Defects/Anomalies Estimation and Management Workshop
PSM 2006

July 27, 2006

John Gaffney, Lockheed Martin
j.gaffney@lmco.com
Chris Miller, SSCI
miller@systemsandsoftware.org

1

Workshop Overview

- **Address Questions such as:**
  - **Do you set goals for defect discovery rates and related measures such as latent defect rates and number of escapes? If you do, what drives the selection of the goals, process improvement objectives, customer requirements, or what?**
  - **Do you use mathematical techniques for defect estimation and projection? If so, what are they? If so, are these techniques imbedded in a tool, and if so, what tool?**
  - **Where do you see defect estimation techniques and the like going in the future? Do you perceive a business need driving their (increasing?) use or not?**
- **Possible Goals/Products of Workshop**
  - **Share experience and data if possible.**
  - **Document some sense of use and practice in defect/anomaly management, including setting goals, tracking/estimating, and taking action.**
  - **Document perceived need for improvements in defect modeling, management of defects, and related matters.**

2

Some Background

3

Problems or Failures

- 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.

4

Major Uses of Defect Models

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

5

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 !”

6

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

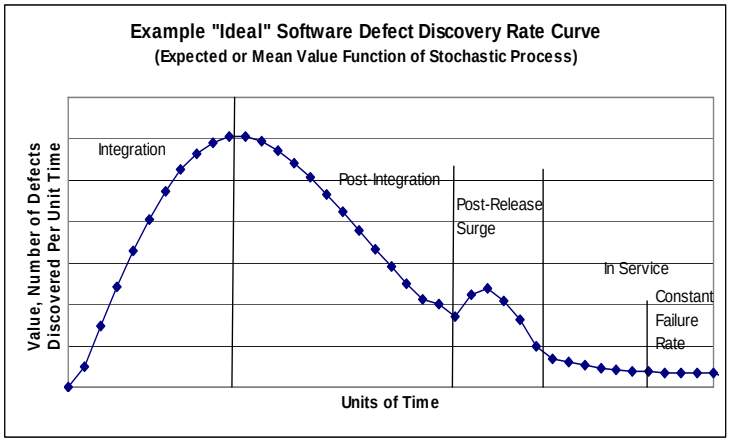
- Software failures are typically modeled as though the failure rate (failures per unit time) is a function of calendar time (it is actually a function of use). The reliability is the inverse of the failure rate (times a constant).
  - Over the life cycle, commencing at the beginning of integration, 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)

7

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

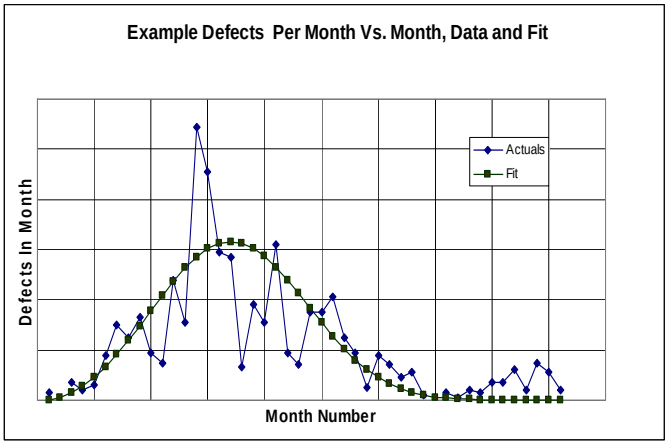
- Although the Weibull models represent the post-delivery rate as a decreasing function of time:
  - It may be convenient for planning purposes to model the post-delivery failure rate for software as a constant, at least after some point in time.
  - It is likely that there will be a "defect surge," 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 = $\text{sqrt}(\lambda(t_0))$.

8



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

9



10

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.

11

Example of STEER II Phase-Based Data Fit/Projection

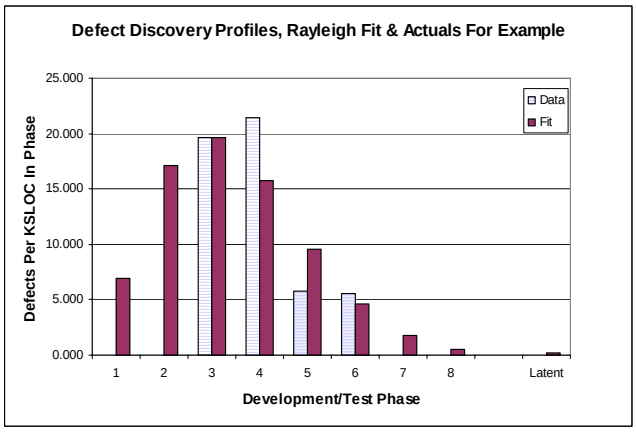
STEER II Three To Nine-Phase Rayleigh Defect Discovery Profile Fit

Defect Discovery Profiles, Rayleigh Fit & Actuals For: Example

Phase Number	Name	Data Defects/KSLOC	Fit To Data	Absolute Value, Rel. Error of Fit	Cumulative Fit	Cumulative 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		

12

Example of STEER II Phase-Based Data Fit/Projection



13

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.

14