ON USE OF SOFTWARE RELIABILITY GROWTH MODEL

DR. DHANANJAYA REDDY

Abstract: This paper presents state of the art in designing, implementing, and testing web software and assesses the software reliability of web applications by using software reliability growth models. This paper discussed the relevant reliability issues and best practices of growth models in terms of web software reliability.

Keywords: growth models, MTTR, network reliability, performance, software reliability.

Introduction: Software reliability is an important component in critical business applications. Developing reliable software is very difficult because there is interdependence among all the software modules as of existing software which deals with software security. Even the security is managed separately; before the security hole is patched any failures of the application will have great impact on the application reliability. This makes difficult the usage of the standard software reliability growth models for insecure systems.The reliability of the web-based applications can be considered as a special case of distributed software running on distributed computer systems, over different kind of networks. If the nodes of the network are assumed to be perfect and the connections among nodes are assumed to fail in a statistically independent manner, then the network reliability can be computed. However, if the nodes are imperfect, then the usage of the algorithm is an efficient solution. All existing software reliability models are developed for the software products that are statically constructed normally by a company or institution that has the full control of the development process. The evolutional shift from the product-oriented software architecture to the Service Oriented Architecture (SOA) and Web Services (WS) invalids many techniques developed for traditional software.

Reliability is probably the most important of the characteristics inherent in the concept Software Quality. Reliability is connected with defects and as we know defects represent the largest cost element in programming. Reliability assumes, totally or partially, many properties that are often quoted as aspects of quality. Reliability represents a user-oriented view of software quality. Initial approaches to measure software quality were based on attempting to count the faults or defects found in the program. Reliability is a much wider measure. It is customer/user oriented rather than developer oriented. It relates to operation rather than design of the program hence it is dynamic rather than static. It takes account of the frequency with which problems occur. Further, it relates directly to operational experience and the influence of faults on that experience. Thus reliability measures are much more useful than fault measures. As computer applications became more diverse and spread through almost every area of everyday life, reliability has become very important characteristics of computer system. During the last three decades, the software reliability engineering has played a central role to provide several quantitative methods used in the real time software development processes. Since the assessment of software reliability is one of the main topics in this area, one needs any mathematical model to assess quantitatively the software reliability which is the probability that the software system does not fail during a specified time period. Measurement of software reliability involves estimation of software reliability or its alternate quantities from failure data. The term software reliability prediction is defined by Hechtr in 1977, the process of computing software reliability parameters from program characteristic. Typically, software reliability prediction takes into account factors such as the size and complexity of a program, and it is normally performed during a program phase prior to test. The principal objective of a software reliability model is to forecast failure behavior that will be experienced when the program is operational. This expected behavior changes rapidly and can be tracked during the period in which the program is tested.

Software reliability growth models: Reliability growth models are based upon the assumption that the reliability of a program is a function of the number of faults that it contains. Such models apply statistical techniques to the observed failures during software testing and operation to forecast the product's reliability. In order to be effective, the data used in the growth model is taken from where the software will be deployed.

There are two types of software reliability models which help to predict software reliability by executing test cases. The first types of models are called "defect density" models. These models use loop, lines of code, input or output and external references to find out the number of faults in the software product. The second types of models are called "software reliability growth models". These models used to correlate defect detection data statistically with known functions such as an exponential function. If the correlation is good, the known function can be used to forecast future behavior [1]. Software reliability is

defined as the probability of failure free software operation for a specified period of time in a specified environment. Software reliability is accepted as the major factor rather than the other factors like as usability, functionality and maintainability etc.

The failure of any one system or service in the path between server and user will in effect cause failure of the entire application as far as the user is concerned. The important characteristics of a website are high reliability, high availability, high security and more interactivity. All these characters are independent on the web application [3]. Reliability growth models are based upon the assumption that the reliability of a program is based on the number of faults. Statistical techniques are used by such models to observe the failures during software testing and operation to measure the product's reliability. If we have N installations of the software, and a total of F failures are reportable at time T, the failure rate of the software can be computed as

λ = F / (N*T). It is important to know the mean time to repair for a component that has failed. Combining this time with the mean time to failure tells us how long the system is unavailable for use; the mean time between failures (MTBF) is simply; MTBF = MTTF + MTTR.

The most important measures of reliability may be the probability of failure on demand. A software reliability growth model can be regarded to be a mathematical expression which fits the experimental data. It may be obtained simply by observing the overall trend of reliability growth. An analytically obtained model has the advantage that its parameters have specific interpretations in terms of the testing process.

The different users use the software due to which it becomes easier to calculate the reliability of software. In the most software reliability growth models a parameter is used to relate the total number of faults contained in a set of code. If this parameter and the current number of faults discovered, we know how many faults remain in the code (Fig 1).Knowing the number of residual defects helps us decide whether or not the software is ready to release and how much more testing is required if we decide the software is not ready to release. It gives us an approximation of the number of failures that our customers will come across during the operation of software. This approximation helps us to plan the suitable levels of support that will be required for fault correction and also verify the cost of underneath software.

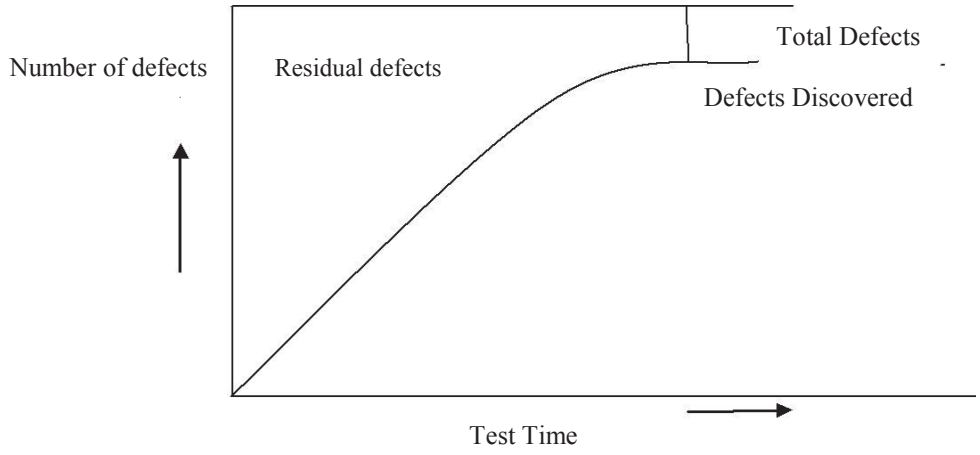


Fig 1. Residual Defects

Exponential approach: Exponential model is a two parameter model. The exponential function follows randomness. Thus, for each *i*, we can express the distribution function as;f(t) = 1 − $e^{-\lambda iti}$ The inference procedure for computing *λi* is to calculate the average of the two previously observed value of *ti*.

Types of reliability growth models: Software reliability growth models have two types of models: concave and S-shaped. Both models have the same asymptotic behavior. In both models, the rate of change is decreases as the number of faults detection increases. Both models are shown in Figure. It is assumed that the rate of change is proportional to the number of faults in the software. Each time a fault is detected or repaired, the fault detection increases and the rate of change decreases. The concave model firmly follows this prototype. The early testing is not as efficient as later testing assumed in the S-shaped model, so there is a ramp-up period during which the defect detection rate increases. Let us suppose if the early Quality Assurance test uncover the faults in other products that prevent QA from finding defects in the product being tested [1] .

In various Software Reliability Growth Models, two important factors affect the reliability. The first factor is the number of actual defects and the second is the

rate of change. These both are used to represent the reliability of software or web applications. After the detection of faults it becomes easier to decide whether the software is suitable for customer to use or not and how much more testing resources are required. It becomes easier to estimate the number of defects that will be encountered in the future. The rate of change is used to measure the effectiveness of fault detection by executing the test cases. The rate of change is assumed constant in maximum cases. The most researchers think that all defects have equal probability of being detected during the testing process due to which the rate of change is constant. But in reality, the rate of change is based on the program size, skill of test teams and software testability. The rate of change can be increase, decrease or remain constant [5]. Software Reliability Growth Model (SRGM) is used to correlate defect detection data with estimated residual defects and time.

SRGMs could be divided into several types based on their reliability growth curves [2] ,such as the Goel-Okumoto (GO), the delayed S-shaped (DeS), the inflected S-shaped (InfS), and the generalized Goel-Okumoto (GGO) models, are considered as the candidate models for our proposed weighted combinations. It is noted that the parameters of these candidate models will be estimated by applying Maximum Likelihood Estimation (MLE) in advance [2].

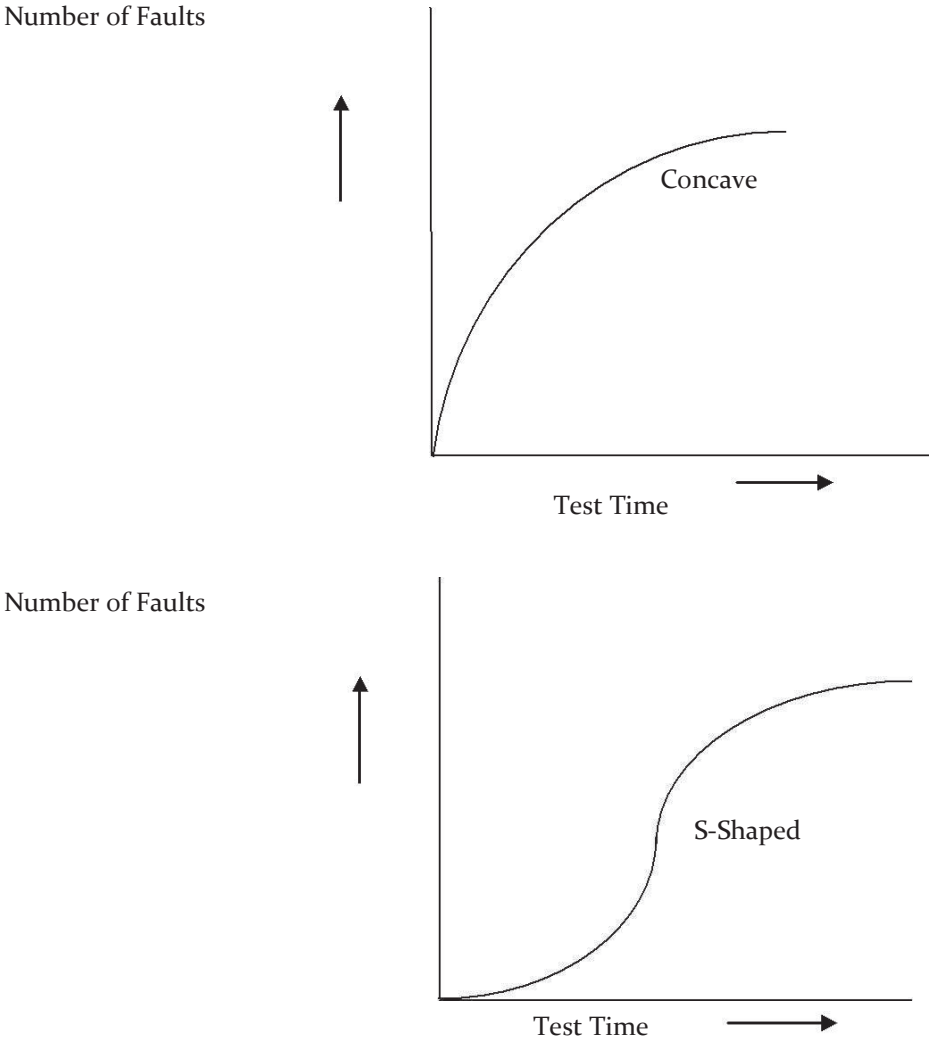


Fig 2: Concave and S-Shaped Models

Web software reliability: The web-based software is built in order to provide some functionality using different web services protocols and frameworks oriented to a specific application, as mentioned by Chu & Qian in 2009. For instance, *E-business XML* (or ebXML) is a useful protocol when processing electronic business information over various platforms. Most of web application software is built using layered architecture. Three tier architecture is widely used now-a-day. This architecture has three

entities; client, server and database. The web application has n-tier architecture based on pattern design, plug-ins, with a modular structure using a specific user interface. An important factor influencing the web server reliability is the network reliability and availability. In web application, there is strong connection between the reliability and d the network performs ability. A global analysis considers both hardware and software fault categories when studying the web application reliability.

Use of software architecture analysis is to study the current software architecture of products and processes with evolution plans for enhancing the software reliability of newer products with minimal impact on existing products. According to ANSI, Software Reliability is defined as: the probability of failure-free software operation for a specified period of time in a specified environment. Software reliability engineering is therefore defined as the quantitative study of the operational behavior of software based systems with respect to user requirements concerning reliability. As a proven technique, SRE has been adopted either as standard or as best current practice by more than 50 organizations in their software projects and reports, including, AT & T, Lucent, IBM, NASA and in many other countries.

Network reliability: The network reliability is studied, to assure, at least theoretically, a solution to the following problems: (1) Compute the probability that there is a path between two distinguished vertices a, and b ie., terminal connectivity; (2) Compute the probability that all vertices remain connected. It is clear that both combinatorial and statistical methods are mixed in order to compute the network reliability.

Considering the most used types of distributed web-servers (DWS) the following architectures are possible: cluster based (with virtual IP address depending on the web service visible to the clients, and a real IP address of the cluster nodes (CN), but hidden to clients), virtual cluster (the nodes sharing the same IP, and only one node will keep a message from clients), and distributed cluster (every node having its IP, and the message being redirected by a dynamic procedure applied related to the Domain Name System). The redistribution is implemented in a switching (SW) system. The web based system reliability can be computed as in the case of serial systems: R(DWS) = R(SW) x R(CN). It is clear that R(CN) depends on the cluster topology, but the estimation of R(CN) depends also on the method of content mirroring, the best results being obtained for complete replication.

Weighted combinational models for web-based software: Since 1970s, many software reliability growth models (SRGMs) have been proposed for the reliability estimation of software products during software development processes [2]. However, application of SRGMs has shown that there is commonly great disagreement in software reliability predictions, while none of them can be trusted to provide consistently accurate results across different applications. SRGMs depend on failure data to estimate failure pattern and reflect different testing conditions [2].

There are three weighted combinational models based on the calibration of arithmetic mean and geometric mean [1], which respectively represent linear and nonlinear relationships between models. These weighted combinational models are equally weighted (EW), linearly weighted (LW) and nonlinearly weighted (NLW) combinational models.

If the linear relationship exists in models, LW can be expressed as follows [3]:

$$\hat{E}(t) = \sum_{i=1}^{n} W_i \times E_i \quad , \quad \sum_{i=1}^{n} W_i = 1 \qquad (1)$$

Where n is the number of models, E_i (t) is the estimated result (i.e., cumulative number of faults) of the *i*th model by time *t*, and $W_i(t)$ is a stationary weight given to the *i*th model.

LW is relevant to the arithmetic summation of estimated results with weights [1].

The relationship between models is nonlinear; NLW can be defined as follows:

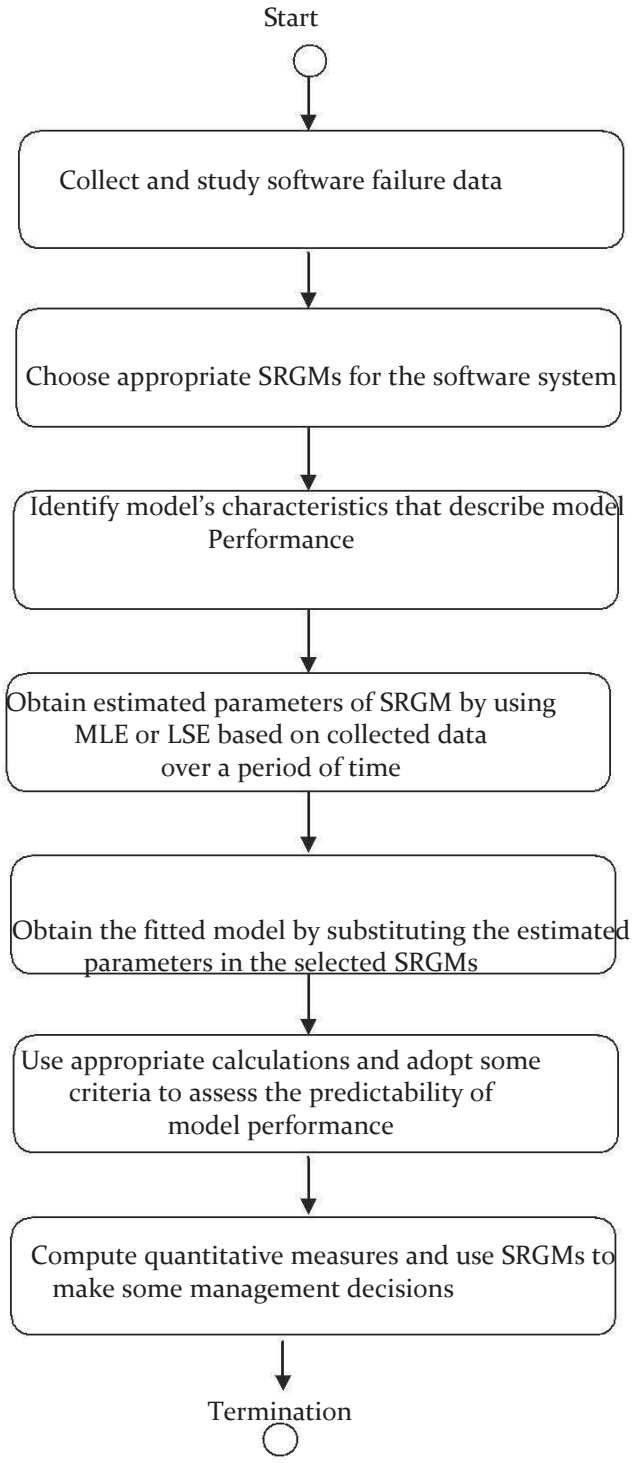
$$\hat{E}(t) = \sqrt[n]{\prod_{i=1}^{n} E_i(t)^{W_i}} \quad , \quad \sum_{i=1}^{n} W_i = n \qquad (2)$$

The identity of NLW is based on the *n*th root of the geometric product of estimated results with exponential weights [1]. Clearly, if we take a logarithm on Eq.(2), it is given as

$$\log \hat{E}(t) = \frac{1}{n}\sum_{i=1}^{n} W_i \times \log E_i(t) \qquad (3)$$

We can see that NLW will become a special case of LW. Likewise, when W_i are all equal to 1 in Eq.(2) or Eq.(3), NLW can have an equal weight.

Procedure of Using SRGM



Methodology: In various Software Reliability Growth Models, two important factors affect the reliability. The first factor is the number of actual defects and the second is the rate of change. These both are used to represent the reliability of software or web applications. After the detection of faults it becomes easier to decide whether the software is suitable for customer to use or not and how much more testing resources are required. It becomes easier to estimate the number of defects that will be encountered in the future. The rate of change is used to measure the effectiveness of fault detection by executing the test cases. The rate of change is assumed constant in maximum cases. The most researchers think that all defects have equal probability of being detected during the testing process due to which the rate of change is constant. But in realty, the rate of change is based on the program size, skill of test teams and

software testability. The rate of change can be increase, decrease or remain constant [4].

Software Reliability Growth Model (SRGM) is used to correlate defect detection data with estimated residual defects and time.In the analysis, test cases are executed to find out the actual defects per day till date. The actual defects are plotted against time. The curve should be flattering out as time progresses. The curve shows the growth of reliability.

This reliability model based on Goel-Okumoto and Confidence interval (CI). This reliability model is based on parameters such as defects; Amount of Testing (in days) and Statistical Techniques (G-O and CI).The amount of testing can be measured in calendar time as well as number of tests run. In this example we will use calendar time and number of days as a parameter [2]. The following steps are used to detect the faults:

Step 1:- Collect data as per table shown in Software Reliability Growth Model. Compute the actual defects and rate of change accurately. Find out the values of total actual defects (a), test case efficiency or rate of fault detection (b) and current time (t).

Table-1

Days	Actual faults	Test executed	Rate of change
1	140	200	.70
2	125	200	.62
3	108	175	.61
4	102	175	.58
5	85	150	.56
6	80	150	.53
7	65	125	.52
8	60	125	.48
9	46	100	.46
10	40	100	.40
11	28	75	.37
12	25	75	.33
	904	1650	.51

Total number of defects (a) = 904

Rate of change or test case efficiency = 0.51

Step 2:- Compute the faults by using Goel-Okumoto model.

$$\mu(t) = a(1 - e^{-bt}) \quad , a>0, b>0,$$

where, $\mu(t)$ = current actual defects, using Goel-Okumoto Model.

a = initially a is taken as total defect detection till date

b = the rate at which the defect-rate decreases or the defects are detected

Goel & Okumoto is one of the most popular NHPP models in the field of software reliability engineering. Goel and Okumoto assume the mean value and intensity functions as Exponential SRM:

Mean Value Function (MVF):-

$\mu(t) = (1 - e^{-bt})$
$\mu(t) = 902$
Failure Intensity Function:-
$\lambda(t) = \mu'(t) = abe^{-bt}$
$\lambda(t) = 0.92$
Failure intensity function is used to estimate the quality of software.

Step 3:- Compute Confidence interval (CI), to get estimated defects with reasonable interval

***Confidence Interval Table*:-**

α	1-α	Zα/2
0.10	90	1.64
0.09	91	1.70
0.08	92	1.75
0.07	93	1.81
0.06	94	1.88
0.05	95	1.96
0.04	96	2.05
0.03	97	2.17
0.02	98	2.33
0.01	99	2.57

The confidence interval is estimated by using normal distribution with standard deviation. Where, α represents total area and α/2 represents half of the normal distribution. The confidence interval for 95% is equal to 1.96, represented by Zα/2.

Confidence Interval (CI) gives probable range of values from the set of sample data. The confidence level is the possibility coupled with a confidence interval, which is expressed as percentage (%).The confidence level tells you how sure you can be. The confidence interval is symmetrically located around the normally distributed estimation. The larger the confidence level is, the narrower the confidence interval [5].Wider confidence interval, gives more possibility to estimate the residual defects prediction in the right range. Hence 95% confidence level is the most frequent and reasonable value than 99% or 90%.

Let us use 95% confidence level to predict faults specified by:-
$CI = \mu(t) + Z\alpha/2 * SQRT(\mu(t))$
Predict defects at (t) day's formula:-$CIres = \mu(t) + Z\alpha/2 * SQRT(\mu(t))$
Predict defects at 13^{th} day:-CIres = 902+1.96*30.03 = 961
Reliability:- Reliability per day
$\lambda = .92$
$t = 1$ day
$R(t) = e^{-\lambda t} = 0.40$
Here we calculate the reliability of one day. From the result it has proven, as the number of faults increases per day the reliability will be decreases.

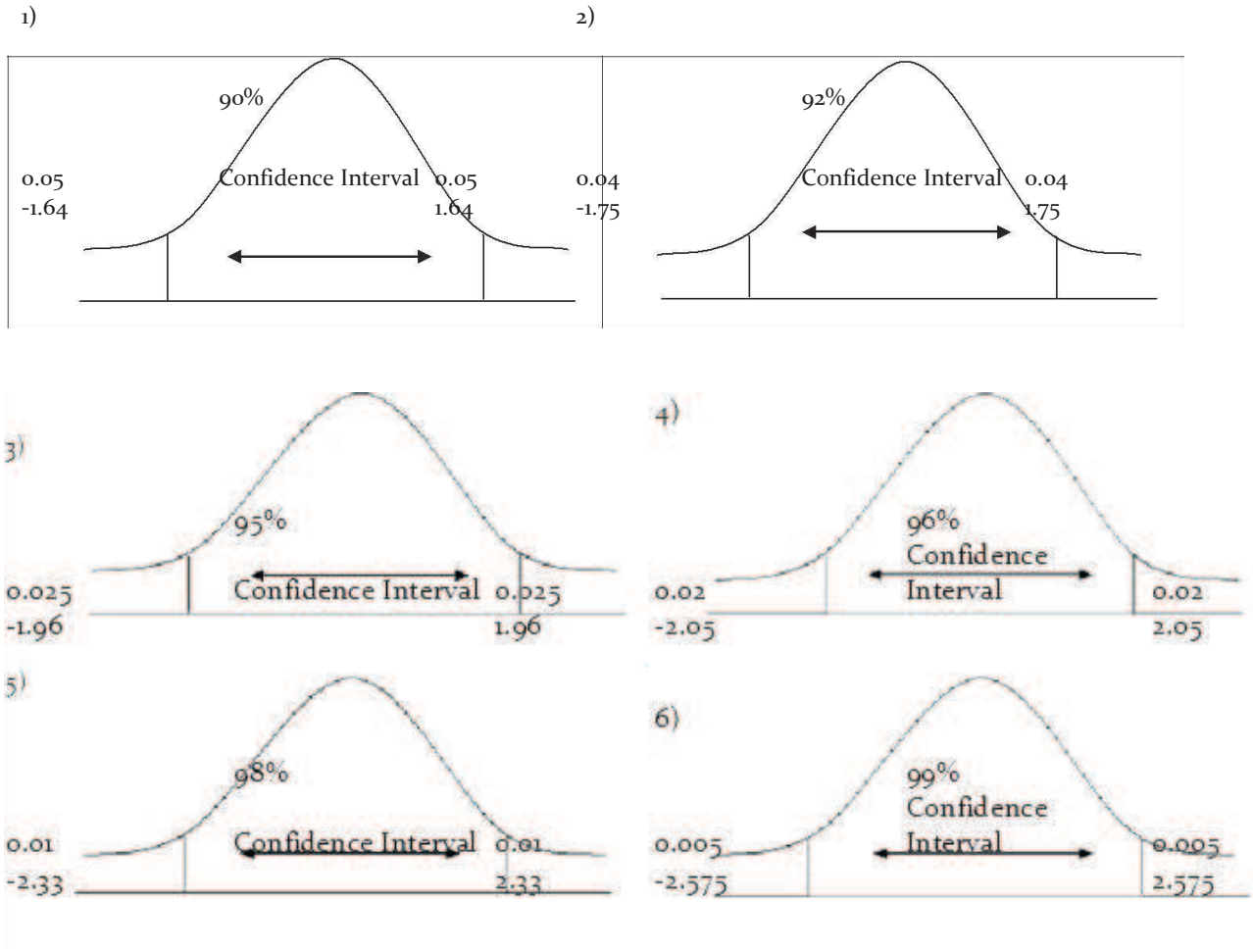
Software vulnerabilities: If omitting the failures generated by cyber attacks, we refer to the intrinsic reliability. In large, the software reliability covers also aspects related with security holes that permit to attackers the crashing of the web application. These security holes are generated by software vulnerabilities as defined in the following. Software vulnerability deals with insecure programming and the possible insertion, by mistake, of the following classes of bugs, as identified by (a) memory-management (b) concurrency-management (c) I/O-management (d) inconsistent integration of security technologies (e) numerical inconsistencies (f) vulnerable entry points. There are possible mistakes not only during design, but also during testing and implementation phases. Environmental and administrative mistakes are common when speaking about web-servers.

The vulnerabilities are possible to be identified: (1) manually, (2) automated by bottom up and/or top down testing (3) by black box testing, (4) by white-box analysis, (5) using scanners, and (6) combined various methods. The software trust ability will be increased by testing the software using environment perturbation. In order to minimize the security type vulnerabilities, the prevention of the cyber attacks is the best strategy and may use the following technologies: security tokens, digital signatures, encryption, and other security tools according to the security management procedure. Taking into account the above classes of bugs and the mentioned security technologies, the following types of web application attacks will be rejected: imposture repudiation information disclosure without permission, information altering, denial of services, and gaining the privilege of administrators or owner applications.

Conclusions: The cost of software application failures grows and as these failures increasingly impact business performance, software reliability will become progressively more important. Employing effective software reliability engineering techniques to improve product and process reliability would be the industry's best interests as well as major challenges**.** In this paper, three weighted combinational models, namely, EW, LW, and NLW, are proposed and can be used for reliability estimation of web-based software. Results from applying the proposed models to a web-based ERP system are compared with traditional SRGMs and

show that the proposed models can give better accuracy in software reliability prediction.

Confidence interval curves



References:

1. Bullen, P.S., Mitrinovic, D.S., and Vasic, M., *Means and Their Inequalities*, Springer, 1988.
2. Lyu, M.R., *Handbook of Software Reliability Engineering*, McGraw-Hill, 1996
3. Lyu, M.R. and Nikora, A., "Applying Reliability Models More Effectively," *IEEE Software*, vol. 9, no. 4, pp. 43-52, 1992.
4. Mitchell, M., *An Introduction to Genetic Algorithms*, The MIT Press, 1998.
5. Sy-Yen Kuo, Chin-Yu Huang, and Michael R. Lyu "Framework for Modeling Software Reliability, Using Various Testing-Efforts and Fault-Detection Rates" in IEEE Transactions on Reliability, Vol. 50, NO.3,Sept2001

* * *

Dr. Dhananjaya Reddy
Assistant Professor in Mathematics, Govt. Degree College, Puttur,
andhra Pradesh, India