CARATS: A Computer-Aided Reliability Assessment Tool for Software Based on Object-Oriented Design

Chien-Chia Chen[1], Chu-Ti Lin[1], Hen-Hsen Huang[2], Shih-Wei Huang[1], and Chin-Yu Huang[1]
[1] Department of Computer Science
National Tsing Hua University
Hsinchu, Taiwan
[2] Institute of Computer Science and Engineering
National Chiao Tung University
Hsinchu, Taiwan

***Abstract*-With the growth of complexity in the software system, to deliver reliable software products on time becomes a critical issue. Many software reliability growth models (SRGMs) have been proposed in the past three decades. However, most software reliability assessment processes and parameter estimations of models depend on the computations of the general-purposed numerical software. In this paper, we will present a powerful computer-aided reliability assessment tool for software based on object-oriented analysis and design–CARATS. CARATS can use both traditional SRGMs and neural-network methods to assess software reliability. This would greatly help project managers to make decisions during software development life cycle. Due to the characteristics of the special-purposed and object-oriented design, CARATS can analyze the software reliability easily, and is also more flexible to adopt different SRGMs than other existing software reliability analysis tools.**

I. INTRODUCTION

Over the past decade, the deployment of computer systems has grown more dramatically. Software is everywhere, but we need reliable software. The techniques of software reliability analysis developed based on hardware reliability incipiently. As the significance of software grows rapidly, software reliability gets more and more attentions. According to the ANSI definition: Software reliability is defined as the probability of failure-free software operation for a specified period of time in a specified environment [1]. Since 1970, many software reliability growth models (SRGMs) [2-4] have been proposed for estimation of reliability growth of products during software development processes. In general, SRGMs are applicable to the late stages of testing in software development. They can provide very useful information about how to improve the reliability of software products.

From our studies [1, 5-7], there are several tools developed for the estimation of software reliability, such as ROBUST (Reliability of Basic and Ultra-reliable Software sysTem), FRestimate, TERSE, SREPT (Software Reliability Estimation and Prediction Tool), SRETOOLS (AT&T Software Reliability Engineering Toolkit), SRMP (Statistical Modeling and Reliability Program), SoRel (Software Reliability Program), CASRE (Computer-Aided Software Reliability Estimation Tool), ESTM (Economic Stop Testing Model Tool), SMERFS (Statistical Modeling and Estimation of Software Reliability Functions), RGA(Software for Repairable System and Reliability Growth Analysis), DACS's GOEL (An automated version of the Goel-Okumoto NHPP Software Reliability Growth Model), etc.

In this paper, we will present a powerful computer-aided reliability assessment tool for software based on object-oriented analysis and design – CARATS. It can run in the Microsoft Windows™ environment and integrates some existing famous SRGMs, such as the Goel-Okumoto model, the Yamada delay S-shaped model, the Rayleigh model, the power model, the inflection S-shaped model, the Musa-Okumoto model, and so on. Actually, SRGMs are mathematical models that represent software failures as a random process and can be used to evaluate development status during testing. Most of SRGMs depend on some assumptions or distributions. Thus, we also offer an artificial neural-networks-based approach for software reliability growth estimation and prediction in CARATS.

The rest of this paper is organized in the following way: Section 2 gives a brief introduction to the selected software reliability models. Section 3 presents the high-level design of CARATS and describes the main functions of CARATS in estimating software reliability. Besides, we also present some screenshots of several working views of CARATS. Finally, the conclusions are drawn in Section 4.

II. SOFTWARE RELIABILITY GROWTH MODELING

A. Software reliability models

In the following, some selected SRGMs are included in CARATS. CARATS allows selection of one or more

software reliability models to be run, using the failure data shown in the main window.

• *Goel-Okumoto Model.* The Goel-Okumoto model was proposed by Goel and Okumoto in 1979 [2]. This model is characterized by the following mean value function:

$$m(t) = N\{1 - \exp(-rt)\}, \quad (1)$$

where N is the number of initial faults in the software and r is the fault detection rate.

• *Yamada Delayed S-Shaped Model.* The yamada delayed S-shaped model was proposed by Yamada in 1984 and is characterized by the following mean value function [2]:

$$m(t) = N\{1 - (1 + \rho t)\exp(-\rho t)\}, \quad (2)$$

where ρ is the fault removal (failure detection and fault isolation) rate parameter.

• *Rayleigh Model.* The Rayleigh model is a member of the family of the Weibull distribution [8] and is characterized by the following mean value function:

$$m(t) = N\{1 - \exp(-\phi t^2)\}, \quad (3)$$

where ϕ is the failure rate parameter.

• *Power Model.* The power model was proposed by Crow in 1974 [2, 8]. This model is characterized by the following mean value function:

$$m(t) = N \cdot t^{\omega}, \quad (4)$$

where ω is the shape parameter.

• *Inflection S-Shaped Model.* The inflection S-shaped model was proposed by Ohba in 1984 [2]. This model is characterized by the following mean value function:

$$m(t) = N\frac{1 - \exp(-\varphi t)}{1 + \psi \cdot \exp(-\varphi t)}, \quad (5)$$

where φ is the failure detection rate, and ψ is the inflection parameter.

• *Musa-Okumoto Model.* This model is a logarithmic Poisson execution time model based on NHPP with an intensity function decreasing exponentially with expected failures experienced [3].

$$\lambda(\mu) = \lambda_0 \exp(-\theta\mu), \quad (6)$$

where λ_0 is the initial failure intensity and θ is the rate of reduction in the normalized failure intensity per failure. This model incorporates the claim that the repair of early failures reduces the failure intensity more than later ones. The expected number of failures is a logarithmic function of (execution) time:

$$\mu(t) = \frac{1}{\theta}\ln(\lambda_0 \theta t + 1). \quad (7)$$

B. Parameters estimation

Two most popular estimation techniques are *Maximum Likelihood Estimation* (MLE) and *Least Squares Estimation* (LSE). The maximum likelihood technique estimates parameters by solving a set of simultaneous equations and is better in deriving confidence intervals. The method of least squares minimizes the sum of squares of the deviations between what we actually observe and what we expect [1-4]. However, MLE may not always produce parameter estimates, and takes more computation time than LSE. LSE usually requires less computation time to estimate parameters, and always finds a parameter estimate. CARATS provides MLE and LSE techniques to estimate the parameters. For example, we can evaluate the parameters of selected SRGMs by minimizing the least square sum as the following:

$$S = \sum_{k=1}^{n}[m_k - m(t_k)]^2, \quad (8)$$

where m_k is the cumulative number of failures consumed in time (0, t_k], and $m(t_k)$ is the cumulative number of failures estimated by the given model. Differentiating S with respect to the parameters of SRGM, setting the partial derivatives to zero and rearranging these terms, we can solve this type of nonlinear least square problems. On the other hand, the likelihood function is defined as the following:

$$L = \prod_{k=1}^{n}\frac{[m(t_k) - m(t_{k-1})]^{(m_k - m_{k-1})}}{(m_k - m_{k-1})!} \cdot \exp[-(m(t_k) - m(t_{k-1}))]. \quad (9)$$

Taking the natural logarithm of the above equation, we can obtain *lnL*. The maximum likelihood estimates of the unknown parameters of SRGMs can be obtained by differentiating *lnL* with respect to each of the unknown parameters and solving the equations simultaneously.

C. Prediction of software reliability using neural networks

Neural networks are constructed by variable number of neurons, and each one has its bias and weight. In the training process, neurons adjust their biases and weights to reach a given goal, and the final outputs are evaluated by a specific activation function, which is to limit the amplitude of output of a neuron [9]. After training with representative historical data, neural networks can predict the software reliability growth model. Note that although most researchers think that the neural networks approach is a black-box method, we still can explain the neural networks from the mathematical viewpoints of software reliability modeling and implement it in CARATS.

III. DESIGN AND ARCHITECTURE OF CARATS

The architecture of CARATS is shown in Fig. 1 and can be divided into three sub-modules. The format of failure data files used by CARATS can be time between failures (TBF) or failure counts (FC). Note that some of the selected models built into CARATS accept only TBF data, while the others accept only FC data. Simply speaking, the data analyzer will analyze the failure data and this work can be step-by-step done by project creation wizard. The numerical analyzer will proceed with the estimation of models' parameters. Finally, the results analyzer will transform the result of analytic data into visual output. In fact, we can select all models from

a list of the built-in SRGMs that were run. The results of these models will then be plotted in the graphic display window. In this version, the optimal software release time based on reliability and cost criteria can also be computed by CARATS and visually presented to the user. Fig. 2 depicts the related UML class diagram of CARATS.

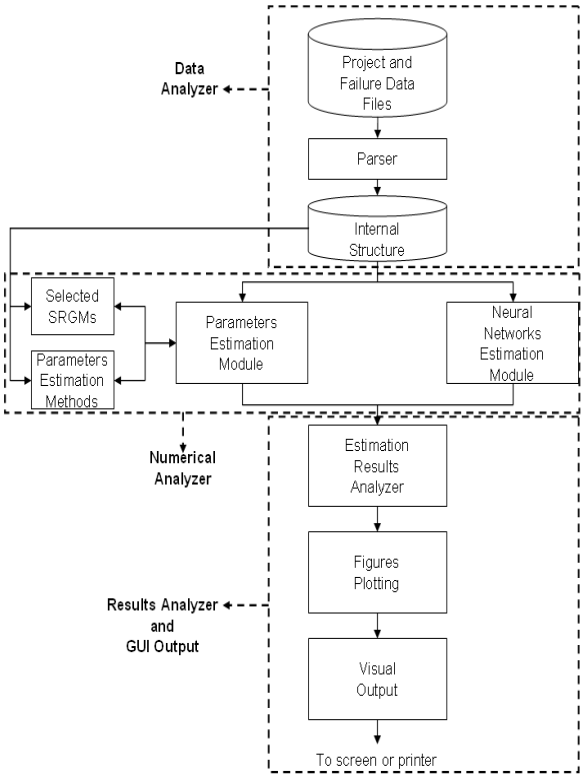


Fig. 1 The architecture of CARATS.

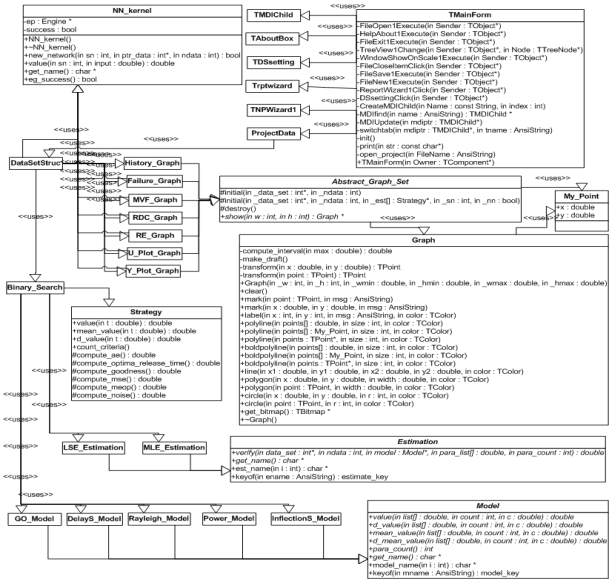


Fig. 2 CARATS UML class diagram.

A. *Project creation*

CARATS supports multiple data sets in a project and each can have different analysis strategies, such as predicting by different models or evaluating model parameters by different estimation methods. Beginners can easily perform this tool by following the project creation wizard of CARATS.

B. *Viewing and adjusting data set settings*

After creating a failure data set and finishing the necessary initial estimation, users can adjust settings by means of the estimation results and read the brief report from the settings dialog as shown in Fig. 3. The "Iterations" in the Fig. 3 is the number of iterations that CARATS spent before models being converged. And the rest five are some criteria to evaluate the model performance [10-12]. Besides, users can also compute and obtain the optimal software release time based on the desired release criteria. The main purpose is to minimize the cost of software development when a desired reliability objective is given.

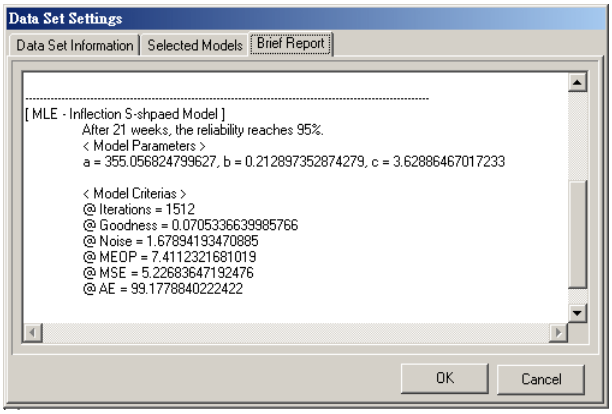
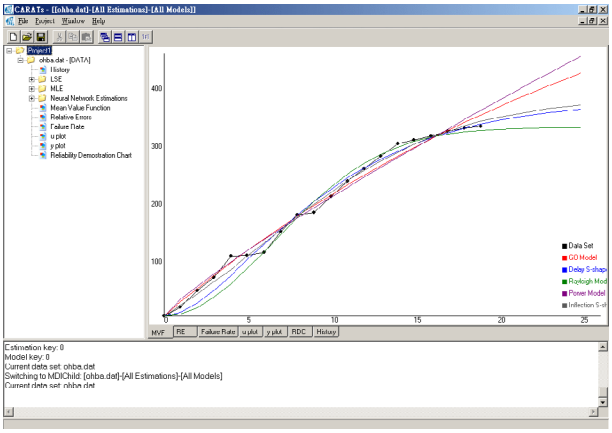


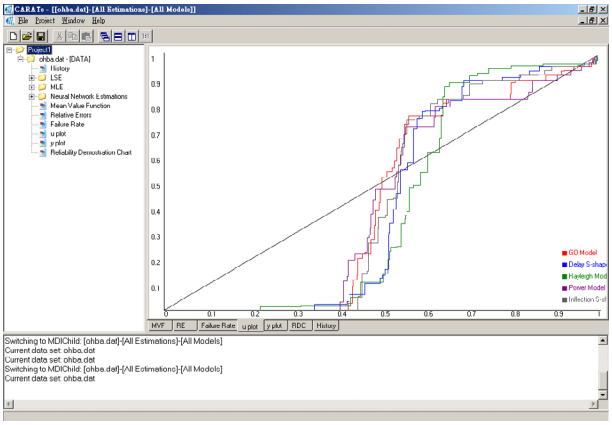
Fig. 3 Brief reports for the selected data set.

C. *Software reliability estimation results*

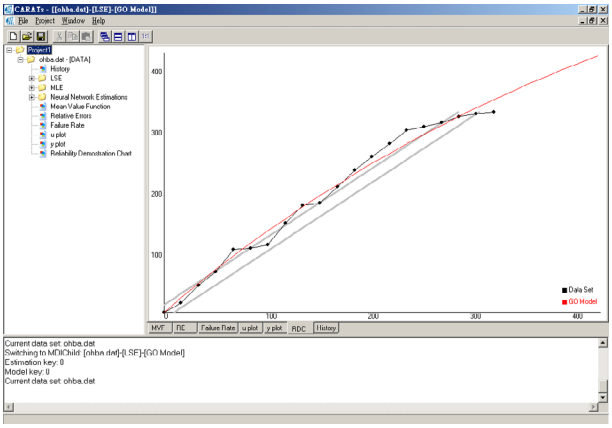
Figures 4(a)-4(c) depict the results of performance comparisons. Note that each simulation result has its own diagram independently. We can also see that Fig. 4(c) is a Reliability Demonstration Chart (RDC) analyzing failure incident data, choosing an operational reliability target against all in-scope errors. The chart can be created using the parameters of Discrimination ratio, Supplier risk, and Consumer risk. These parameters can be set in advance.



(a) Mean Value Functions.



(b) U plots.



(c) Reliability Demonstration Chart.

Fig. 4 Estimation results.

D. Report generator

CARATS also can generate a very detailed reliability analysis report, which contains all of the estimation results and figures. In practice, weekly evaluation report can be used to verify the reliability growth of software product during testing. All of these report items can be customized in CARATS.

IV. Conclusions

In this paper, we present a user-friendly software reliability assessment tool–CARATS, which can be used to measure and predict the reliability of software product. CARATS allows selection of one or more software reliability models to be run, using the failure data shown in the CARATS main window. Furthermore, CARATS can also compute and suggest the optimal release time based on the desired release criteria, such as cost and reliability. In fact, CARATS can indeed provide a total solution for systematic software reliability prediction.

Acknowledgment

This research was supported by the National Science Council, Taiwan, under Grant NSC 94-2213-E-007-087 and also substantially supported by a grant from the Ministry of Economic Affairs (MOEA) of Taiwan (Project No. 95-EC- 17-A-01-S1-038).

References

[1] M. R. Lyu (Editor), *Handbook of Software Reliability Engineering*, McGraw-Hill, New York, NY, 1996.

[2] M. Xie, *Software Reliability Modeling*, World Scientific Publishing Company, 1991.

[3] J. D. Musa, *Software Reliability Engineering: More Reliable Software, Faster Development and Testing*, McGraw Hill, 1999.

[4] Pham, *Software Reliability*: Springer-Verlag, 2000.

[5] P. Hudepohl, S. J. Aud, T. M. Khoshgoftaar, E. B. Allen, and J. Mayrand, "Emerald: Software Metrics and Models on the Desktop", *IEEE Software*, September 1996, pp. 56-60.

[6] N. Li and Y. K. Malaiya "ROBUST: A Next Generation Software Reliability Engineering Tool" *Proceedings of IEEE Int. Symp. on Software Reliability Engineering (ISSRE'95)*, pp. 375-380, Oct. 1995

[7] S. Ramani, S. Gokhale and KS Trivedi, "SREPT: Software Reliability Estimation and Prediction Tool", *Proceedings of the 10th Intl. Conference on Modelling Techniques and Tools (Tools '98),* Palma de Mallorca, Spain, September 1998.

[8] S. H. Kan, *Metrics and Models in Software Quality Engineering*, Addison-Wesley, 1994.

[9] S. Haykin, *Neural Networks A Comprehensive Foundation*, 2nd Edition, Prentice Hall, 1999.

[10] M. R. Lyu and A. Nikora, "Applying Software Reliability Models More Effectively," *IEEE Software*, pp. 43-52, Jul. 1992.

[11] C. Y. Huang, M. R. Lyu, and S. Y. Kuo, "A Unified Scheme of Some Non-Homogenous Poisson Process Models for Software Reliability Estimation," *IEEE Transactions on Software Engineering*, Vol. 29, No. 3, pp. 261-269, March 2003.

[12] S. Yamada and M. Kimura, "Software Reliability Assessment Tool Based on Object-Oriented Analysis and Its Application," *Annals of Software Engineering*, Vol. 8, Numbers 1-4, pp. 223-238, 1999.