Prediction Capabilities of Vulnerability Discovery Models

Omar H. Alhazmi, Colorado State University
Yashwant K. Malaiya, Ph. D., Colorado State University

Key Words: Security holes, Operating Systems, Vulnerability, Intrusions, Quantitative models.

SUMMARY & CONCLUSIONS

Quantitative approaches for software security are needed for effective testing, maintenance and risk assessment of software systems. Vulnerabilities that are present in an operating system after its release represent a great risk. Vulnerability Discovery Models (VDMs) have been proposed to model vulnerability discovery and have has been fitted to vulnerability data against calendar time. The models have been shown to fit very well. In this paper, we investigate the prediction capabilities that these models offer by evaluating accuracy of predictions made with partial data. We examine both the recently proposed logistic model and a new linear model. In addition to VDMs, we consider static approaches to estimating some of the major attributes of the vulnerability discovery process, presenting a static approach to estimating the initial values of one of the VDM's parameters. We also suggest the use of constraints for parameter estimation during curve-fitting. Here we develop computational approaches for early applications of the models and examine the predictive capability of the models. We use data from Windows 98, Windows 2000 and Red Hat Linux 7.1. We examine the impact of using a specific constraint when the parameters of the logistic model are estimated Plots for the prediction error are given. The results demonstrate that the prediction error is significantly less when a constraint based on past observations is added. It is observed that the linear model may yield acceptable projections for systems for which vulnerability discovery has not yet reached saturation. The results also suggest that it may be possible to improve the prediction capability by combining static and dynamic approaches, or by combing different models.

1. INTRODUCTION

Security vulnerabilities in operating systems, servers and other software have emerged as a major threat to systems with internet connectivity. A *Software Vulnerability* is defined as "a defect which enables an attacker to bypass security measures" (Ref. 1) or as "a weakness in the security system that might be exploited to cause loss or harm" (Ref 2). A number of vulnerabilities are discovered after an operating system or server is released. The number of undiscovered vulnerabilities and their discovery rates are among the major factors that contribute to the risk.

Vulnerabilities are a subset of software defects. For general software defects, several quantitative methods that use static metrics or software reliability growth models (SRGMS) are available. Because vulnerabilities are defects, we can expect to be able to develop some analogous methods for vulnerabilities. Recently there have been attempts to develop models for describing the discovery of vulnerabilities termed vulnerability discovery models (VDMs) (Refs. 3-6). Such models can be applied to assess the security risks associated with a system. Software developers can integrate such models in the development process to allocate testing and patch development resources in a cost effective manner.

One way to evaluate the applicability of a model would be to see how well it fits the available data (Ref. 7). However a manager may often be interested in making projections when only part of the data is available. In the software reliability engineering field, several researchers have investigated the predictive capability of a number of reliability growth models (Refs. 8, 9). Generally, the two of the measures of interest often are average magnitude of error, and the average bias which measures the tendency of the model to overestimate or underestimate the number of vulnerabilities (Ref. 9). We will examine the prediction capability of vulnerability discovery models using a similar two-component predictability measure. The capabilities of the VDMs will be examined by using actual data from major software systems and plotting the error in estimated values. This will allow us to compare different schemes, identify procedures that are likely to keep the error small, and assess the accuracy of projections.

The available data for several major operating systems suggest that vulnerability discovery tends to manifest different trend during different phases. We consider the question of how we can obtain preliminary estimates of the parameters of the models in the early phases in presence of such shifts. This requires interpretation of the model parameters.

The models proposed by Alhazmi and Malaiya demonstrate very good fit with actual data (Ref. 4). The authors used chi-square (χ^2) goodness of fit tests on several data sets and showed the fit to be significant. They also showed that some of the major attributes such as the number of vulnerabilities and vulnerability density fall within a range for different systems when normalized (Ref. 7). However, a good fit does not necessarily imply good predictive capability. Because trends change, a model that fits well in the early stage may no longer fit when additional data is obtained. Here we evaluate two models using partial data to examine the predictable capabilities of the models. We will then examine

the accuracy of the projection by matching the estimation using later data.

The model parameters are related to the critical points and other observable attributes of the discovery process. We will discuss the significance of these attributes. Non-linear regression often requires initial estimates of model parameter to ensure that the fit is globally optimal. With limited initial data, the parameter values may be unstable. An understanding of the model parameters will assist in interpreting the values computed. Some comparable work has been done with Software Reliability Growth Models (SRGMs) in (Ref. 10), in which the authors have interpreted some of the parameters of the SRGMs.

In the next section we discuss the two models examined. We next introduce the measures used for evaluation and the approaches used. Finally, we present and discuss the results obtained.

2. VULNERABILITY DISCOVERY MODELS

We consider two models: Alhazmi-Malaiya Logistic model (AML) and Linear Vulnerability Discovery model (LVD). The second model may be considered to be a simple approximation.

Alhazmi-Malaiya Logistic model (AML): This model was proposed by Alhazmi and Malaiya in (Ref. 4); AML has shown good fit on a range of operating systems vulnerability data reported in (Refs. 11,12). Below, Figure 1 below shows an example of the fitted model, together with actual data for cumulative number of vulnerabilities Ω for Windows 95. As this figure shows, the vulnerability discovery rate $d\Omega/dt$ increases at the beginning, reaches a steady rate and then starts to decline. The cumulative number of vulnerabilities thus shows an increasing rate at the beginning as the system starts attracting an increasing share of the installed base. After some time, a steady rate of vulnerability finding yields a linear curve. Eventually, as the vulnerability discovery rate starts dropping, there is saturation due both to reduced attention and a smaller pool of remaining vulnerabilities.

The model presumes that the rate of change of the cumulative number of vulnerabilities $d\Omega/dt$ is governed by two factors, as given in Equation 1 below. The first factor ($A\Omega$) increases with the time needed to take into account the rising share of the installed base. The second factor ($B-\Omega$) declines as the number of remaining undetected vulnerabilities declines. While it is possible to obtain a more complex model, this model provides a good fit to the data (Ref. 4).

Let us assume that the vulnerability discovery rate is given by the differential equation:-

$$\frac{d\Omega}{dt} = A\Omega(B-\Omega) \qquad (1)$$

where Ω is the cumulative number of vulnerabilities, t is the calendar time, and initially t=0. *A* and *B* are empirical constants determined from the recorded data.

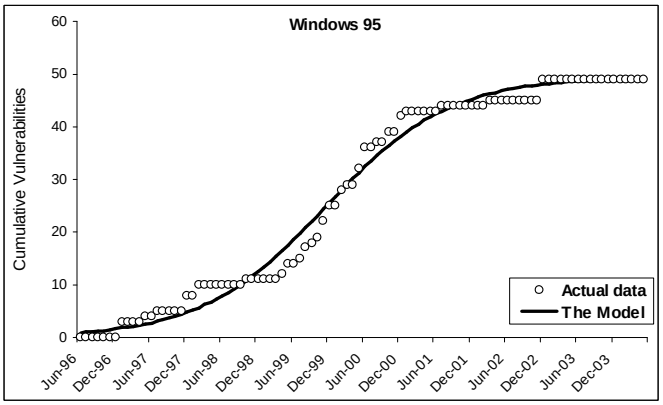

Figure 1 – Alhazmi-Malaiya Logistic (AML) model fitted to Windows 95 vulnerability data set.

By solving the differential equation we obtain

$$\Omega(t) = \frac{B}{BCe^{-ABt} + 1} \qquad (2)$$

where *C* is the integration constant introduced while solving Equation 1. It is thus a three-parameter model given by the logistic function. In Equation 2, as t approaches infinity, Ω approaches B. Thus, the parameter B represents the total number of accumulated vulnerabilities that will eventually be found.

It should be noted that the saturation phase may not be seen in an OS which has not been present for a sufficiently long time. Also, if the initial adaptation is quick due to better prior publicity, in some cases the early learning phase (when the slope rises gradually) may not be significant. This may cause the available data to appear largely linear, and the second model considered later may apply.

A non-linear regression may require an initial estimate of the parameter values. An initial estimate of B may be obtained by noting the size of the software and using the typical vulnerability density values of similar software. The curve bends at two *transition points.* To identify the transition points we take the derivatives of Equation 2 with respect to time t.

$$\frac{d\Omega}{dt} = \frac{AB^3Ce^{-ABt}}{(BCe^{-ABt}+1)^2} \qquad (3)$$

From Equation 3, the highest vulnerability discovery rate occurs at the midpoint of Figure 2 at time

$$T_m = \frac{-\ln\left[\frac{1}{BC}\right]}{AB}$$

The second derivative is:

$$\frac{d^2\Omega}{dt^2} = \frac{2A^2C^2B^5e^{-(ABt)^2}}{(BCe^{-ABt}+1)^3} - \frac{A^2B^4Ce^{-ABt}}{(BCe^{-ABt}+1)^2} \qquad (4)$$

The second derivative exhibits a maximum and a minimum. Thus two transition points in Figure 2 are obtained by equating the third derivative to zero. The transition points occur at times t equal to

$$T_1 = \frac{-\ln\left[\frac{2-\sqrt{3}}{BC}\right]}{AB} \quad \text{and} \quad T_2 = \frac{-\ln\left[\frac{2+\sqrt{3}}{BC}\right]}{AB}.$$

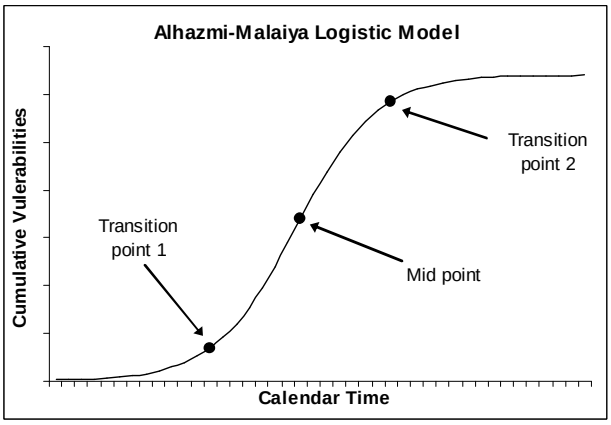

Figure 2 – AML model with the transition points and mid point shown

We can demonstrate that the maximum slope is given by $AB^2/4$, which occurs at $\Omega = B/2$. It can be shown that the two transition points are $2.63/AB$ time period apart. Therefore, the duration of the linear phase decreases as AB grows and increases as AB drops. If we regard B, the total number of vulnerabilities to be constant, we conclude that the parameter A controls the duration of the linear phase. An estimate of the duration between the two transition points can be obtained from the data from prior software systems. We will use this mathematical result later to constrain the range of the regression parameters values.

Parameter C impacts the location of the first transition point. As we can observe from Equation 2, the influence of parameter C weakens as t increases.

Linear Vulnerability Discovery model (LVD): Here we propose a new linear model for comparison. Available data for some systems (Refs. 4, 7) demonstrates the early learning phase is sometimes negligible. This may be due to the similarity between the new release and its prior version which results in a shorter learning phase for the testers to become familiar with the new release. Furthermore, it has been observed that at the saturation phase, the vulnerabilities discovered in the next version include vulnerabilities shared with the modeled version. This can prolong the linear trend even though an increasing market share is taken by the newer version. The linear model (LVD) is an approximation of the AML model and is expected to perform well on software systems that are relatively new. We want to determine whether a linear model can be a reasonable approximation in some cases.

The linear model is given by a linear equation,

$$\Omega(t) = p + qt \qquad (5)$$

where q is the slope and p is a constant factor. The linear model may fit the cases in which much of the data is linear (for example in Figure 3) and largely falls within the two transition points shown in Figure 2. It can thus be seen as an approximation of the AML model.

There are other proposed VDMs that need to be examined similarly in future (Ref. 6).

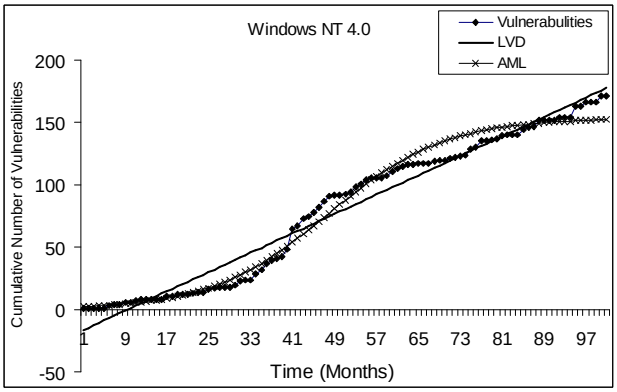

Figure 3 – (AML) and (LVD) models fitted to Windows NT 4.0 vulnerability data set.

3. *PREDICTABILITY MEASURES*

We evaluate predictability by examining the accuracy of prediction using early partial data. We apply the model under consideration for n number of times (t_1, t_2, t_3,…, t_n) with equal calendar time interval periods between estimations. For each t_i, the partial data at t_i is fitted to the model at the best fit using regression analysis to determine the best values for the parameters. The parameters are used to plot the complete prediction curve to obtain the estimated number of vulnerabilities (Ω_i). These estimated numbers of vulnerabilities (Ω_1, Ω_2, Ω_3, …, Ω_n) are compared with the actual number of vulnerabilities (Ω) to determine the normalized estimation error $(\Omega_i - \Omega)/\Omega$. We then take the

average of the normalized error magnitude values to obtain the measure average error (AE). Average bias (AB) is similarly obtained when the sign of the error is also considered (Ref. 9). The average error (*AE*) and average bias (*AB*) are defined as:

$$AE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{\Omega_i - \Omega}{\Omega}\right| \quad (6)$$

$$AB = \frac{1}{n}\sum_{i=1}^{n}\frac{\Omega_i - \Omega}{\Omega} \quad (7)$$

where Ω is the actual number of vulnerabilities while Ω_i is the number of vulnerabilities predicted at time t_i.

An alternative approach is to evaluate the next step prediction is that the data up to t_i is used to estimate the value at t_{i+1} (Refs. 13, 14). However, in cases where models are needed to make longer term projections, predictive capabilities for a variable period are also needed, and AE and AB are more appropriate measures.

Next, the estimation approaches will be previewed and the predictive capabilities of these approaches will be evaluated.

4. PREDICTING THE NUMBER OF VULNERABILITIES

We examine three approaches for estimating the number of vulnerabilities. These approaches are the static approach which uses the program size as the static metric. The dynamic approach uses the vulnerability discovery models (VDMs). The third approach combines the dynamic capabilities with the static estimation to filter out the extreme estimations.

The Static Approach: This approach requires the knowledge of the vulnerability density of comparable software systems. Vulnerability density (Ref. 7) is defined as the number of vulnerabilities per thousand lines of code. The knowledge of vulnerability density of at least one previous software system is a requirement of the static approach. Such previous software systems should preferably have been developed in a comparable manner-i.e., developed by a similar team in the same organization. Then assuming that the vulnerability density is close to the older system, we will have an error range that can be predetermined for addressing factors such as better development practices, more thorough testing before release, fraction of code that is access-related, etc.

The Dynamic Approaches: We have used two different implementations of the AML model in addition to the LVD model.

1) Unconstrained AML: This approach does not place any prior constraints on the parameters; rather the AML model given by Equation 2 is directly applied. The three parameters are estimated using iterative computation utilizing a least squares fit.

2) AML constrained (AML-C): We chose to apply the constraint that the duration between the two transition points is obliged to be within a certain limit. The duration has been shown to be *2.63/AB* for the AML model. Hence, we can limit 2.63/AB between some minimum and maximum values. The choice of the minimum and maximum values determined by the values form the previous software systems. This constraint assumes that the transition points are within the time-frame of the expected lifetime of the software, thus enforcing the S-shape and anticipating the two transition points.

3) Linear (LVD): The linear LVD model given by Equation 5. The parameters values are determined simply by linear regression analysis.

The Combined Dynamic/Static Approach: The Dynamic vulnerability discovery estimation approaches (AML), (AML-C) and (LVD) are applied to estimate the total number of vulnerabilities. However, we can also use the static approach to estimate ceiling and floor values when the dynamic model is applied and some extreme estimation occurs due to some bumps that appears early in the vulnerabilities datasets. That would require the use of vulnerability density data from prior similar systems. In AML and AML-C the static estimation was used to obtain the bounds of the parameter B.

5. TESTING THE ACCURACY OF ESTIMATION

We have applied the approaches mentioned above to estimate the number of vulnerabilities in three systems: Windows 98, Windows 2000 and Linux 7.1. Because the use of static methods and constraints requires the use of prior data, operating systems such as Windows 95 and Windows NT 4.0 cannot be used due to lack of data from applicable previous versions.

Table 1 illustrates the use of prior data for the three systems. The vulnerabilities data is current up to April 2005 (Refs. 7, 12, 13). The static approach will only be able to give estimate only for the final point as a whole figure but not estimates for different points of time. From Table 1 we note that the vulnerability densities for Windows 98 and Linux 7.1 do lie within ±25% of the value of their respective predecessors. However the vulnerability density for Windows 2000 is outside of the ±25% of the value of its predecessor Windows NT. This may be due to the fact that Windows 2000 contains a larger fraction of non-access related code compared with NT.

The Figures 3-5 show the normalized error in estimations at different times for the three systems. The time scale is

given as a percentage of the total time period considered. As expected, projections made using AML or LVD closer to the end time exhibit less error. The shaded region in these figures shows where ceiling and floor values would be if Ω was estimated using the static approach assuming the defect density to be within ±25% of the predecessor.

Table 1 – Estimating the number of vulnerabilities using the static approach

Software Systems	Comparable V_D	Code Size	Est. Ω	With 25±%	Actual Ω
Windows 98	0.0033 (Win95)	18	60	45 to 75	66
Windows 2000	0.0112 (Win NT 4.0)	35	392	294 to 490	172
RH Linux 7.1	0.00694 (RHL 6.2)	30	208	156 to 260	164

The plot for AML error for Red Had Linux 7.1 (Figure 5) demonstrates that the fitted model first shows underestimation until about 45% of the time, followed by overestimation. At about 65% of the time, the projection stabilizes and is reasonably accurate. Use of constraints during the parameter value search can cut out such swings. For Windows 98 (Figure 3) AML generally underestimates while AML-C overestimates. For Windows 2000, both AML and AML-C underestimate. In all cases the error reduces significantly as we approach last part of the life time.

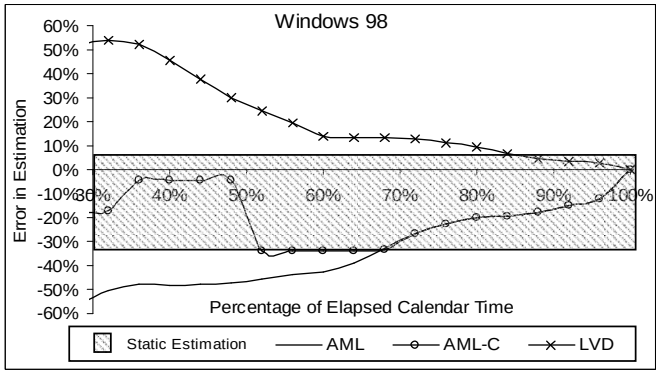

Figure 3 – Estimation error values for Windows 98

The first constraint on B assumed a 25% range (Table 1). This constraint had to be dropped for Windows 2000. The second constraint on Windows 98 and Windows 2000 was chosen to be $24 \leq (2.63/AB) \leq 48$. However, for Red Hat Linux 7.1 the constraint was $18 \leq (2.63/AB) \leq 36$ since Linux versions tend to have shorter lifetime than Windows. The figures show improvements in the accuracy of estimations for the AML model. The improvement is quite significant over the earlier estimations as the impact of extreme projections is minimized by applying the two constraints.

Table 2 shows the average error (AE) and the average bias (AB) given by Equations 6 and 7 have been calculated for the three data sets. We note the significant improvement in both AE and AB when the constraint is used in AML.

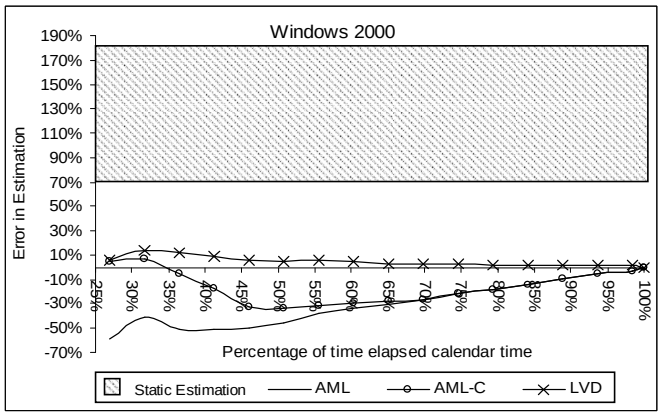

Figure 4 – Estimation error values for Windows 2000

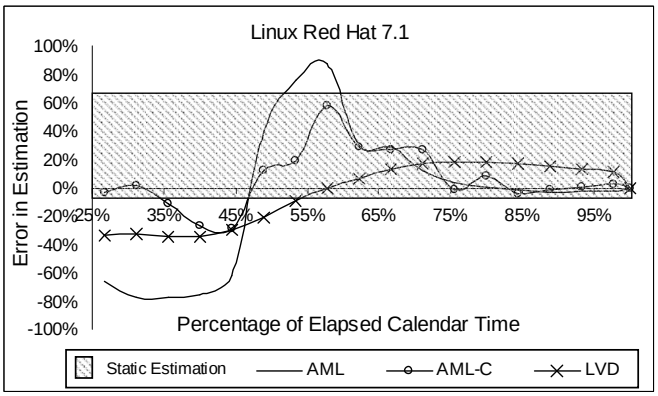

Figure 5 – Estimation error values for Red Hat Linux 7.1

Table 2 – The Average Error (AE) and Average Bias (AB) for the estimations using dynamic models

Approach	AML		AML with constraints		Linear (LVD)	
Software System	AE	AB	AE	AB	AE	AB
Windows 98	43.8%	-43.8%	20.1%	-20.1%	29.1%	29.1%
Windows 2000	29.2%	-29.8%	16.8%	-15.5%	4.6%	4.6%
Red Hat Linux 7.1	37.9%	-5.3%	15.4%	6.3%	19.2%	-4.0%

It is interesting to note that LVD actually does a good job for Windows 2000 and RH Linux 7.1. This is because a noticeable saturation in vulnerability discovery has not yet set in for these two systems.

From Figures 3-5 we can see that a combination of static and dynamic approaches can improve the prediction capability. The estimator has more flexibility in estimating the number of vulnerabilities in a system. Moreover, the estimator can have more confidence in the estimation results if more than one approach is used.

REFRENCES

1. E. E. Schultz Jr., D. S. Brown, and T. A. Longstaff, "Responding to Computer Security Incidents," Lawrence Livermore National Laboratory, ftp://ftp.cert.dfn.de/pub/docs/csir/ihg.ps.gz, July 1990.

2. C. P. Pfleeger, "Security in Computing," Prentice-Hall, 1997.

3. R. J. Anderson, "Security in Opens versus Closed Systems - The Dance of Boltzmann, Coase and Moore," Open Source Software: Economics, Law and Policy, Toulouse, France, June 2002.

4. O. H. Alhazmi and Y. K. Malaiya, "Quantitative vulnerability assessment of systems software," Proceedings of 51st Annual Reliability and Maintainability Symposium, Alexandria,VA, January 2005, pp. 615 – 620.

5. E. Rescola, "Is finding security holes a good idea?" Security and Privacy, *IEEE Security and Privacy*, Vol. 03, No. 1, January-February 2005 pp. 14-19.

6. O. H. Alhazmi and Y. K. Malaiya, "Quantitative vulnerability assessment of systems software," to appear in Proc. International Symposium on Software Reliability Engineering, Chicago, IL, November 2005.

7. O. H. Alhazmi, Y. K. Malaiya, I. J. Ray, "Security Vulnerabilities in Software Systems: A Quantitative Perspective," to appear in the Proceedings of the 19th Annual IFIP WG 11.3 Working Conference on Data and Applications Security, Storrs, CT, August 2005.

8. J. D. Musa, K. Okumoto, "A Logarithmic Poisson Execution Time Model for Software Reliability Measurements," Proceedings of 7th International Conference on Software Engineering, Silver Spring, MD, March 1984,pp. 230-238.

9. Y. K. Malaiya, N. Karunanithi, and P. Verma. "Predictability of software reliability models," *IEEE Transactions on Reliability*, Vol. 41, No. 4, pp. 539–546, December 1992.

10. Y. K. Malaiya and J. A. Denton. "What do software reliability parameters represent?" In *Proc. International Symposium on Software Reliability Engineering*, pages 124–135, Albuquerque, NM, November 1997.

11. ICAT Metabase, http://icat.nist.gov/icat.cfm, April 2005.

12. The MITRE Corporation, http://www.mitre.org, April 2005.

13. A. A. Abdel-Ghaly, P.Y. Chan, and B. Littlewood, "Evaluation of Competing Software Reliability Predictions," IEEE Transactions on Reliability, Vol. SE-12, No. 9, September 1986, pp. 950-967.

14. S. Brocklehurst, P.Y. Chan, B. Littlewood and J. Snell, "Recalibrating Software Reliability Models," IEEE Transactions on Software Engineering, Vol. 16, No. 9, April 1990. pp. 458-470.

BIOGRAPHIES

Omar H. Alhazmi
Computer Science Department
Colorado State University
Fort Collins, CO 80523-1873 USA.

E-mail: omar@cs.colostate.edu

Omar Alhazmi is currently a Ph.D. student at Colorado State University since January 2002. He received his Master's degree in computer science from Villanova University in 2001.

Yashwant K. Malaiya,*Ph.D*
Computer Science Department
Colorado State University
Fort Collins, CO 80523-1873 USA.

E-mail: malaiya@cs.colostate.edu

Yashwant K. Malaiya is a Professor in Computer Science Department at Colorado State University. He received MS in Physics from Sagar University, MScTech in Electronics from BITS Pilani and PhD in Electrical Engineering from Utah State University. He has published widely in the areas of fault modeling, software and hardware reliability, testing and testable design since 1979. He has also been a consultant to industry. He was the General Chair of 2003 IEEE International Symposium on Software Reliability Engineering. He is a senior member of IEEE.