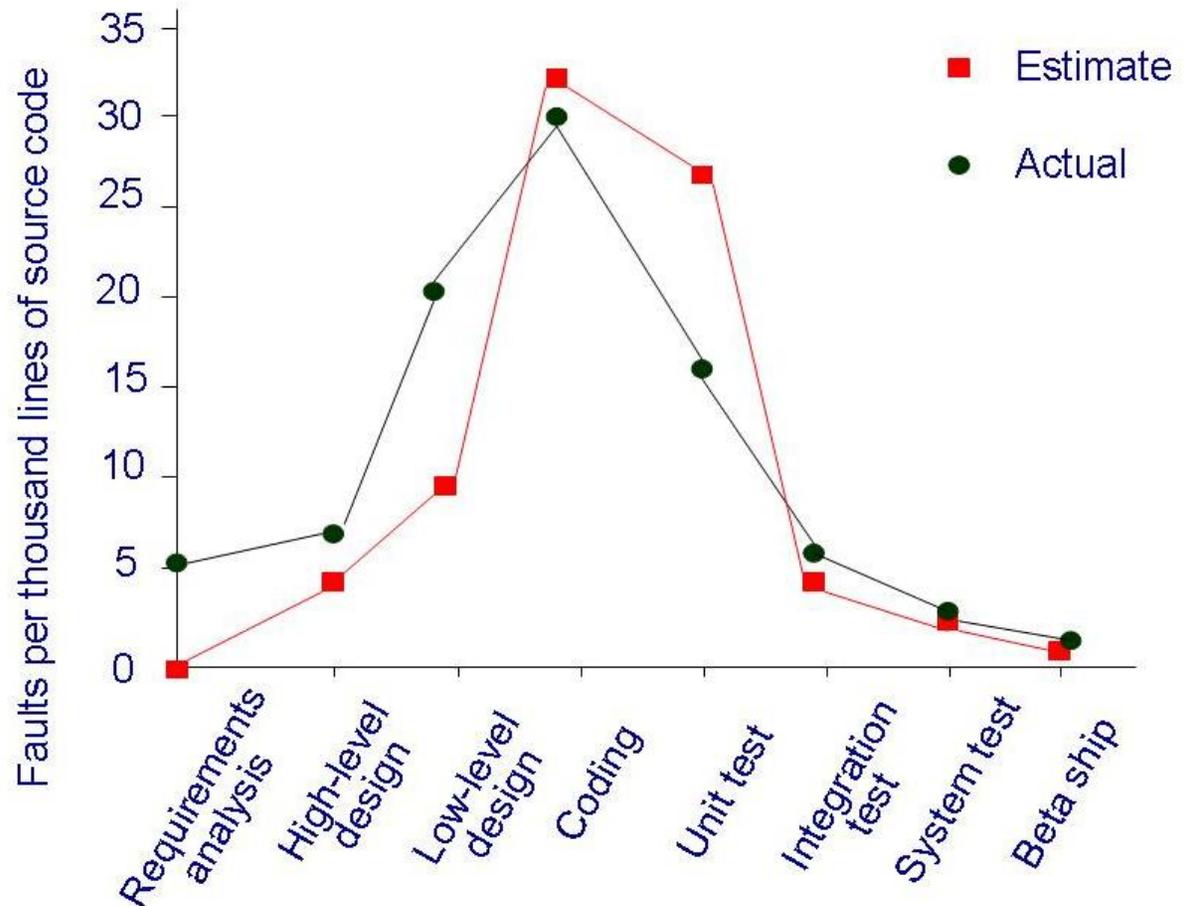
Director, Operational Test & Evaluation (DOT&E)/Science Advisor

Scientific Research Corporation

January 2013

Purpose

Develop methodology for effectively applying software reliability growth models (SRGM) to track and predict software reliability growth by applying categorizations of software usage in DoD systems.



"All Models are Wrong - Some are Useful."

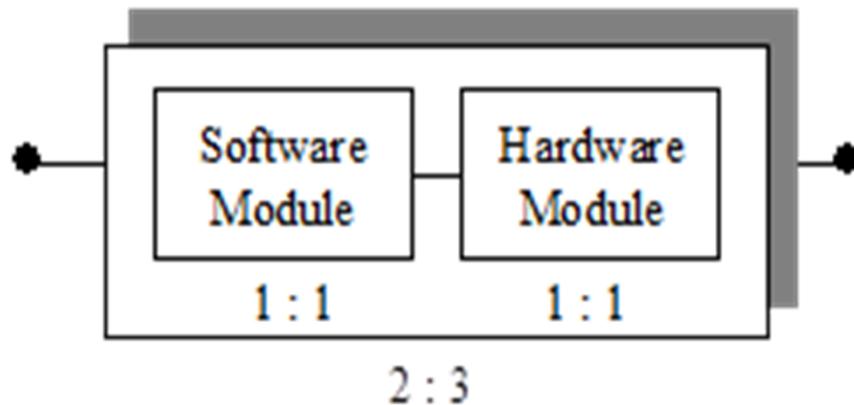
George E. P. Box

Background

- Software issues are common in programs of all types, and can often result in systems being rated as not operationally effective or not operationally suitable.
 - Between 1997 and 2012, 12% of systems were found not reliable in IOT&E as result of failure modes attributed to software.
- Tasking from OSD/Dr. Catherine Warner, DOT&E Science Advisor:
 - Improve reliability tracking and growth modeling for all programs that incorporate software with focus on software centric systems, by allowing Program Managers (PMs) to:
 - better determine whether the current system state is sufficient for a product release;
 - determine if the release date should be deferred if existing software issues require resolution and corrective actions;
 - Make more informed tradeoffs during earlier development phases to optimize reliability within program constraints.

Categorizations of Software Usage Systems:

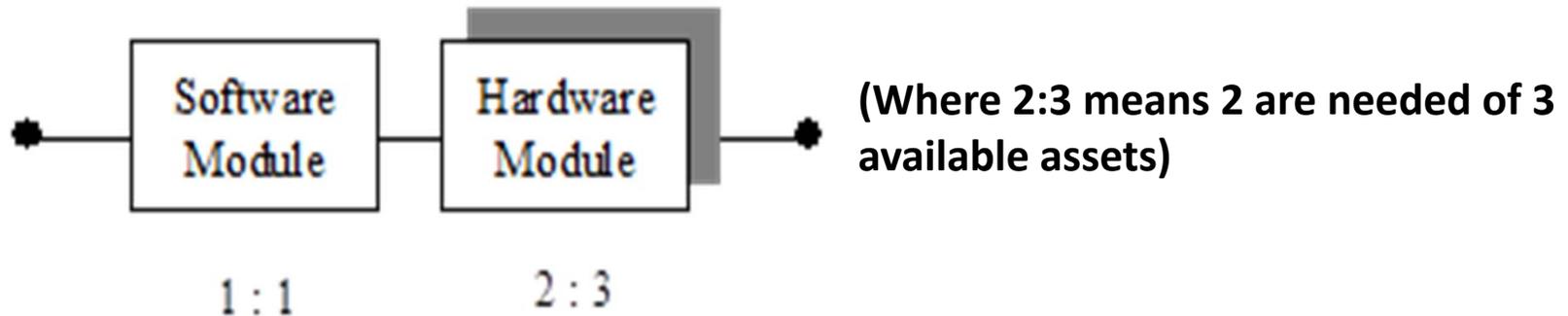
Hybrid



(Where 2:3 means 2 are needed of 3 available assets)

- Modeled using traditional reliability growth modeling techniques for HW centric systems, because functionality results from HW and SW working together.
- Difficult to separate SW related failure from the function it relates to and the HW it controls
- Reliability modeling of these systems is well-described by log-Poisson Non-Homogeneous Poisson Process (NHPP) models in the relation to time (e.g., the AMSAA Maturity Projection Model) due to their simplicity, convenience, and tractability.

Categorizations of Software Usage Systems: Software Centric and Space Systems

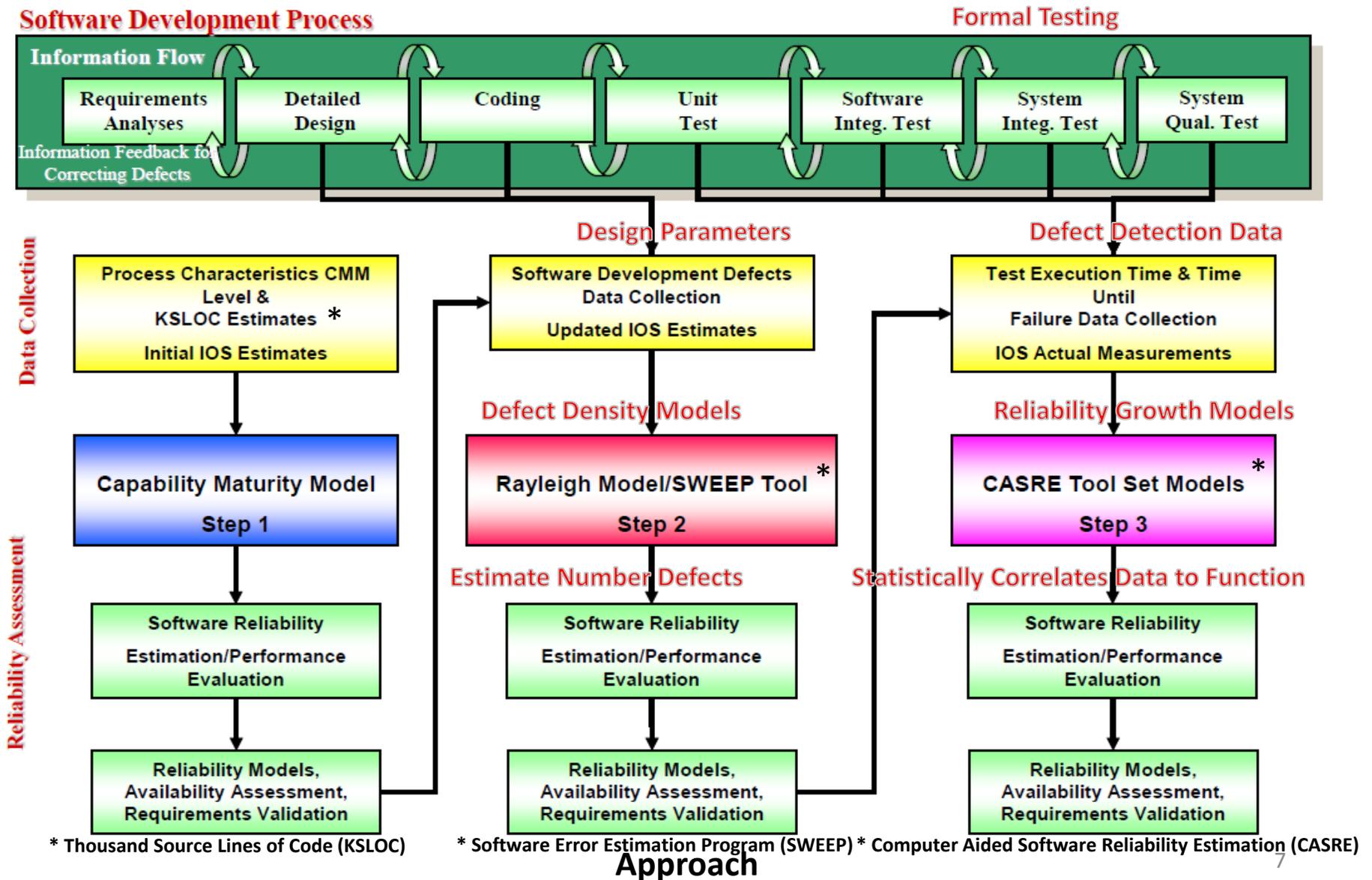


- Use similar probabilistic models as do hybrid systems, but differ in that SRGM use design faults as their source for failures.
 - Design faults are typically usage dependent and time independent.
- SRGM specify the general form of the dependence of the failure process on the principal factors that affect it: fault introduction, fault removal and the operational environment.
- Space real-time data networks are very comparable to software centric systems for reliability modeling.
 - Satellite ground control centers use the same hardware and software infrastructures as conventional information technology applications.

Approach

- **Leverage existing work to model software reliability**
 - IEEE/AIAA P1633™ 2008, Recommended Practice on Software Reliability
 - Promotes a systems approach to software reliability predictions
 - Provides a sequence of steps for assessing software reliability that is independent of specific development processes and SWRGMs.
- **Use readily available tools: "defect density" models; "software reliability growth" models to estimate reliability during design and integration testing**
- **Focus is on reliability growth modeling for software centric and space systems**

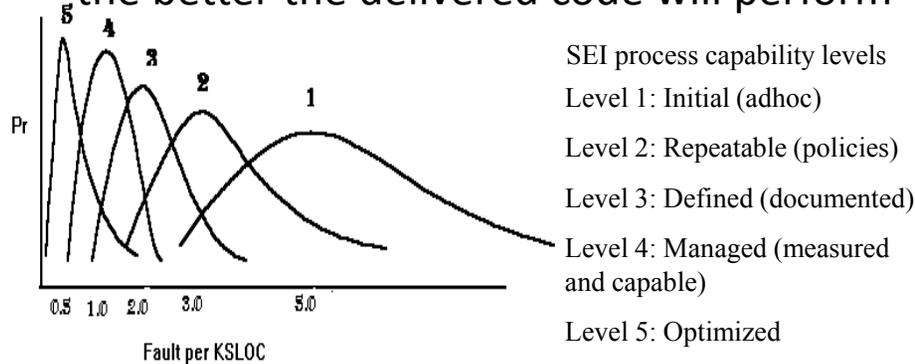
Integration of IEEE 1633 into Development Process



Step 1: Capability Maturity Model (Keene Model)

Theory

- The better the process capability ratings, the better the delivered code will perform



Data

- SEI level for the developing organization,
- Number of months for the software failure rate to stabilize,
- Number of failure replications expected prior to fault removal,
- Fault activation rate,
- Percentage of all failures that are critical failures, and the
- Projected run time per month.

Assumption

- SEI Development Level Correlate to Latent Faults

Enables

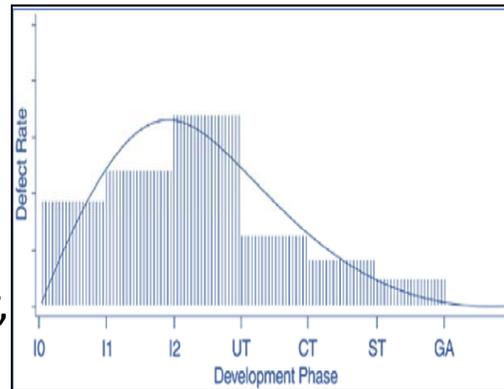
- Transforms the latent fault density into in an exponential reliability growth curve over time to estimate reliability

Useful to Flowdown or Decompose Requirements to Lower Tiers

Step 2: Rayleigh Models/Software Error Estimation Program (SWEEP) Capabilities

Theory

- Rayleigh Model: Weibull distribution with shape parameter, $m = 2$.



Data

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

Assumptions

- The defect rate observed during the development process is positively correlated with the defect rate in the field.
- Given the same error injection rate, if more defects are discovered and removed earlier, fewer will remain in later stages.

Enables

- Predicting and tracking the rate at which defects will be found;
- Predicting the latent defect content of software products;
- Analyzing estimated errors injected in each phase of the software development cycle;
- Measuring percentage of critical failures

Useful To Predict the Software's Latent Fault Density

Step 3: Computer Aided Software Reliability Estimation (CASRE)

- Software reliability measurement tool that runs in the Microsoft Windows environment...developed by Dr. Allen Nikora at JPL.
- The modeling and analysis capabilities are provided by the public-domain software reliability package, Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS).
 - 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 fielding.
- CASRE should be applied to modules for which you expect to see at least 40 or 50 failures--Experience shows that at the start of software test, modules having more than 2K source lines of code (2KSLOC) will tend to have enough faults to produce at least 40 to 50 failures.

CASRE Reliability Growth Models

- **The exponential model is regarded as the basic form of SWRGM**
 - Since the early 1970s, more than a hundred models have been proposed
 - Few have been tested in practical environments; even fewer are in use.
- **Models currently being used by type, form, and assumptions are:**

Model Types:	CASRE Model Forms:	Assumptions:
I. Predict Time Between Failures	Execution Time Models	
Input Data (Typically from Problem Reports): <ul style="list-style-type: none"> • Error Number (integer) • Time since last failure (floating point) • Error Severity (integer) Parameters (for most models): $\mu(t)=\alpha F(t)$, wh: α is the expected no. of defects and $F(t)$ is a CDF; $F(0)=0$, $F(\infty)=1$, $\mu(\infty)=\alpha$	Geometric	<ul style="list-style-type: none"> • There are N unknown software faults at the start of testing • Failures occur randomly • All faults contribute equally to failure • Fix time is negligibly small • Fix is perfect for each fault
	Jelinski-Moranda	
	Littlewood-Verrall Linear	
	Littlewood-Verrall Quadratic	
	Musa Basic	
	Musa-Okumoto	
	Non-homogeneous Poisson (NHPP)	
II. Predict Failures Count	Time Intervals Models	
Input Data (Typically from Problem Reports): <ul style="list-style-type: none"> • Interval Number • Number of Errors • Interval Length • Error Severity Parameters (for most models): $\mu(t)=\alpha F(t)$, wh: α is the expected no. of defects and $F(t)$ is a CDF; $F(0)=0$, $F(\infty)=1$, $\mu(\infty)=\alpha$	Generalized Poisson (interval weight optional)	<ul style="list-style-type: none"> • Testing intervals are independent of each other • Testing during intervals is reasonably homogeneous • Number of defects detected is independent of each other
	Schneidewind	
	Schneidewind (combines 1 st s-1 intervals)	
	Shick-Wolverton	
	Yamada S-shaped	
	Non-homogeneous Poisson (NHPP)	

CASRE Model Selection Rules for Picking The “Best Fit Model”

Rule	Description
1. Estimate the model parameters using statistical techniques such as maximum likelihood or least-squares methods.	The maximum likelihood technique solves for optimal parameter values. The least squares method solves for parameter values that best fit a curve to the data.
2. Goodness-of-fit Test (i.e., Kolmogorov-Smirnov (K-S) test or Chi-Square) on the model	Tests the null hypothesis that the appropriate model at an alpha confidence level of 5% fits the data. A “Yes” response indicates the model does fit the data.
3. Prequential Likelihood Ratio (PLR)	Given a prior belief that either model A or B is equally appropriate, the PLR defines the likelihood that model A will produce more accurate estimates than model B.
4. Model Bias	Quantifies the extent to which a model consistently makes predictions that are larger or smaller than those observed.
5. Model Bias Trend	Determine extent bias changes over time

Software Reliability Assessment for an Example System

- The modeled system is a navigation system. The following software modules of varying lines of code and critical failure events were analyzed using CASRE:

Module	KSLOC	# Events
A	203.9	117
B	43.3	13
C	74.8	18
D	145.9	29

- Data modeled here are for SW centric systems. Peterson et alii modeled five software modules using both Time Between Failures and Failures Count models. They conclude that “There was not enough data to support a high significance level of fit for the failure count models, so without further analysis, the inter-fail time models (Execution Time Models) are preferred by inspection.”
- So, this example will be limited to reliability growth modeling using only the time between failure models.**

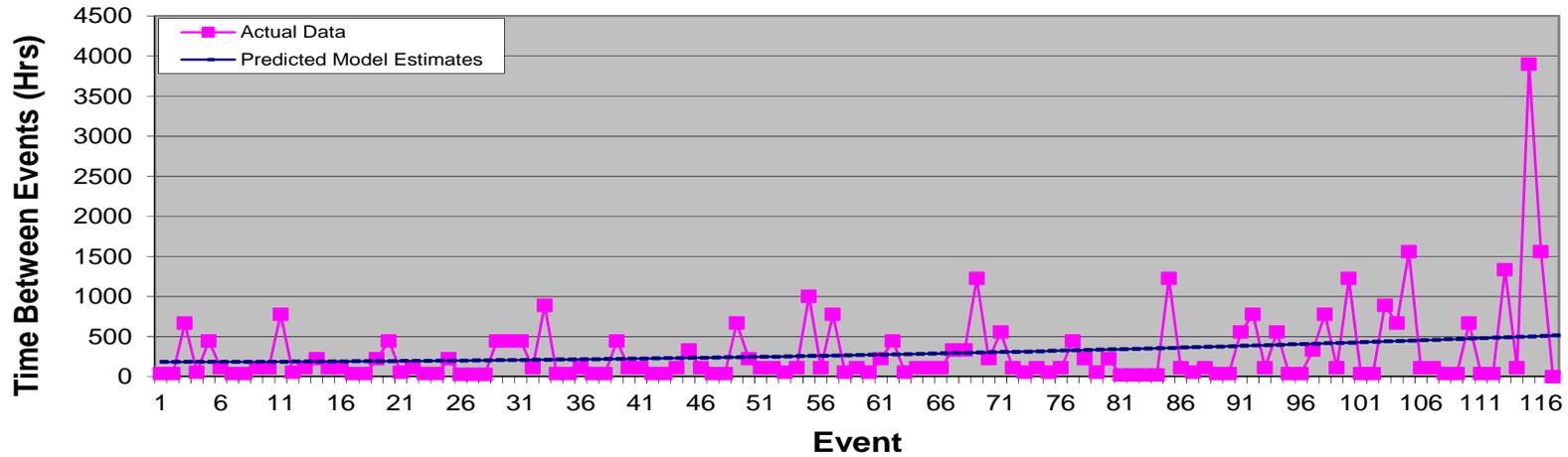
Goodness of Fit and Model Ranking Results

Module	Model Name	KS Distance	5. % Fit	Significance (%)	Rank	Estimated Time To Failure at Last Interval (Hrs.) (Failure Rate For Critical Priority 1,2, & 3 Failures)
A	Quadratic LV	0.0930	Yes	25.8%	1	513
	Linear LV	0.1019	Yes	16.4%	2	429
B	Quadratic LV	0.2438	Yes	36.4%	1	2053
	Musa- Okumoto	0.2132	Yes	53.6%	2	2491
	Geometric	0.2318	Yes	42.7%	3	3411
	Jelinski-Moranda	0.2141	Yes	53.0%	4	2072
	Musa Basic	0.1827	Yes	75.0%	5	2320
C	Quadratic LV	0.2179	Yes	31.4%	1	2053
	Musa- Okumoto	0.2053	Yes	38.4%	2	2861
	Geometric	0.2507	Yes	17.5%	3	2862
	Jelinski-Moranda	0.2031	Yes	39.7%	4	2861
	Musa Basic	0.1929	Yes	46.3%	5	9480
D	Quadratic LV	0.1392	Yes	59.5%	1	750
	Geometric	0.2356	Yes	6.7%	2	638
	Linear LV	0.0809	Yes	≈ 82.5%	3	984

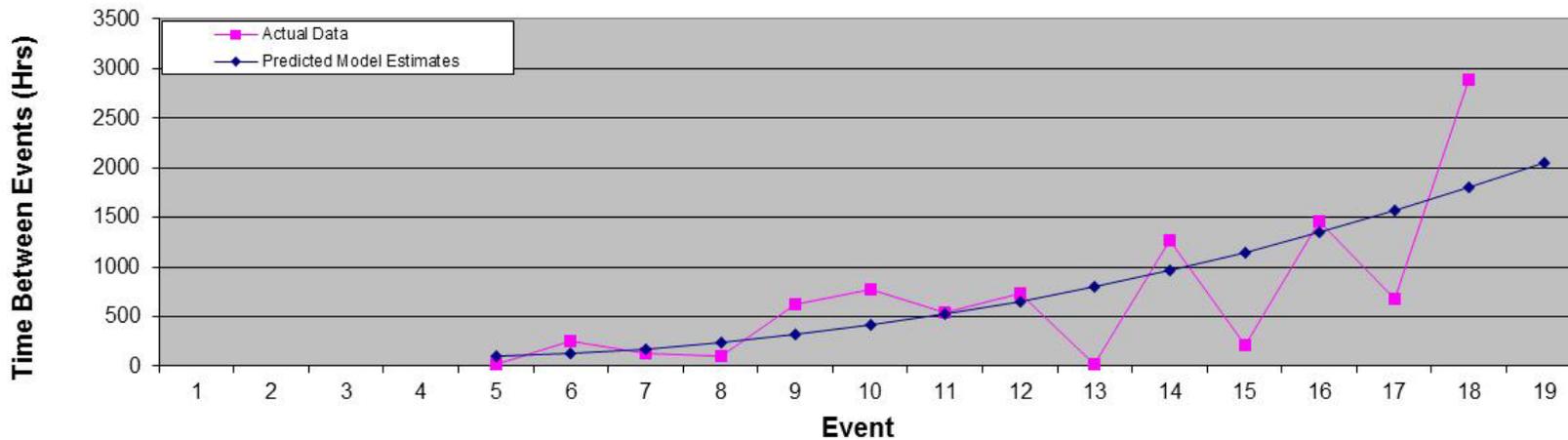
Selection of Best CASRE Model

Actuals vs. CASRE Tool Predictions

Plot of Actual Vs Predicted Module A



Plot of Actual Vs Predicted Module B



CASRE Tool Predictions Vs. Actuals

Summary/Conclusions

- The concerted applications of the reliability models, Keene/CMM, Rayleigh/SWEEP, and CASRE make the best use of available data to predict the code reliability at the various stages of development.
- Simple models perform as well or better than complex models— software reliability SMES [refs 4, 5,8] indicate the simple exponential model tend to outperform more complex models in terms of both stability and predictive ability.
- Execution time is the best measure of the amount of testing.
- Problem reports are a good surrogate for defects.
- The great advantage of CASRE is that it consolidates many well-established reliability models in one place and automates the determination of various ranking criteria for each model.

What Makes A Model Useful?

- 1. Stability During the Test Period and Remains Stable Until the End of the Test**
- 2. Reasonable Prediction of the Number of Defects that Will Be Discovered In Field Use**

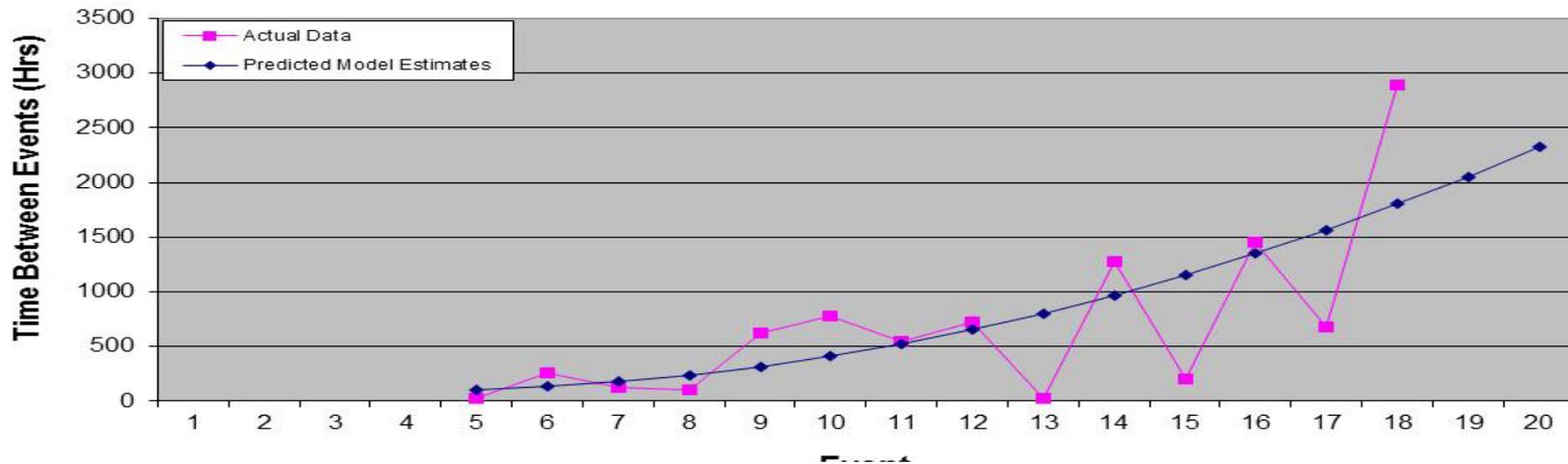
Additional Slides

References

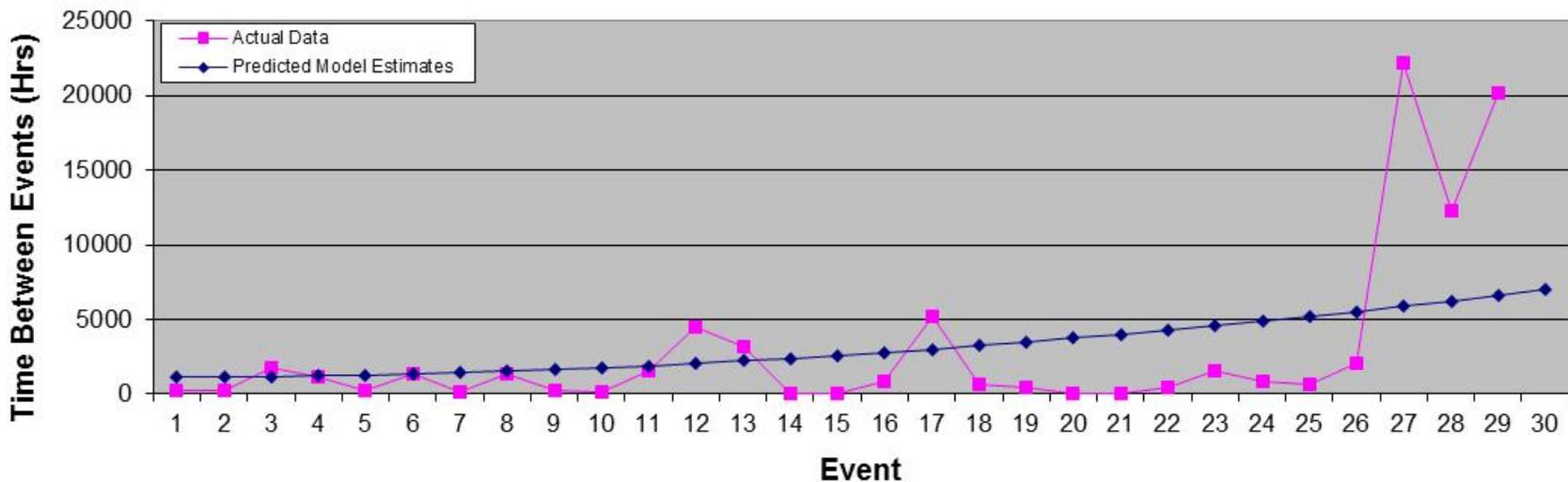
1. Burrows Evan , Keith McGough, Sue Tisdale. "An Introduction to Software Reliability Growth Modeling for Operational Test and Evaluation Personnel", MITRE, Dec. 2011.
2. Gullo Lou, Software Reliability Growth Approach, Raytheon, Oct. 2008
3. IEEE Std 1633™ 2008, IEEE Recommended Practice on Software Reliability, Jun. 2008
4. Keene Samuel, "Modeling Software R&M Characteristics," ASQC Reliability Review, Part I and II, Vol 17, No.2&3, Jun. 1997
5. Nikora Allen, Bill Farr et alii, CASRE (Computer Aided Software Reliability Estimation) v3 Beta, Copyright (C) National Aeronautics and Space Administration, Feb. 2000.
6. Peterson Jon, Sam Keene, Meng-Lai Yin. "Modeling Software Reliability by Applying the CARE tool Suite to a Widely Distributed, Safety Critical System", ISSRE 2001, Hong Kong, Nov. 2001.
7. Pfleeger Shari L. , Joann M. Atlee, Software Engineering: Theory and Practice, 4th Edition, Pearson Prentice Hall, 2006
8. Wood Alan, Software Reliability Growth Models, Technical Report 96.1, Sep. 1996

Actuals vs. CASRE Tool Predictions

Plot of Actual Vs Predicted
Module C

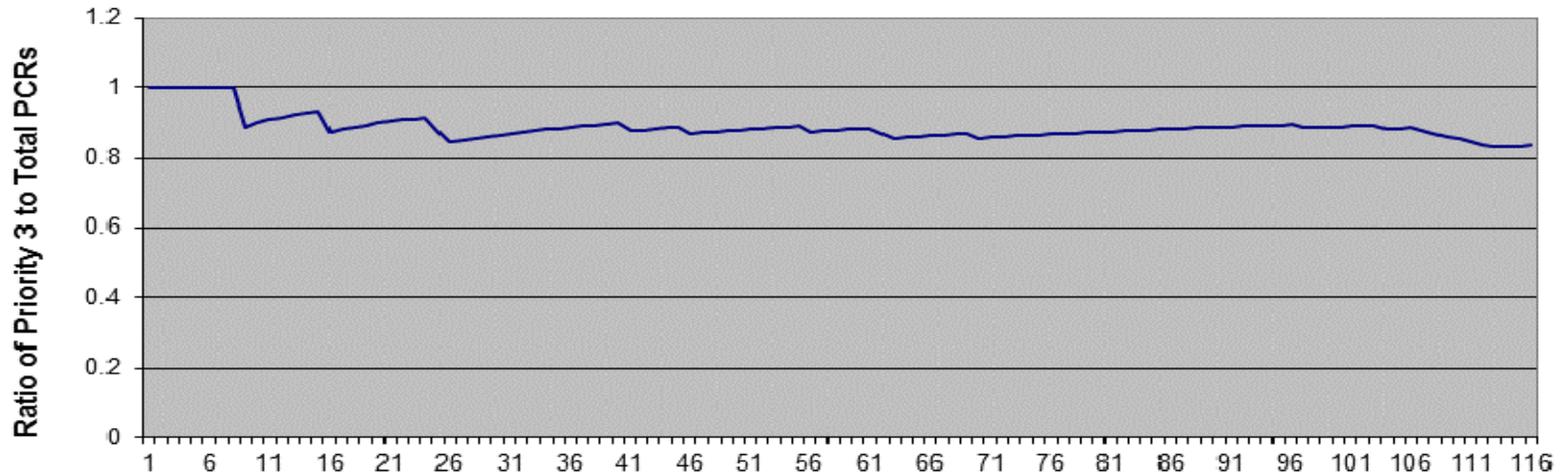


Plot of Actual Vs Predicted
Module D



Fraction of Non-Critical Failures Vs. Time

Fraction of Priority 3 vs Time for Module A



Fraction of Priority 3 vs Time for Module B

