AGING TEST AND SOFTWARE RELIABILITY ANALYSIS METHOD FOR PC-BASED CONTROLLER

Jun Yeob Song

Dept. of Intelligence & Precision M/C
Korea Institute of Machinery & Materials
171 Jang-Dong, Yusung-Ku
Daejeon, Korea, 305-343

Ju Su Jang

Moasoft Corp.
Yean Am B/D 502
Karak-Dong 175-14, Songpa-Ku
Seoul, Korea,138-160

ABSTRACT

This paper presents a survey of software reliability modeling and it's application to pre-built software system combined with hardware such as numerical controller based on personal computer systems. Many a systems in these days are much more becoming software intensive and many software intensive systems are safety critical. For this reason, the technique well developed to measure of software reliability is very important for whom to assess such a system. This paper provides a brief idea of method to evaluate such a system's reliability based on hardware performance.

1. INTRODUCTION

A method or procedure for doing an evaluation of system's reliability has been developed and focused only on hardware system and the methodology, in some degree, has been grown up to the reasonable levels. Especially, reliability prediction procedure at the system's design stage is well established in electronic industry as a reliability analysis.

Many a systems, however, such as weapon system, plant, and big equipments are becoming more software-intensive and rapidly standing out its' relative importance. In other words, comparing to the earlier systems, most recent systems are the smaller with less weight and this reason makes it impossible to discern a function of hardware or software. Therefore, it could be a serious problem to do an analysis of system's reliability placing only emphasis on hardware.

This paper, after explaining of PC-based controller and its aging tester configuration briefly, introduces a method of reliability analysis for the system, which is combined PC-based controller with the aging test software program. In fact, most control equipment has a capability to check itself own performance and status, but almost all do not care of why its performance be poor or what should be needed to make the performance higher to the level of end user's requirements. As a simple approach, this paper deals with this problem and makes an alternative measure by using performance data on aging tester.

2. SYSTEM CONFIGURATION AND RELIABILITY MODELS FOR AGING TEST

This study brings system's operational states into focus. By the control flow of the system and field data of which is described in Figure 1, it is possible to assess the system state. That is to say, embedded software called aging test triggers some events to the system and gets operational profile data from the control flows. By using this data, it is available to do an analysis of reliability on running system currently.

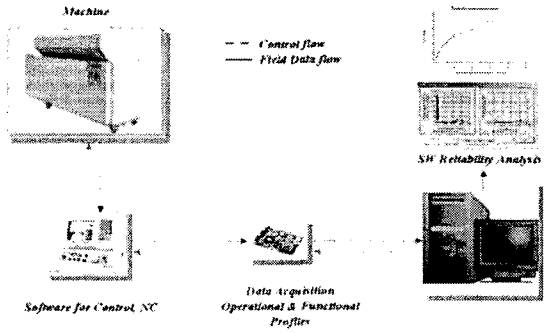

Figure 1: System Configuration for Aging Test

2.1 WORKFLOW DIAGRAM FOR AGING TEST

Aging tester and its function can be obtained from breakdown of Figure 1. There is generic control equipment for machine and software program installed on PC for doing a control of the machine. So every process of the controller is controlled by software program and every process can be measured by performance criterion. With this workflow, a certain event for doing a test for hardware performance or software execution on the hardware can be injected as a test of aging. The concept for this workflow is designated to Figure 2.

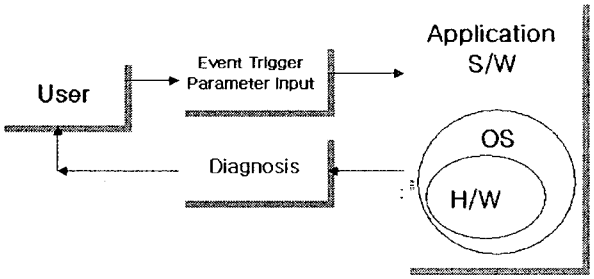

Figure 2: Functional Block Diagram for Aging Test

A trigger for a process makes a diagnosis on hardware and basis data for reliability analysis. For instance, any process performance combined with a hardware component could be a criterion by the basis of performance level, which is set for fault or failure. According to the level of performance, a process can be measured for reliability.

2.2 SOFTWARE RELIABILITY WITH PERFORMANCE

Software reliability represents a user-oriented view of software quality. It relates directly to operation rather than design of the program, and hence it has not static properties but dynamic properties. For this reason, software reliability is interested in failures occurring in a program. So in order to discriminate failures from faults, it is needed to define failure and fault:
Failure means a function of the software that does not meet user requirements. It is an external behavior of the system deviating from that required by its specifications. In other words, failure is something dynamic ,et al., occurring at execution time. It is not a bug or fault. It is that there more general concepts within.
On the other hand, a fault or bug, is a defect in a program that executed under particular conditions and can be a source of more than once failure. A fault may result from an error made by the programmer. Errors occur because of incomplete communication between the people involved in a project or between different times for the same person.

The probability that a given program will work as intended by the user in a specified environment and for a specified duration can be termed as software reliability. A commonly used approach for measuring software reliability is by using an analytical model whose parameters are generally estimated from available data on software failures.
In this paper, failure and fault is the same meaning. Because these things are occurred in a process, performance for the process must be affected by it. So this paper predicts intrinsic failure rate of aging tester by using its source code lines and after that estimates operational failure rate of the tester under hardware performance. Finally the measure of failure rates can be used as system's reliability index.

2.2.1 RELIABILITY MODEL FOR INITIAL FAILURE RATE

There is only one software reliability prediction model developed by Air Force's Rome Laboratory, which can be transformed into reliability measures such as failure rates. A number of factors related to failure density at the earlier life cycle phases can be selected in the following categories:

A. Application type(denoted by A) : Real time system, scientific, information management etc.)
B. Development environment(denoted by D) : Organic, embedded modes etc.
C. Requirement metrics, including anomaly management (SA), traceability(ST) and quality review results(SQ)
D. Implementation metrics, including language type(SL), program size(SS), modularity(SM), extent of reuse(SU), complexity(SX), and standard review results(SR).

With these factors, the initial failure density can be defined as follows :

$$\delta_0 = A \times D \times (SA \times ST \times SQ) \times (SL \times SS \times SM \times SU \times SX \times SR) \quad (1)$$

Once the initial failure density has been found, a prediction of the initial failure rate is made as follows :

$$\lambda_0 = F \times K \times (\delta_0 \times KLOC) = F \times K \times W_0 . \quad (2)$$

Where ,
$KLOC$: Unit of 10^3 lines of source code

The number of inherent failure is $W_0 = \delta_0 \times KLOC$. F is the linear execution frequency of the program, which is the average machine instruction rate divided by the number of object instructions in the program. K is the failure ex-

pose ratio, where $1.4 \times 10^{-7} \leq K \leq 10.6 \times 10^{-7}$ based on historical projects. This initial failure rate is used in software reliability growth model which is described in later section.

2.2.2 RELIABILITY MODELS FOR OPERATIONAL FAILURE RATE

Many models for software reliability analysis have been developed and under developing at present. Some is useful and some is not. In general, the models developed so far can be classified in terms of their attributes as follows:

A. Time domain : Calendar or execution time
B. Category : Finite or infinite number of failures experienced in infinite time. For finite failures category models there are a number of classes according to the functional form of failure intensity in terms of time. For infinite failures category there are a number of families according to the functional form of the failure intensity in terms of the expected number of failures expected.
C. Type : Distribution of the number of failures experienced by time t.

From the various models in Table 1, Actually NHPP(Non Homogeneous Poisson Process) model is used for software test data or operational data set but this model can also be useful for dong an analysis of reliability with performance criterion. This paper uses NHPP model and provides performance reliability criterion. The model is like as follows:

$$\lambda(t) = \alpha \beta e^{-\beta t} \quad (3)$$

$$m(t) = \alpha(1 - e^{-\beta t}) \quad (4)$$

where,

$\lambda(t)$: Failure rate or lower level of performance at time t

β : Occurrence of a failure or lower level of performance

α : Expected total number of detected failure or lower level of performance

$m(t) = \int \lambda(s) ds = E[N(t)]$,

$N(t)$: Cumulative number of errors or at time t and has a form of

2.2.3 RELIABILITY GROWTH MODEL WITH PERFORMANCE FACTORS

This model is based on hardware performance data according to the exponential class in Table 1 and its formulation is as follows:

$$\lambda_0 = \frac{H_s K W_0}{I_s} \text{ failure per CPU second} \quad (5)$$

Finite failure category				
	Poisson type	Binomial type	Other types	
Exponential class	Musa Moranda Schneidewind Goel-Okumoto	Jelinski-Moranda Shooman	Goel-Okumoto Musa Keiller-Littlewood	
Weibull class		Schick-Wolverton		
Pareto class		Littlewood		
Gamma class	Yamada			
Infinite failure category				
	Type 1	Type 2	Type 3	Poisson Type
Geometric family	Moranda			Musa-Okumoto
Inverse linear Family		Littlewood		
Inverse polynomial class		Verral		
Power family				Crow

Table 1: Software Reliability Models

where,

H_s : Main processor speed(instructions/sec)

K : Failure exposure ratio

W_0 : Initial(Inherent) software failure rate ($\delta_0 \times KLOC$)

I_s : $Expansion\ Ratio \times KLOC$

Equation (5) can be applied to real execution or operation defined as

$$\lambda(t) = \lambda_0 e^{-\beta t} \quad (6)$$

where,

$\lambda(t)$: Software failure rate integrated with performance at time t

λ_0 : Initial failure rate integrated with performance from category A

t : CPU execution time

$\beta = B \frac{\lambda_0}{W_0}$

B : Failure reduction factor

3. SOFTWARE PERFORMANCE RELIABILITY GROWTH FOR AGING TEST

For software performance reliability, this paper assumed that there's no limit of performance level. It is only dependent on software code and hardware execution of every running process of controller.

3.1 SOFTWARE PERFORMANCE RELIABILITY OF SINGLE PROCESS

For single process performance reliability growth at the design stage, this paper uses the equation (6) of 2.2.3 section and obtains average data from every 5 minutes (300 seconds) running. Its result is as Table 2.

Table 2: Reliability Growth for Single Process

Initial failure rate(λ_0)	0.00023 / CPU second
$\lambda(300)$	0.000762 / CPU second

3.2 SOFTWARE PERFORMANCE RELIABILITY OF MAJOR 5-PROCESSES

5 major processes for aging test, assumed that every process is running independently and same conditions to single process case, its result is as Table 3. Figure 3 provides real time performance reliability growth information of the processes.

Table 3: Reliability Growth for Major 5 Processes

Initial failure rate(λ_0)	0.00093 / CPU second
$\lambda(300)$	0.001361 / CPU second

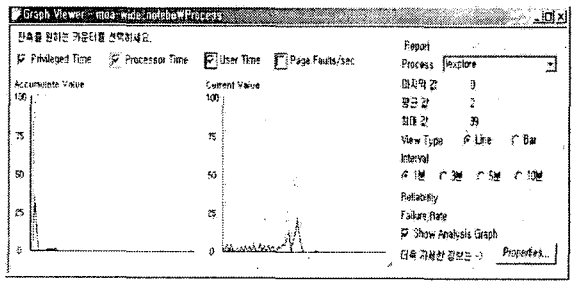

Figure 3: Performance Reliability Growth for Processes

4. OPERATIONAL PERFORMANCE RELIABILITY FOR AGING TEST AND DIAGNOSIS

4.1 PARAMETER ESTIMATION FOR EACH PERFORMANCE LEVEL

By using same code and performance environment on the aging tester, this paper assumed that every process performance do not comply to the requirement level is a failure. Therefore, its performance level is dependent on hardware with software execution environment. The parameters for NHPP model is as Table 4:

Table 4: Parameter Estimation of Execution Result

Performance	α	β
80% or less	10.84	0.0016
90% or less	1.24	0.0021
95% or less	1.01	0.00126

4.2 DIAGNOSIS

With the result of performance reliability analysis, aging tester can extract some symptom, where a process does not meet the requirement. In fact, this information is very useful for whom to control some machine connected aging tester to evaluate the machine performance. Figure 4 generates operational performance profile and its performance reliability.

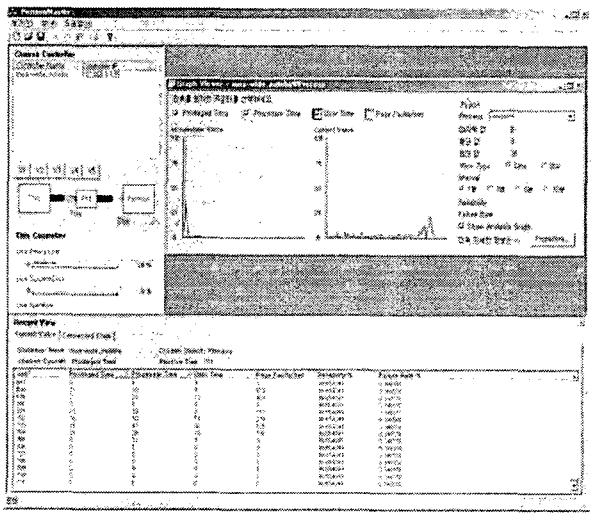

Figure 4: Operational Performance Profile

5. CONCLUSION

Although this paper does not include any hardware information and its reliability measures but briefly introduces some points for software reliability analysis and its important measures. In fact, there have been many models for software reliability analysis developed in the last two decades and invalidated because of invalid assumptions, inaccuracy, costliness, and impracticality. Many of models developed to nowadays are based on error counts but do not

consider any hardware factors where application software installed. For these reasons, the future of study should have another direction of method including hardware system and its environment factors.

REFERENCES

Lyu, Michael R., 1996. *Handbook of Software Reliability Engineering, McGraw Hill.*

Musa, J., 1999. *Software Reliability Engineering, McGraw Hill.*

Broekman, B and Notenboom, E., *2003. Testing Embedded Software, Addison Wesley.*

Myers, G. J., 1976. *Software Reliability Principles and Practice, Wiley Interscience.*

Fenton, N. and Littlewood, B., 1991. *Software Reliability and Metrics, Elsevier.*

Song, J. Y. and Park, H. Y., et al., 2004. *Development of Reliability Analysis Technology for High Speed and Intelligence System. Technical Report BSM 523-1112.M, Korea Institute of Machinery and Materials.*

Jang, J. S., Song, J. Y. and Shin, J. H., 2003. *A Study of Software Reliability on Numerical Control Program. In Proceeding of 2003 annual Military Science and Technology : 297-300.*

Jang, J. S., Song, J. Y. and Shin, J. H., 2003. *A Survey of Software Reliability Model. In Proceeding of 12th Guided Weapon System : 129-132.*