Abstract—Software fault-tolerance techniques have been widely used in computing systems to achieve high level of quality. Rejuvenation, a modern software fault-tolerance technique, has attracted a large number of researchers in software engineering area. Evaluating the effectiveness and feasibility of this technique becomes extremely important in selecting, comparing and applying it in actual software systems. The study of important quality-attributes is the scientific basis for assessing the performance of software fault-tolerance techniques. This paper presents availability, reliability, safety evaluation of rejuvenation systems. Derived mathematical relations between failure probabilities and modeling parameters enable us to gain a great deal of quantitative results.

Availability; reliability; safety; software fault tolerance; rejuvenation; preventive maintenance

I. INTRODUCTION

Today, information technologies along with software systems have been used widely in many fields of life, economic and society. However, software systems, from the design and construction to the operating activities, are always including the potential risks of fault infections. These errors can be generated themselves from the program or from the environment in which these software applications operate. Concept: “Software fault tolerance is the feature which allow the computer keep executing with the presence of defects” has provided a new direction to reach the goal of improving reliability and quality characteristics of the application systems.

Since the concept of fault-tolerant software was presented so far, many techniques have been proposed and applied successfully in practice. Rejuvenation (preventive maintenance - PM) is a new software fault tolerance technique, which Y. Huang was proposed in 1995 [2] and now it has attracted the interest and the research of many scientists [2], [3], [5], [4]. Assessing effectiveness and feasibility of this technology becomes extremely important in choosing, comparing and applying it to practical software systems.

Reliability, safety, and availability are some of the most basic attributes of dependability, so they decide the quality and the performance of software. Therefore, these attributes are used in almost all the surveys about software preventive maintenance techniques. Moreover, evaluating these properties also provides the accurate basis for costs and time estimations to build rejuvenation systems.

There are two main approaches to analysis the quality of fault-tolerant software: practical testing and modeling. Results of practical testing are more believable than those from modeling, which is more well-known. However, that testing can only establish the presence of errors but cannot assure their absence. Also, for highly dependable systems, the testing method is not always feasible and tends to be expensive to implement and then, to obtain statistically significant results.

For complex fault-tolerant systems, modeling and prediction have become an integral part of the system design processes. Thus, an early analysis during system development is possible and it provides information regarding whether the current design will be able to meet the quality requirements, and which parts of the design are the weak points with respect to these quality properties. Based on these arguments, practical tests may be designed in order to prove the assumptions of the model, giving a cost effective method for quality validation of the system. Interaction between modeling and experimentation can help us both in understanding the problems and yielding specific numerical measures.

II. SOFTWARE FAULT TOLERANCE TECHNIQUE – REJUVENATION

When software applications execute continuously for long periods of time (scientific and analytical applications run for days or weeks, servers in client-server systems are expected to run forever), the processes corresponding to the software in execution age or slowly degrade with respect to effective use of their system resources. The causes of process aging are: memory leaking, unreleased file locks, file descriptor leaking, data corruption in the operating environment of system resources, etc. Process aging will affect the performance of the application and eventually cause the application to fail [2].

This work is partly support by Research Grant B2010-01- 411 -TD, funded by Ministry of Education and Training of Vietnam

Nguyen Huu Nam Duong
Department of Computer Engineering
SOICT, Hanoi university of Technology
Hanoi, Vietnam
duongnhn@soict.hut.edu.vn

Huynh Quyet Thang
Department of Software Engineering
SOICT, Hanoi university of Technology
Hanoi, Vietnam
thanghq@soict.hut.edu.vn
If an application is developed in a perfect development environment and it operates correctly in the scenario work, the implementation process associated with this application will not be aging. However, practical software systems rarely are perfect. Therefore, their processes will be aging in the operating environment. The process aging and the software aging are fairly different. Software aging is related to source program which will be inappropriate when requirements and maintenance are changing after many years. On the contrary, process aging is related to the decrease of application functions after several working days or weeks.

Software preventive maintenance (software rejuvenation) is a concept related to periodically reboot the system and turn the application back to the initial clean status after each maintenance [2], [3]. Here, we have an overview figure describing four states of the system when rejuvenation technique is applied. (Fig. 1)

![Status model of Rejuvenation system](image)

Figure 1. Status model of Rejuvenation system

### III. THE MENTIONED SOFTWARE PROPERTIES

#### A. Reliability

Reliability is an attribute which can be evaluated quantitatively through the probability that service system is provided accurately as specifications. Reliability is one of the most important properties to evaluate software quality. The concept of software reliability can be defined as [2]: "Reliability is the probability of software which operates in a fault-free way in a time period in a given environment, with a particular purpose".

#### B. Availability

Availability is an attribute of dependability and can be quantitatively assess. The availability of system is the probability that system can work and provides services to users when requests are encountered.

On the other hand, the differences between reliability and availability depend not only on individual system but also on the time needed to repair fails which destroys the system. Accurately, we can define: "The availability of software is the probability that this software at a time will work and be able to provide services when it is requested" [2].

### C. Safety

The safety is the probability that there are no serious accidents which occurred under the given conditions in a specific time. Software safety is an important attribute of the system safe-critical [2]. The safe-critical systems will cause heavy damage to people or environment if they encounter a failure.

### IV. GENERAL METHOD TO EVALUATE FAULT-TOLERANT SYSTEMS

#### A. Markov model implementation

The authors K. S. Trivedi and Goseva-Popstojanova have proposed to use Markov model in evaluation fault-tolerant system [2], [3], [5], [8]. In the first step, a Markov state map is developed by identifying the status of the system and the transition between states.

#### B. Building Chapman-Kolmogorov equations

The second step converts Markov state chart, which has been developed, to a collection of the Chapman-Kolmogorov equations to find the matrix of transition state probability of the system.

#### C. Solving Chapman-Kolmogorov equations

Solving Chapman-Kolmogorov equations is relatively complex. Some current resolutions such as analytics analysis, Laplace-Stieltjes transform or use ODEs in Matlab can simplify this task.

#### D. Calculating and assessing the attributes of the fault-tolerant software

With each specific system, the software attributes will be evaluated according to specific parameters.

### V. RELIABILITY, AVAILABILITY AND SAFETY EVALUATION OF REJUVENATION SYSTEMS

#### A. System Status Implementation

Supposed that software system is built following server – client model with a queue containing finite number of requests. System exists only errors, which seriously affecting the functionality of total system and we are not interested in the other errors, which are considered as they occur and are repaired immediately, do not decrease the reliability of the system. When the system encounters serious error, all requests will be canceled and the system will become unsafe (state B), then evoke the error-recovery process.

![Status and behavior of rejuvenation system](image)

Figure 2. Status and behavior of rejuvenation system
We consider two different policies which determine the time to perform preventive maintenance:

1) Policy I – Purely time based: Preventive maintenance is initiated after a constant time $\delta$ has elapsed since it was started (or restarted).

2) Policy II – Instantaneous load and time based: The actual preventive maintenance interval is determined by the sum of preventive maintenance wait and the time it takes for the queue to get empty from the point onward.

Let $\pi_i$ be the steady state probability of the software being in state $i \in \{A, B, C\}$. From the well know relation $\pi = \pi P$, we have:

$$\pi = \begin{bmatrix} \pi_A \\ \pi_B \\ \pi_C \end{bmatrix} = \begin{bmatrix} 1 \\ \frac{1}{2} P_{AB} \\ \frac{1}{2} P_{AC} \end{bmatrix}$$

Let $U$ be a random variable denoting the sojourn time in state $A$ with its expectation $E[U]$. The steady state availability can be given as:

$$A_{SS} = \Pr\{\text{system is in state } A\} = \pi_A E[U]$$

Substituting the values of $\pi_A$, $\pi_B$, $\pi_C$:

$$A_{SS} = \pi_A E[U]$$

The steady state safety can then be obtained as followed:

$$S = 1 - \Pr\{\text{system is in state } B\}$$

In policy I, system is surveyed in the period $(0, \delta]$, so average reliability can be obtained as:

$$R_{av} = \frac{\int_0^\delta \left( \sum_{i} p_i(t) \right) dt}{\delta}$$

In policy II, system is surveyed in the period $(0, \infty)$, so average reliability can be obtained as:

$$R_{av} = \lim_{T \to \infty} \frac{\int_0^T \left( \sum_{i} p_i(t) \right) dt}{T}$$

### Table I. Meaning of Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_{AB}$</td>
<td>Probability when changing from state A (available) to state B (recovery)</td>
</tr>
<tr>
<td>$P_{AC}$</td>
<td>Probability when changing from state A (available) to state C (rejuvenation)</td>
</tr>
</tbody>
</table>

---

**1) Policy I**

![Markov process with policy I](image)

For $\mu(\cdot) = \mu(L(t))$ and $\rho(\cdot) = \rho(L(t))$, $L(t)$ is defined by:

$$L(t) = \int_0^t \sum_i c_i p_i(\tau) d\tau$$

The expected sojourn time in state A is given by:

$$E[U] = \int_0^\delta \left( \sum_{i=0}^K p_i(\delta) \right) dt$$

**2) Policy II**
In this case, we need to distinguish between \( t \leq \delta \) and \( t > \delta \), as policy II assumes that preventive maintenance will be initiated if and only if the buffer is empty after time \( \delta \) has elapsed. Similar to policy I, on step transition probability \( P_{AB}^{i} \) is computed by solving the system of ODEs at \( t = \infty \) and is given as: \( P_{AB}^{i} = \sum_{i=0}^{K} p_{i}(\infty) \) then \( P_{AC}^{i} = 1 - P_{AB}^{i} = p_{0}(\infty) \).

The mean sojourn time in state A is now given by:

\[
E[U] = \int_{t=0}^{\infty} \left( \sum_{i=1}^{K} p_{i}(t) \right) dt + \int_{t=0}^{\delta} \left( \sum_{i=1}^{K} p_{i}(t) \right) dt
\]

\[
= \int_{t=0}^{\delta} p_{0}(t) dt + \int_{t=0}^{\delta} \left( \sum_{i=1}^{K} p_{i}(t) \right) dt.
\]

### B. Reliability, Availability and Safety evaluation

The models are solved for multiple values of \( \delta \) and optimum value is determined. Using programming solution tool in Matlab, we can estimate Chapman-Kolmogrov equations, thereby simulating the variability of \( A_{ss}, P_{loss} \) and the upper bound of response time \( T_{res} \) with system parameters. Model parameters: \( \gamma_{r} = 0.85 \text{ (h)}; \lambda = 6.0 \text{ (h}^{-1}); K = 50; \text{MTTF} = 240 \text{ (h)} \text{ where } h = \text{hours.}

1) Experiment 1

In this experiment, \( \gamma_{r} \) is varied to ascertain the effect on the measures and on optimal \( \delta \). Service rate and failure rate are assumed to be functions of real time, i.e., \( \mu(.) = \mu(t) \) and \( \rho(.) = \rho(t) \), where \( \rho(t) = \beta \alpha t^{-\alpha-1} \), which is the hazard function of Weibull distribution.

\( \alpha \) is fixed at 1.5 and \( \beta \) is calculated from \( \alpha \) and the MTTF as:

\[
\beta = \left[ \frac{\Gamma \left( 1 + \frac{1}{\alpha} \right)}{\text{MTTF}} \right]^{\alpha}
\]

and \( \mu(t) \) is defined as:

\[
\mu(t) = \begin{cases} 
\mu_{\text{max}}, & \text{for } t \leq \alpha \\
\frac{\mu_{\text{max}} - \mu_{\text{min}}}{\mu_{\text{min}}}, & \text{for } t > \alpha
\end{cases}
\]

where \( \alpha = \frac{\mu_{\text{max}} - \mu_{\text{min}} \cdot \text{MTTF}}{\mu_{\text{max}}}; \mu_{\text{max}} = 15 \text{ h}^{-1}; \mu_{\text{min}} = 5 \text{ h}^{-1}. \)

Under both polices, it can be seen that the higher the value of \( \gamma_{r} \), the lower is the availability for any particular value of \( \delta \).

### Figure 5.

Availability under policy I

### Figure 6.

Availability under policy II

### Figure 7.

Safety under policy I where \( \gamma_{r} = 0.35 \)

Fig. 7 and Fig. 8 show that safety will decrease when increasing the value of parameter \( \delta \), while safety increases when raising the value of parameter \( \gamma_{r} \). Since then, we can comment that under the policy I, the sooner the preventive maintenance will be conducted, the safer the system will be.
The safety of system under policy II will increase with the decrease of $\gamma_t$ (Fig. 9). However, the dependency is relatively small. In addition, the safety will reduce rapidly along with the increase of $\delta$ to a threshold (marked on the drawings) and then will be almost unchanged. From theoretical calculation, we can see the average reliability of the system does not depend on $\gamma_t$. Therefore, we fix the value $\gamma_t = 0.55$ and survey the influence of the reliability on the time to wait to perform the preventive maintenance $\delta$.

The average reliability of system under policy I rises with the decrease of parameter $\delta$ (Fig. 10). Clearly, under policies I, the sooner the preventive maintenance is conducted, the higher the level of reliability of system is kept. Under policy II, the reliability is calculated throughout the time domain. It can see that the reliability will increase along with the increase of $\delta$ to a threshold and then be kept stable.

2) **Experiment 2**

In this experiment, $\gamma_t$ is fixed at 0.15; $\alpha$ is assigned values of 1.0, 1.5 and 2.0, respectively.

For $\alpha = 1$, the time to failure has an exponential distribution, which, because of its no-memory property, contradicts aging. It is better not to perform Rejuvenation in this case if the objective is to maximize availability. For other two values of $\alpha$, however, rejuvenation maximizes availability at certain $\delta$. For a specific policy, the bigger the failure density, i.e., higher the value of $\alpha$, the higher is the maximum steady state availability. Also, with higher values $\alpha$, this maxima occurs at lower values of $\delta$.

Fig. 12 and Fig. 13 show that the higher the failure density $\alpha$ is, the higher the value of safety will be in the low value domain of $\delta$. On the other hand, when $\alpha$ increases, the ability in which system in the state A will decreases, so the reliability of system will be improved. In addition, the average reliability of system under policy II will increase a threshold (marked on the drawings) along with the increase of $\delta$, and then stops and be kept relatively stable. Meanwhile, the value of parameter $\delta$ does not influence the reliability of the system any great deal.

Safety under two policies where $\alpha$ varies

Figures 8 and 9 illustrate the safety under different policies as $\gamma_t$ varies. Figure 10 shows the reliability under two policies where $\gamma_t = 0.35$.
VI. ACTUAL EXPERIMENT IN OPAC SYSTEM

A. System Description

OPAC system is used in the electronics library Ta Quang Buu to serve the needs of searching documents, books etc. by students and professors in Hanoi University of Technology. This system was built in a Client – Server model and was provided the services through a website http://opac.hut.edu.vn/.

To improve the quality of this system, a strategy to use preventive maintenance technique (Rejuvenation) has been applied to server. Accordingly, at the same time when the system performs backup data process, the preventive maintenance is conducted by rebooting the server.

B. Testing and results

Here are the practical experiments which were conducted on the Electronics Library Ta Quang Buu:

<table>
<thead>
<tr>
<th>Actual parameter / features</th>
<th>Experiment I</th>
<th>Experiment II</th>
</tr>
</thead>
<tbody>
<tr>
<td>System with preventive maintenance (rejuvenation)</td>
<td>Not apply</td>
<td>Apply</td>
</tr>
<tr>
<td>Conduct time</td>
<td>May, 2009</td>
<td>August, 2009</td>
</tr>
<tr>
<td>Total survey-system time</td>
<td>72 hours = 259200 seconds</td>
<td>960 hours = 345600 seconds</td>
</tr>
<tr>
<td>Total number of request</td>
<td>28 472</td>
<td>48 1394</td>
</tr>
<tr>
<td>Total no-respond time</td>
<td>27 seconds</td>
<td>359 seconds</td>
</tr>
<tr>
<td>Availability of system</td>
<td>(259200-27) / 259200 = 0.999895833</td>
<td>(345600-359) / 345600 = 0.999896123</td>
</tr>
</tbody>
</table>

From the experimental results, there was not an obvious different between the quality properties (availability) of the Rejuvenation server and the no-rejuvenation server. As it was mentioned in previous part, the data number in the above table also shows that the quality of the system will be decreasing through operating hours and the sooner we make a rejuvenation, the better quality the system can provide. However, we can still realize that the system (in experiment II) which applies the preventive maintenance has higher quality than those of the system (in experiment I) which does not apply this technique.

VII. CONCLUSIONS

Applying the theory of mathematical Markov model, the theory of rejuvenation, we have built a general model and has applied it to evaluate the software attributes of rejuvenation systems.

In the future, based on the relationship between these software attributes and fault tolerance techniques, the research will be further developed. From the evaluation of the software attributes of fault-tolerant systems, we can deliver the construction cost in rejuvenation-applied software. In addition,
we can use the obtained results in the evaluation of the software attributes of the rejuvenation systems to study about client-server systems with K queues (K > 1) and distributed fault-tolerant software systems.

REFERENCES


BIOGRAPHIE


He is working as a lecturer at Department of Computer Engineering, School of Information & Communication, Hanoi University of Science & Technology since 9/2009. His current research interests are software quality evaluation, embedded systems, electronic communication.

Mr Duong has won the Certificate of Merit for Excellent Graduation, Hanoi University of Science & Technology in 2009.

Huy Quyet Thang, Hanoi, 24/05/1967.

He received his Master and Phd. degree in computer Science from the Varna Technical University, Bulgaria, in 1990 and 1995, respectively.

He is working as an associate professor at Department of Software Engineering, School of Information and Communication Technology, Hanoi University of Science and Technology since 10/1998.