The Estimation of the Reliability of a Large Tactical Information System

John Gaffney, Thomas Mulvehill,

and Paul Marmillion

Lockheed Martin Mission Systems

July 15, 2003

e-mail: j.gaffney@LMCO.COM

Tel.: 301-240-7038

FAX: 301-240-7009

Objectives of Presentation

- Describe an approach to estimating the reliability of software-intensive systems.
  - Variations of the reliability estimation methodology described here have been applied to various projects at Lockheed Martin as well as a predecessor organization.
- Consider how to estimate:
  - The defect discovery profile and latent defect content of a software system
  - The MTBF (or MTBO), *M*ean *T*ime *B*etween *F*ailures or *O*utages
- Present the application of the methodology to a large software intensive tactical information system, the *Tactical Input Segment* (TIS).

<u>*T*</u>actical <u>*I*</u>nput <u>*S*</u>egment (TIS)

- TIS was developed by Lockheed Martin and has been installed aboard the *USS Nimitz* aircraft carrier. TIS has also been delivered to the Navy's Washington Planning Center and the Naval Surface Air Warfare Center.
- The TIS system gives the Navy the capability to digitally receive and process reconnaissance imagery from multiple sensor platforms such as the U-2, Global Hawk, and the F/A-18 Shared Reconnaissance Pod (ShaRP). With the recent acceptance of the first system deployment, the TIS was immediately fielded as the tactical component of the Navy's Joint Services Imagery Processing System (JSIPS-N), the reconnaissance imagery program of record.
- Through a partnership with Utah State University's Space Dynamics Laboratory (SDL) and the Naval Research Laboratory, Lockheed Martin accelerated the processing of imagery from the F/A-18 SHaRP pod through the TIS system. SDL's expertise with the Navy Information System (NAVIS) was key to the ShaRP enhancement.

What is ***Reliability***?

- There are various definitions for ***reliability***. One good one is:
  - The probability that the system (or component thereof) will not fail for some specified period of time, commencing at some point in time.
- Some others are:
  - The probability of system success. (Shooman)
  - The probability that the software [system] will work without failure for some time. (Musa)
  - The probability of an item performing as specified under stated conditions for a specified period of time. The ability of an item to perform a required function under stated conditions for a stated period of time. (Software Productivity Consortium Glossary)
  - The probability that there is no failure during the time interval τ. A failure occurs when the system produces an incorrect result for [in response to] a valid input. (Conte)
  - The probability that software will not cause the failure of a system for a specified time under specified conditions. (IEEE Std-982)

Reliability and *Problems* or *Failures*

- Reliability has to do with the expected time between problems or failures of a system of interest or of one of its elements, such as its software.
- Various terms are used for *failure, problem*, etc.
  - There are no really universally agreed-upon definitions, and often one that is used in one instance may not be desirable in another due to the "political" baggage that it carries.
- The fundamental idea is that a *problem, defect, failure***,** etc. are words to cover the concept of deviations of a system or of a software or a hardware element of a system from its requirements or the standards to be followed in its construction.
- The focus here is to how to determine (estimate) the mean (average or expected) time between *countable or relevant* failures, commencing at some *point in time after delivery* of the system.
  - The estimate is based on data obtained during the development and testing of the system plus data about prior systems. Therefore, the better the data and projection models, the better the estimate.

Major Uses of Reliability Models

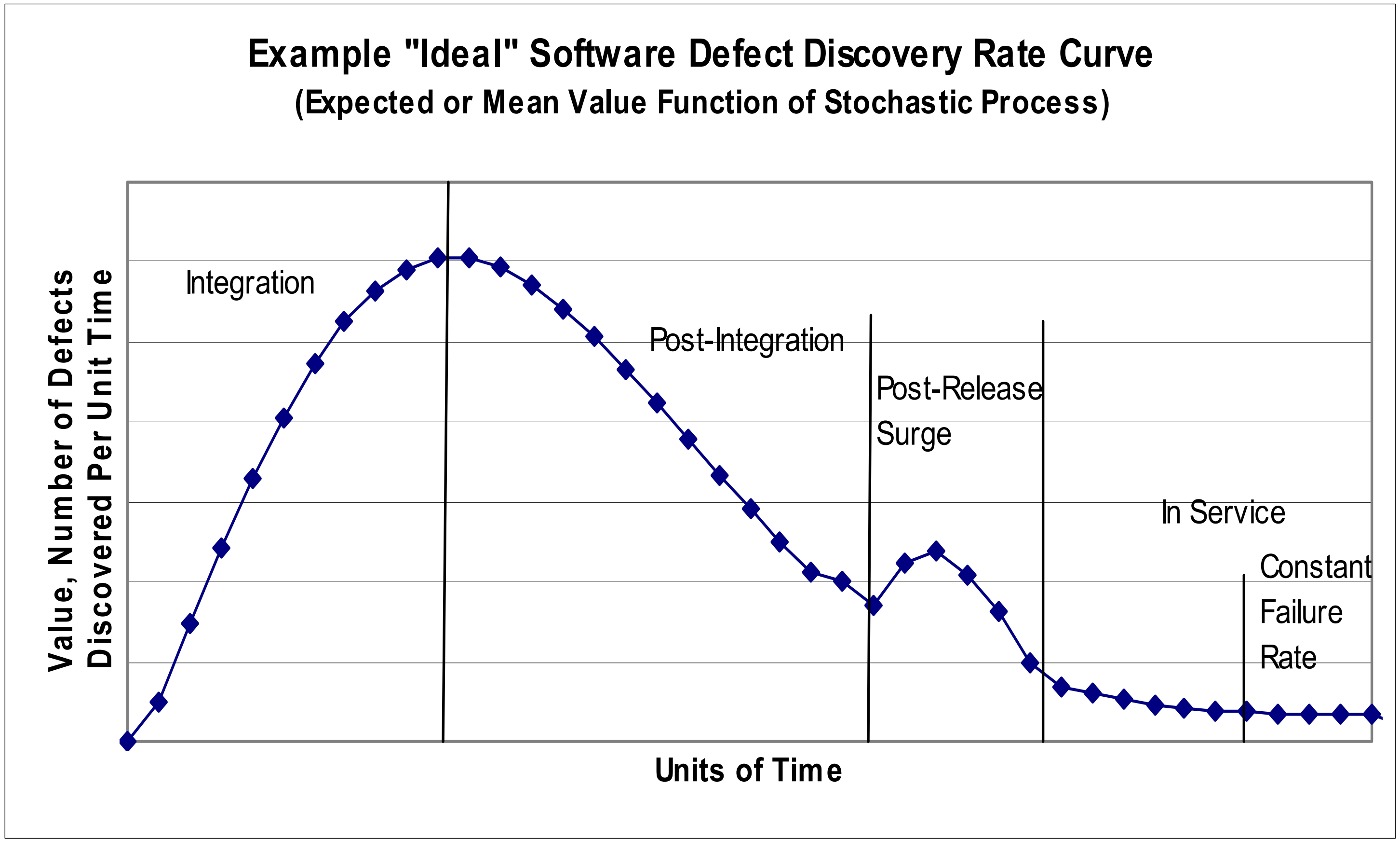
- Prediction: Use to ensure (at some level of confidence) that a proposed system will be able to meet its requirements. Will it be feasible in the reliability sense?
- Comparative Analyses: What are the reliabilities of other (similar, if possible) systems at delivery or at some particular time after?
- Development Control: We should set goals for the reliabilities of the software, the hardware, and the procedures (if applicable). What do we have to do to have confidence that the system that we are developing will meet it is reliability objectives? Development methodology? Test methodology? Estimation methodology?

Overview of Time-Based Software Failure and Reliability Models, 1 of 2

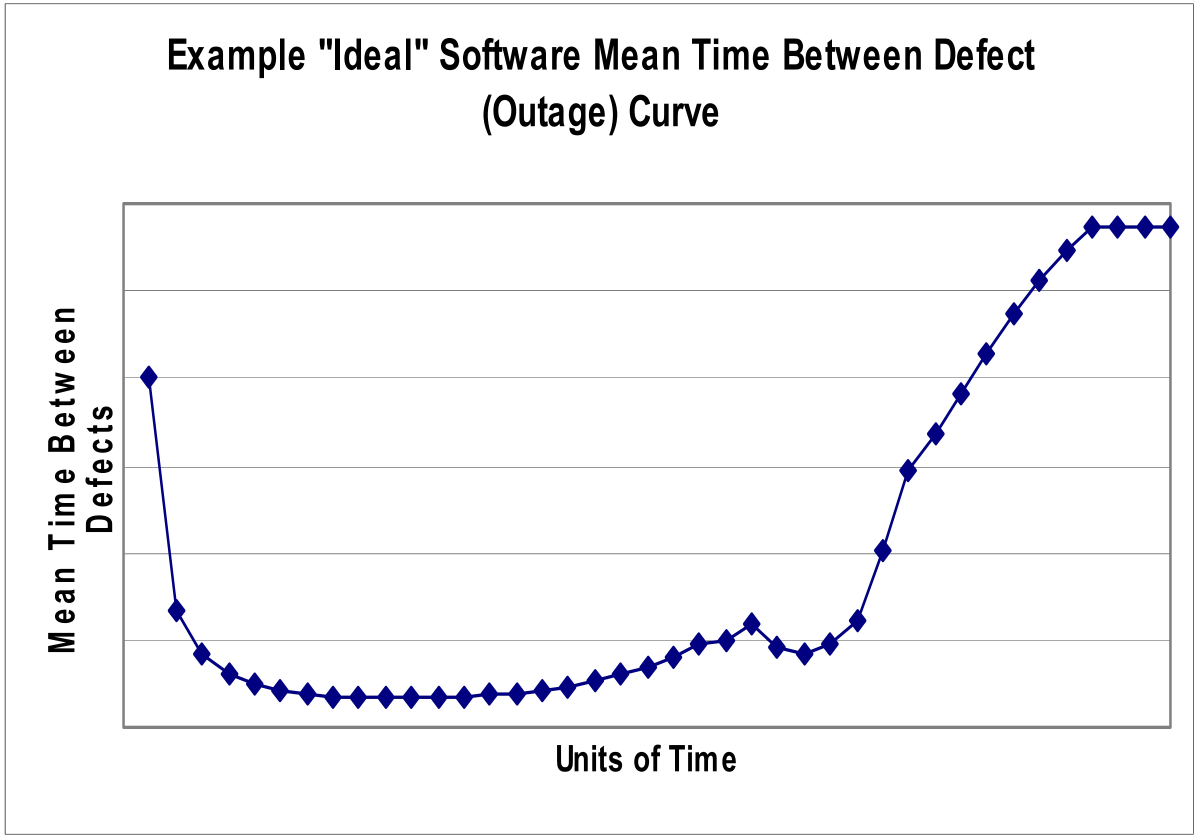
- Software failures are typically modeled as though the failure rate (failures per unit time) is a function of calendar time (it is actually a function of use). The reliability is the inverse of the failure rate (times a constant).
  - Over the life cycle, commencing at the beginning of integration, the failure rate typically initially increases and then decreases.
    - Often modeled as a Rayleigh curve (one of the family of Weibull curves)
    - Cumulative Version of Weibull: $N(t)=E*(1-(t/c)^x)$;where: E=total number of findable failures or defects; N(t)=number of failures from time 0 to t; x=shape parameter (x=1 for exponential and 2 for Rayleigh);c=scale parameter.
      - More convenient form:$N(t)=E*(1-b*t^x)$; where: $b=1/c^x=v/t_p^x$
      - V=a number that depends on x ;t_p is the location of the peak (for x>1.0) of the curve (failures or defects found versus time).V=0.5 when x=2.0 (for a Rayleigh distribution).
    - Post-integration and post-delivery, the rate is modeled as a monotonically decreasing function of time
      - Often modeled as a decaying exponential curve (also one of the family of Weibull curves)

Overview of Time-Based Software Failure and Reliability Models, 2 of 2

- Although the Weibull models represent the post-delivery rate as a decreasing function of time:
    - It may be convenient for planning purposes to model the post-delivery failure rate for software as a constant, at least after some point in time.
    - It is likely that there will be a "defect surge," a "bump" in defect discovery, for a period immediately after delivery, because of additional error paths opening up due to differences of the testing environment from the operational environment. Model this as an addition or "delta" on top of the Weibull curve.
- When we have estimated the mean value function for failure occurrence, $\lambda(t)$, we can obtain the corresponding estimate for the mean time between failures, MTBF, as $(1/\lambda(t))$.
    - For example, if $\lambda(t) = 5$ failures per day, at some value of t, then the MTBF= 0.2 days between failures, or perhaps more conveniently expressed, 4.8 hours between failures, at that time.
    - At each point in time, $t=t_0$ (think of an interval of time, practically speaking), the **expected number** of defects to be found is $\lambda(t_0)$, and the **actual number is distributed according to a Poisson distribution**, with mean $\lambda(t_0)$ and standard deviation = $\mathrm{sqrt}(\lambda(t_0))$.

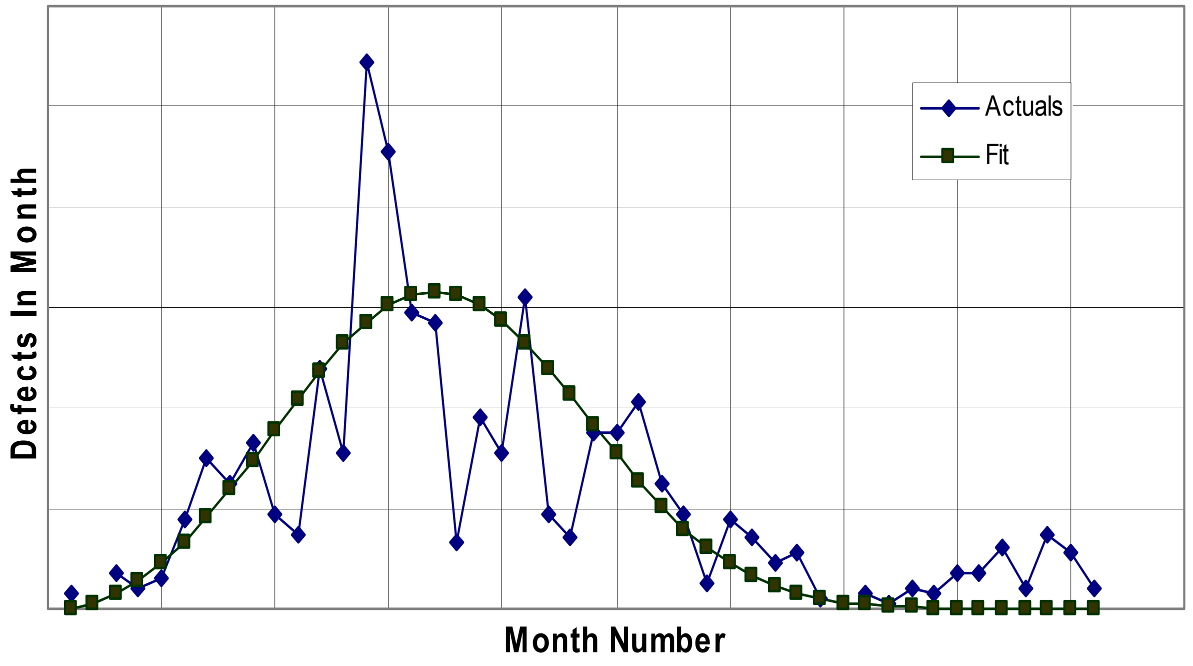


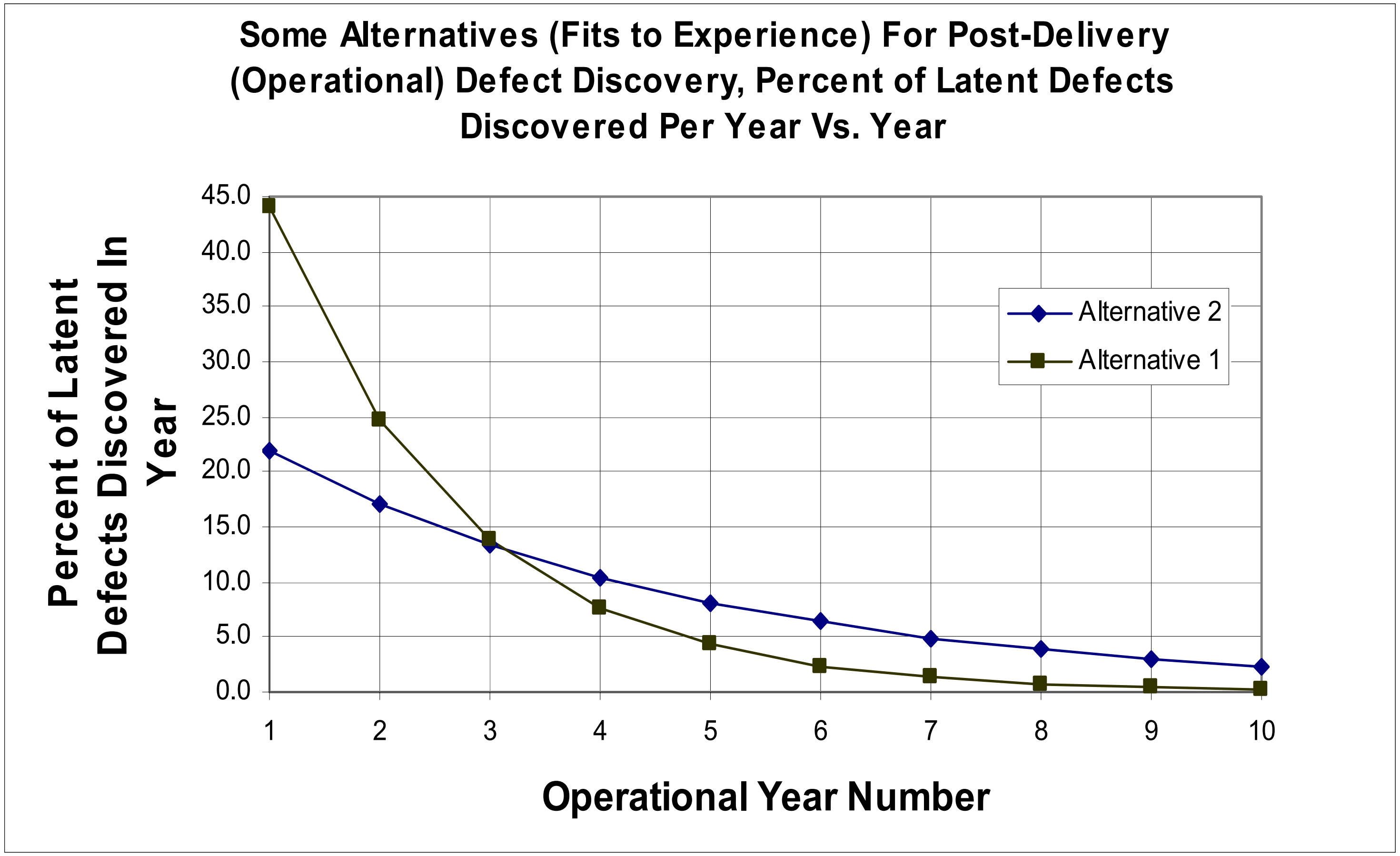
"Ideal" because there are no jiggles in the plot as there would be with "real" data

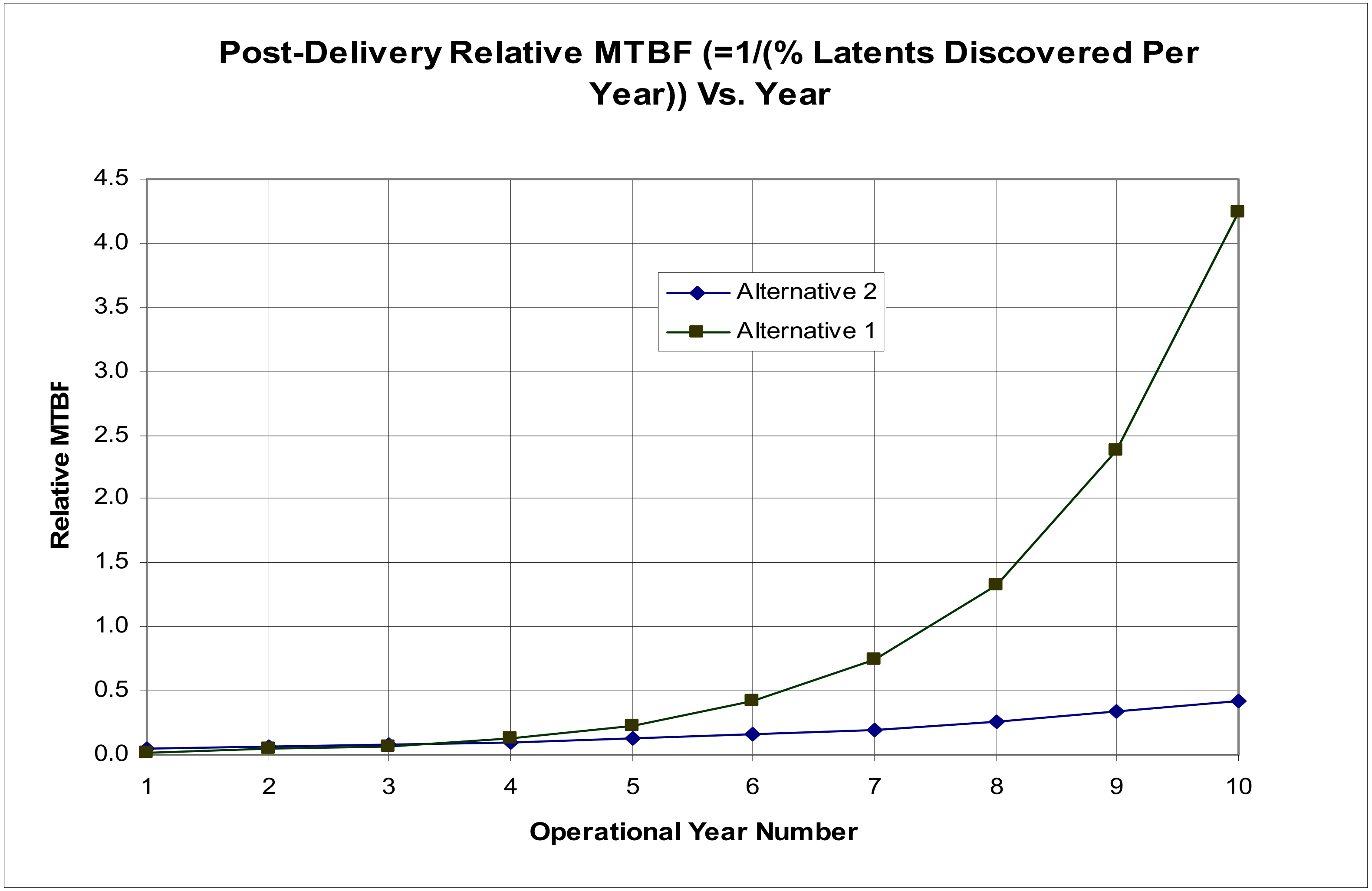


"Ideal" because there are no jiggles in the plot as there would be with "real" data

Example Defects Per Month Vs. Month, Data and Fit




Some Alternatives (Fits to Experience) For Post-Delivery (Operational) Defect Discovery, Percent of Latent Defects Discovered Per Year Vs. Year
Percent of Latent Defects Discovered In Year
Operational Year Number
Alternative 2
Alternative 1
45.0
40.0
35.0
30.0
25.0
20.0
15.0
10.0
5.0
0.0
1
2
3
4
5
6
7
8
9
10


Post-Delivery Relative MTBF (=1/(% Latents Discovered Per Year)) Vs. Year
Relative MTBF
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
1
2
3
4
5
6
7
8
9
10
Operational Year Number
Alternative 2
Alternative 1

Defect Data Fitting and Projection Using the STEER II Model

- STEER II is the latest version of a tool (currently excel-based) that was originated in the former IBM Federal Systems Division, a predecessor organization of Lockheed Martin Mission Systems, developed circa 1985.
- A subsequent version of the tool was developed at the Software Productivity Consortium.
- STEER II develops fits and projections for phase-based and time-based software defect discovery data.

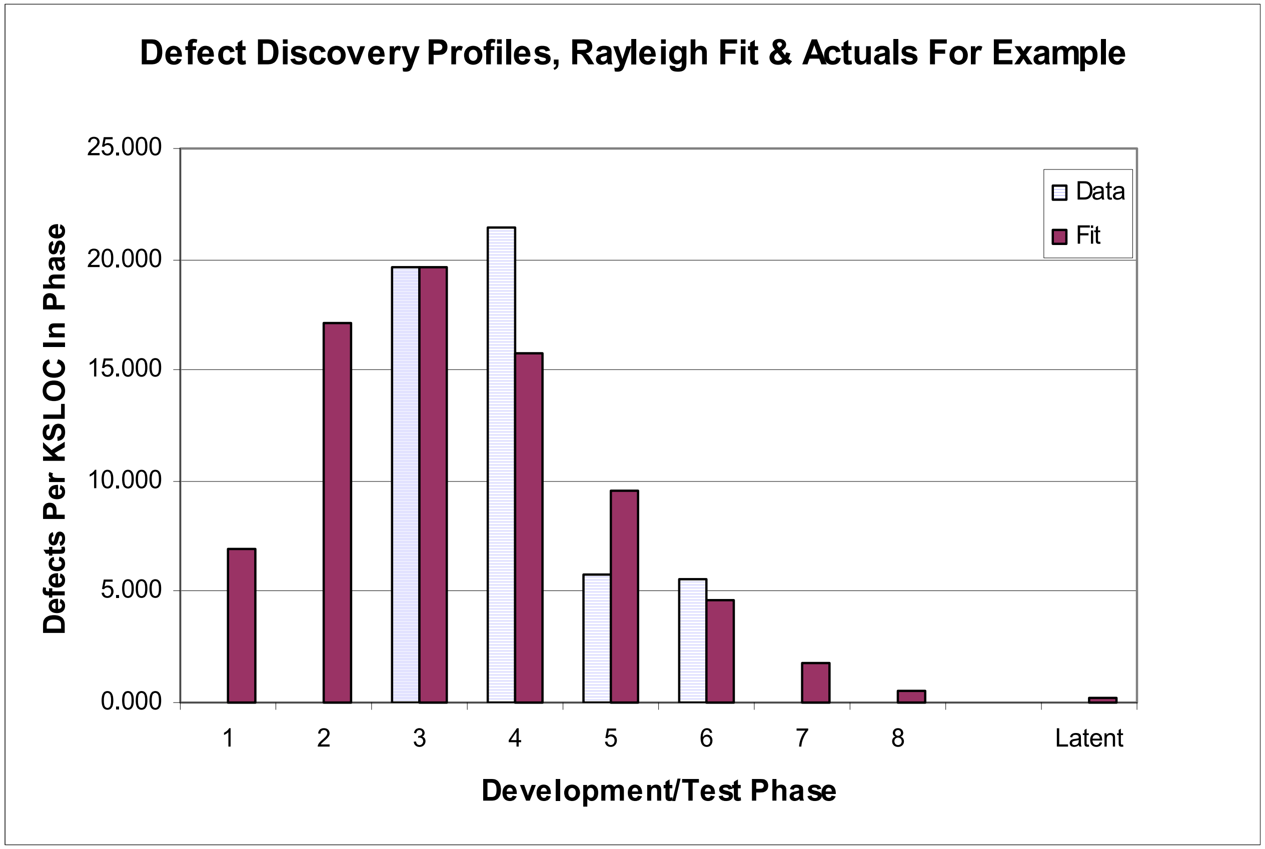
Example of STEER II Phase-Based Data Fit/Projection

STEER II Three To Nine-Phase Rayleigh Defect Discovery Profile Fit

Defect Discovery Profiles, Rayleigh Fit & Actuals For Example

Phase		Data	Fit To Data	Absolute Value,	Cumulative Fit	Cumulative
Number	Name	Defects/KSLOC		Rel. Error of Fit		Entered
1	1		6.89		6.89	
2	2		17.14		24.02	
3	3	19.600	19.66	0.0031	43.68	
4	4	21.400	15.72	0.2654	59.40	
5	5	5.750	9.58	0.6653	68.98	
6	6	5.600	4.60	0.1786	73.58	
7	7		1.77		75.35	
8	8		0.55		75.91	
	Latent		0.18		76.08	
		Average Rel. Error of Fit=		0.2781		

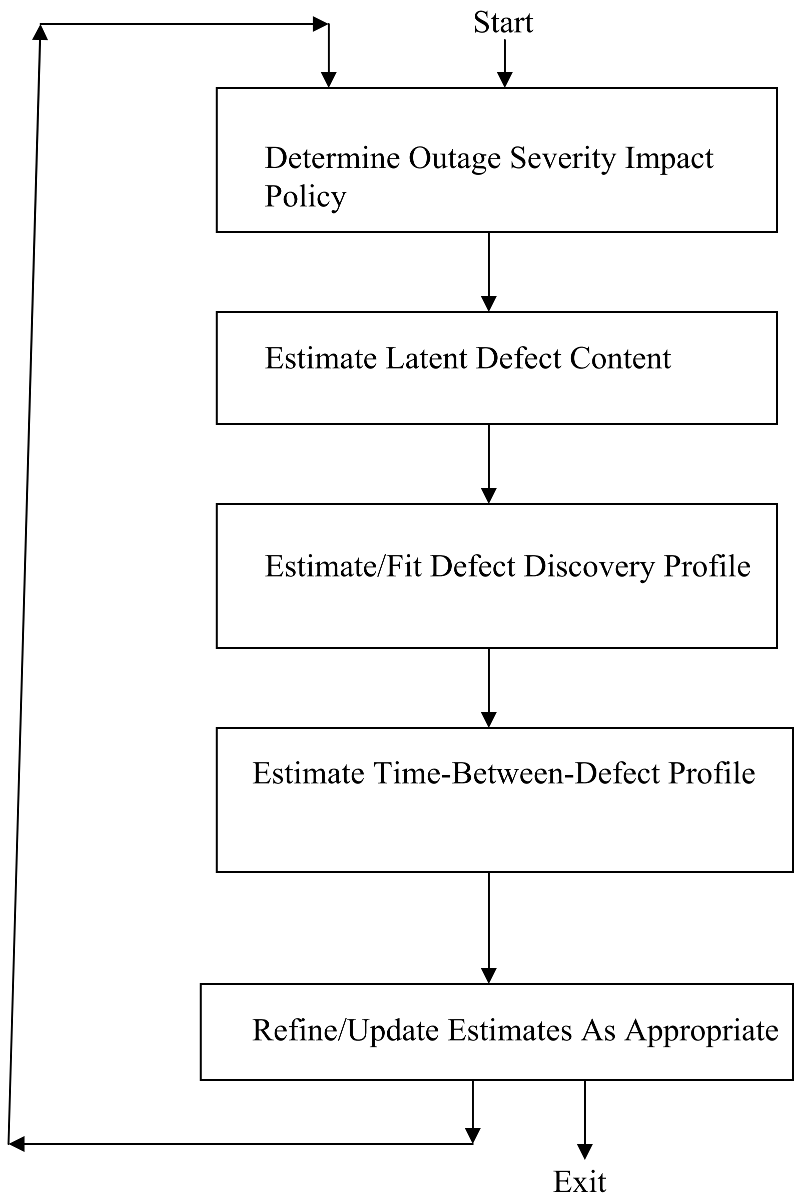
Example of STEER II Phase-Based Data Fit/Projection



Software Reliability Methodology Summary

- Estimate Latent (Post-Delivery) Defect Content:
  - Initially, use phase-based defect estimation; enables you to estimate latent defect content before (time-based) defect discovery data is available.
    - If estimated figure is not desirable, the early availability of the estimate may make corrective action feasible.
  - When time-based data becomes available, use it to refine the latent estimate.
- Estimate Latent Defect Discovery vs. Time (Rate) Profile
  - Determine form of discovery curve; the MTBO or MTBF curve has the form of its inverse.
    - Initially, estimate using normalized defect discovery profiles based on prior project experience.
    - Later, when time-based data is available, develop estimate for project.
- Estimate MTBO/MTBF Vs. Time Profile
  - This is done for each of the several types of code that compose the software system of interest, e.g., new/modified from supplier 1,…, reused from supplier 1,…. and then combine the values to obtain values for the software system overall.

Reliability Estimation Methodology Flow


Start
Determine Outage Severity Impact Policy
Estimate Latent Defect Content
Estimate/Fit Defect Discovery Profile
Estimate Time-Between-Defect Profile
Refine/Update Estimates As Appropriate
Exit

Summary of TIS Project Reliability Estimation Process

- Used the STEER II tool several times to estimate the latent defect content for the new code and modifications to existent code, as more data about defect discovery became available during the development and testing process.
- Initially, estimated the latent defect content for the new, modified, and the reused code developed by the Utah State Space Dynamics Laboratory based on a little data and estimates based on the Lockheed TIS code and other Lockheed projects' code.
  - Updated the estimates as testing and use data became available.
- Combined the latent estimates and the defect (failure) profiles for each type of code to create an overall TIS Time-Based Defect Discovery (expected software-caused system failure) Profile using the *TIS Software Reliability Estimation Tool*.

Factors That Contribute To Poor Estimates

- Lack of accurate and reliable data; data is often quite noisy
- Lack of historical data with which to compare estimates
- Focus on getting “the right answer” (“what the boss wants”) instead of the best answer.
- Too much reliance on unthinking use of models and/or estimator naiveté; lack of estimating experience
- Lack of a systematic estimation process, sound techniques, or models suited to the project's needs
- Unrealistic expectations and assumptions
  - “We will do much better on this project than on the last one.”
  - Failure to recognize and address the uncertainty inherent in software estimates.
    - “The model says xxxxx, therefore, that must be the case !”

Final View

- Care should be given to the definitions used for *defect*, *problem*, etc. when fitting data to models.
- Estimates are only as good as the data and the models used to compose them.
- The Weibull family of models has been found quite useful in estimating reliability and availability.
- Don’t wait until testing data is available (from “dynamic’ verification stages) to make defect discovery and reliability estimates for your project.
  - Initially, make a phase-based estimate using data from inspections and other “static” verification stages.
  - When sufficient time-based data is available, update the estimate.