Method for Evaluation of Quality Properties in SaaS Rejuvenation using Markov Model

Huynh Quyet-Thang, Nguyen Ngoc-Dung, Nguyen Hung-Cuong
School of Information and CommunicationTechnology
Hanoi University of Science and Technology
Hanoi, Vietnam

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.**

Keywords—software reliability, software availability, software safety, Markov chain

I. INTRODUCTION

Nowadays, computer science has more and more application in human life, from economic to society, from education to medicine. So there is a requirement that developer has to build a high quality system to support user work. Most regular properties of software quality are reliability, availabilityand safety, that are studied in many fields: in component based by Larsson [1], consider maintenance and security issues by Xiong [2], base on properties and architecture of system by Roshandel [3], etc… The reliability relates with the correctness of result of work, whereas the availability ensures that system is ready to serve and the safety minimize the probability that a serious accident occurs in running time. There are two main approaches to analysis those properties 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, 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.

Since the concept of fault-tolerant software was presented so far, many techniques has 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], [4], [5]. Assessing effectiveness and feasibility of this technology becomes extremely important in choosing, comparing andapplying it to practical software systems.

Markov chain (more specific: discrete-time Markov chain) is a stochastic math system, containing a set of finite (or countable) number of possible states and a set of transitions between two of them. Given the past and present circumstance of Markov chain system, future behavior only depends on the present state and not on the past one. This model has large number of applications in natural science.

There are several techniques to evaluating quality properties of computer system with different approaches [6,7,8]. Thus, based on advantages of Markov chain model, this study introduces a model and applies it to evaluate the quality attributes of rejuvenation-software systems.

This paper is organized as follow: after this introduction section, section 2 explains definition of three aspects of software quality: reliability, availability and safety. Next, section 3 introduces rejuvenation - a software fault-tolerance technique. Section 4 proposes a method for evaluation those quality properties in rejuvenation systems using Markov model. Then section 5 shows experimental result of proposed method in simulation experiments. In this section we present also the experiment results in real system – BKOJ software, run as SaaS in the BKCLoud system. Section 6 discusses some related and future problem to extend current work.

II. BASIC ASPECTS OF SOFTWARE SYSTEM QUALITY

Larsson [1] introduce dependability is the main qualityattribute of safety-critical system development. Althoughsoftware quality has complex meaning and depends on manyproblems, there are three aspects that are discussed following.

A. Reliability

Definition of reliability is based on probability that a system will fail in a specific period of time in given context and can be reflected by mean time to failure ($MTTF$) equation:

$$\text{Reliability}(A) = \frac{1}{P_f(A)} \quad (1)$$

While A is a module and $P_f(A)$ is a probability that this module fails per time unit. Being an important property of system, reliability is focused widely by researchers: Roshandel [3], Hoang P. [4], etc... It often used as an indicator for software release policy and can be got by using practical analysis of math models. Roshandel [3] introduce technique to calculate system reliability from this property of element. Hoang P. [4] summaries some statistical models and focuses on NHPP models.

B. Availability

Although problems of availability is larger than reliability,Xiong [2] notes that availability is a probability that systemisready-for-work in given time and given environment. Relate with reliability attribute, Larsson [1] introduce formalcalculation:

$$\text{Availability}(A) = \frac{MTTF(A)}{MTTF(A) + MTTR(A)} \quad (2)$$

While $MTTR$ is mean time to repair. This attribute of systemhas high commercial contribution: users will satisfy if theycan use product service at every time. The difference between reliability and availability is that availability dependson the dynamic state of the system.

C. Safety

Larsson [1] consider software safety as an attribute thatrelates with the interaction between the system and the environment. It is a full-system property, either a component or an assembly property. Safety depends on where and howthe system is deployed, in other way is dependent on the environment of system, so a top-down approach should be used in analyzing process. Safety of system is more important inthe safe-critical systems, which will cause heavy damage topeople or environment if they encounter a failure.

D. General method to evaluate fault-tolerantsystems

Authors K. S. Trivedi and Goseva-Popstojanova [5, 6]have proposal to use Markov model in evaluating fault-tolerantsystem:
Step 1. Markov model implementation
In the first step,the Markov state map is being developedby identifying the status of the system and the transition between states.
Step 2. Building Chapman-Kolmogorov equations
In the second step, the Markov state chart, whichhas been developed, is being converted to a collection of the Chapman-Kolmogorov equations to find the matrix of transition state probability of the system.
Step 3. Solving Chapman-Kolmogorov equations
Solving Chapman-Kolmogorov equations is relativelycomplex. Some current resolutions such as analytics analysis, Laplace-Stieltjes transform or use ODEs inMatlab can simplify this task.
Step 4. Calculating and assessing the attributes of the fault-tolerant software
With each specific system, the software attributes will be evaluated according to specific parameters.This is general mechanism when using Markov chain in modeling. Real application depends on properties of environment, context and meaning.

III. SOFTWARE REJUVENATION

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

If an application is developed in a perfect developmentenvironment and it operates correctly in the scenario work,the implementation process associated with this applicationwill not be aging. However, practical software systems rarelyare perfect. Therefore, their processes will be aging in theoperating environment. The process aging and the softwareaging are fairly different. Software aging is related to sourceprogram, which will be inappropriate when requirements andmaintenance are changing after many years. On the contrary, process aging is related to the decrease of applicationfunctions after several working days or weeks.

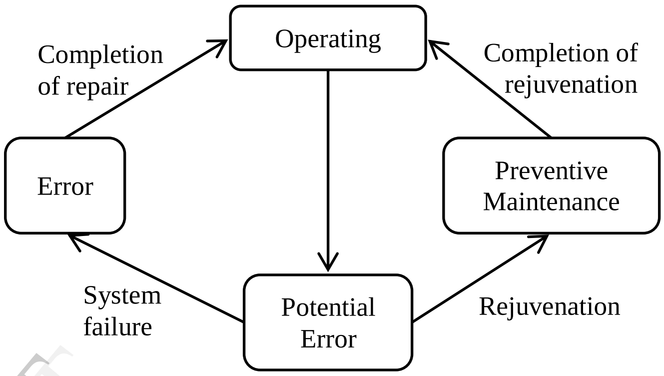

Figure1.Status model of Rejuvenation system

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 rejuvenationtechnique is applied (Figure**1**).

IV. PROPOSED METHOD FOR EVALUATION OF RELIABILITY, AVAILABILITY AND SAFETY OF REJUVENATIONSYSTEMS

A. System Status Implementation

Used symbols are showed in table 1 as followed:

Table 1.MEANING OF SYMBOLS

Symbol	Meaning
P_{AB}	Probability when changing from state A(available) to state B (recovery)
P_{AC}	Probability when changing from state A(available) to state C (rejuvenation)
$p_i(t)$	Probability when having i transactions inqueue at the time t
γ_f	Estimated time to complete the process ofrecovering from errors
γ_r	Estimated time to complete the process ofpreventive maintenance
U	Time when system is in state A
λ	Speed of transactions to system
$\mu(.)$	Speed for serving
$\rho(.)$	Failure rate
$L(t)$	Mean processing time since the system rejuvenated last

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

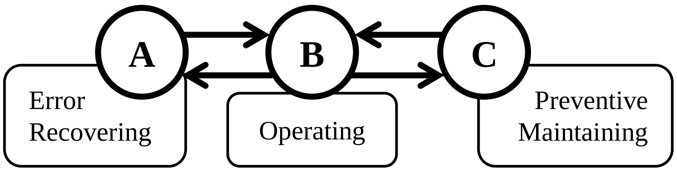

Figure2.Status and behavior of rejuvenation system

We consider two different policies, which determine the timeto perform preventive maintenance:

- Policy I. Purely time based: Preventive maintenance is initiated after a constant time δhas elapsed since it was started (or restarted).
- Policy II. Instantaneous load and time based: The actualpreventive maintenance interval is determined by the sumof preventive maintenance wait and the time it takes forthe queue to get empty from the point onward.

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

$$\pi = [\pi_A, \pi_B, \pi_C] = [\frac{1}{2}, \frac{1}{2}P_{AB}, \frac{1}{2}P_{AC}] \tag{3}$$

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

$$A_{SS} = \Pr\{\text{System is in state } A\} = \frac{\pi_A E[U]}{\pi_A E[U] + \pi_B \gamma_f + \pi_C \gamma_r} \tag{4}$$

Substituting the values ofπ_A, π_B, π_C:

$$A_{SS} = \frac{E[U]}{P_{AB}\gamma_f + P_{AC}\gamma_r + E[U]} \tag{5}$$

The steady state safety can then be obtained as followed:

$$S = 1 - \Pr\{\text{System is in state } B\} \tag{6}$$

And:

$$S = 1 - \frac{\pi_B \gamma_f}{\pi_A E[U] + \pi_B \gamma_f + \pi_C \gamma_r} \tag{7}$$

Substituting the values ofπ_A, π_B, π_C:

$$S = 1 - \frac{P_{AB}\gamma_f}{E[U] + P_{AB}\gamma_f + P_{AC}\gamma_r} \tag{8}$$

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

$$R_{mI} = \frac{\int_0^{\delta} [\sum_i p_i(t)] dt}{\delta} \tag{9}$$

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

$$R_{mII} = \lim_{T \to \infty} \frac{\int_0^{T} [\sum_i p_i(t)] dt}{T} \tag{10}$$

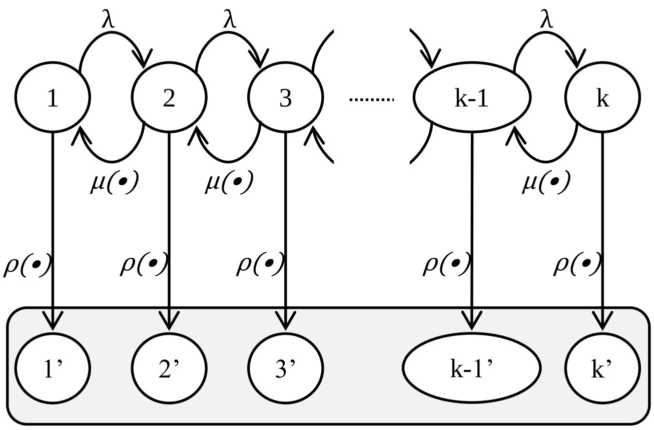

Figure3.Markov process with policy I

B. Policy I

$$\frac{dp_0(t)}{dt} = \mu(\cdot) - [\lambda + \rho(\cdot)]p_0(t) \tag{11}$$

$$\frac{dp_i(t)}{dt} = \mu(\cdot)p_{i+1}(t) + \lambda p_{i-1}(t) - [\mu(\cdot) + \lambda + \rho(\cdot)]p_i(t) \quad 1 \le \text{i} \le \text{k} \tag{12}$$

$$\frac{dp_K(t)}{dt} = \lambda p_{K-1}(t) - [\mu(\cdot) + \rho(\cdot)]p_K(t) \tag{13}$$

$$\frac{dp_{i'}(t)}{dt} = \rho(\cdot)p_i(t) \quad 1 \le \text{i} \le \text{k} \tag{14}$$

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

$$L(t) = \int_{\tau=0}^{t} \sum_i c_i p_i(\tau) d\tau \tag{15}$$

The set of ODEs is first augmented by the following differential equation:

$$\frac{dL(t)}{dt} = \sum_i c_i p_i(t) \tag{16}$$

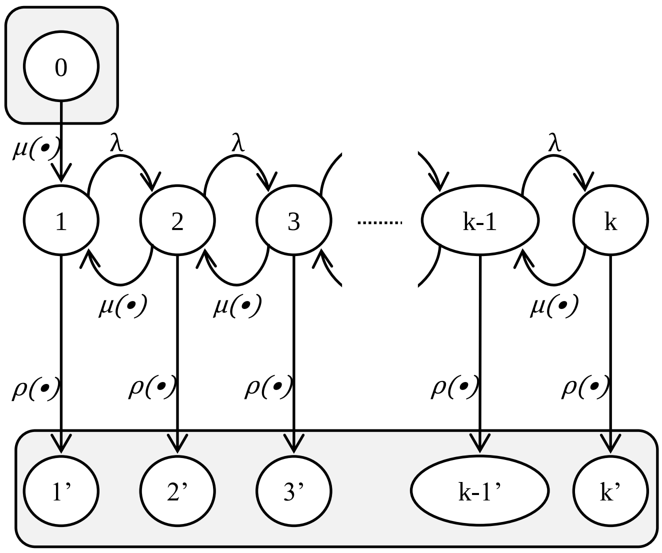

Figure4.Markov process with policy II

The initial conditions: $p_0(0) = 1$, $p_i(0) = 0$ for$1 \leq i \leq L$ and$p_{i'}(0) = 0$ for$0' \leq i' \leq k'$. Then

$$P_{AB} = \sum_{i=0}^{K} p_i(\delta) \tag{17}$$

And

$$P_{AC} = 1 - P_{AB} \tag{18}$$

The expected sojourn time in state A is given by:

$$E[U] = \int_{t=0}^{\delta} \left[\sum_{i=0}^{K} p_i(t) \right] dt \tag{19}$$

C. Policy II

In this case, we need to distinguish between$t \leq \delta$and$t > \delta$, as policy II assumes that preventivemaintenance will be initiated if and only if the buffer isempty after time δ has elapsed. Similar to policy I, onstep transition probability P_{AB} is computed by solvingthe system ofODEs at$t = \infty$ and is given as:

$$P_{AB} = \sum_{i=0}^{K} p_i(\infty) \tag{20}$$

Then

$$P_{AC} = 1 - P_{AB} = p_0(\infty) \tag{21}$$

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

$$E[U] = \int_{t=0}^{\delta} \left[\sum_{i=0}^{K} p_i(t) \right] dt + \int_{t=\delta}^{\infty} \left[\sum_{i=1}^{K} p_i(t) \right] dt \tag{22}$$

$$\Leftrightarrow E[U] = \int_{t=0}^{\delta} p_0(t) dt + \int_{t=0}^{\infty} \left[\sum_{i=1}^{K} p_i(t) \right] dt \tag{23}$$

V. RELIABILITY, AVAILABILITY AND SAFETY EVALUATIONBY SIMULATION EXPREMENTATION

The models are solved for multiple values of δ and optimum value is determined. Using programming solution tool inMatlab, we can estimate Chapman-Kolmogrov equations, thereby simulating the variability of A_{ss}, P_{loss} and the upper bound of response time Tres with system parameters.
Model parameters: $\gamma_f = 0.85(\mathrm{h})$; $\lambda = 6.0(\mathrm{h}^{-1})$;
$k = 50$; $MTTF = 240(\mathrm{h})$Where h = hours.

A. Simulation experiment I

In this experiment,γ_r is varied to ascertain the effect on the measures and on optimal δ. Service rate and failure rate are assumed to be functions of real time, i.e., $\mu(\cdot) = \mu(t)$ and $\rho(\cdot) = \rho(t)$, where $\rho(t) = \beta\alpha t^{\alpha-1}$, which is the hazard function of Weilbull distribution. αisfixed at 1.5 and β is calculated from α and the $MTTF$ as:

$$\beta = \left[\frac{\Gamma\left(1 + \frac{1}{\alpha}\right)}{MTTF} \right]^{\alpha} \tag{24}$$

And$\mu(t)$ is defined as:

$$\mu(t) = \begin{cases} \mu_{max}\left(1 - \frac{1}{MTTF}\right) \text{if } t \leq a \\ \mu_{min} \text{ if } t > a \end{cases} \tag{25}$$

Where

$$\beta = \frac{\mu_{max} - \mu_{min}}{\mu_{max}} MTTF \tag{26}$$

$\mu_{max} = 15h^{-1}$;$\mu_{min} = 5h^{-1}$. Under both polices, itcan be seen that the higher the value of γ_r, the lower isthe availability for any particular value of δ.

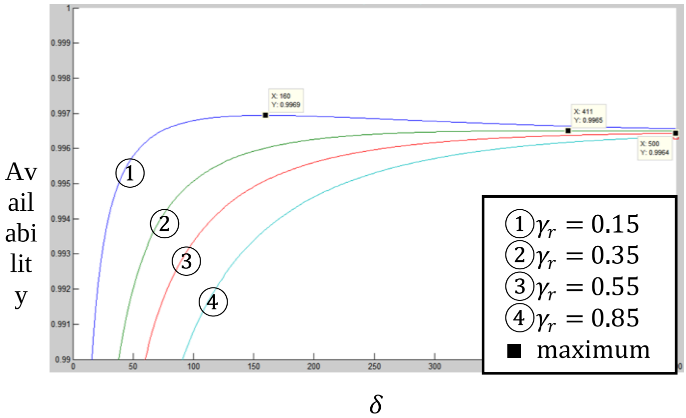

Figure 5.Availability under policy I

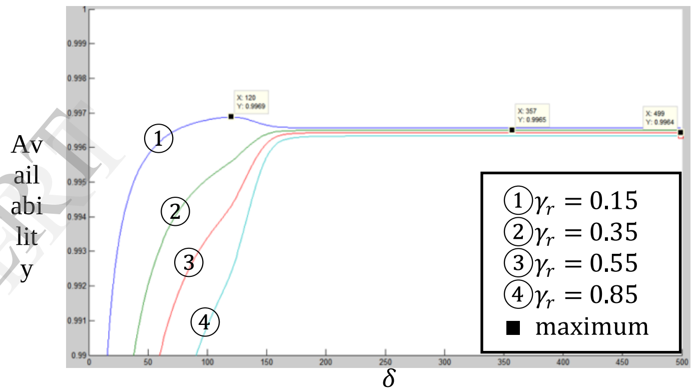

Figure 6.Availability under policy II

Figure 7and Figure 8show that safety will decrease when increasing the value of parameterδ, while safety increases when raisingthe value of parameterγ_r. Since then, we can commentthat under the policy I, the sooner the preventive maintenance will be conducted, the safer the system will be.

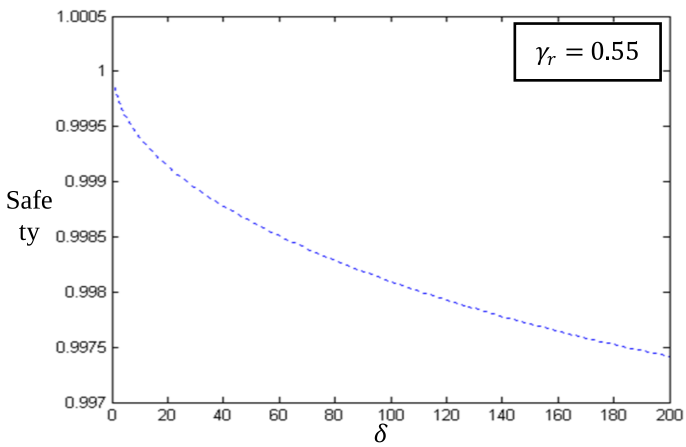

Figure 7.Safety under policy I whereγ_r= 0.35

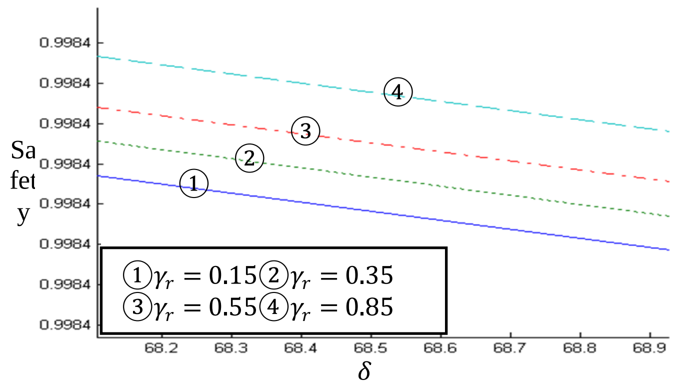

Figure 8.Safety under policy I where γ_r varies

The safety of system under policy II will increase with thedecrease of γ_r (Figure 9). However the dependency is relatively small. In addition, the safety will reduce rapidlyalong with the increase of δ to a threshold (marked onthe drawings) and then will be almost unchanged. Fromtheoretical calculation, we can see the average reliability of the system does not depend onγ_r. Therefore, we fixthe value$\gamma_r = 0.55$and survey the influence of the reliability on the time to wait to perform the preventivemaintenance δ. The average reliability of system underpolicy I rises with the decrease of parameter δ (Figure 10).Clearly, under policies I, the sooner the preventive maintenance is conducted, the higher the level of reliability ofsystem is kept. Under policy II, the reliability is calculated throughout the time domain. It can see that thereliability will increase along with the increase of δ to athreshold and then be kept stable.

B. Simulation experiment II.

In this experiment,γ_r is fixed at 0.15; α is an assigned value of 1.0,1.5 and 2.0, respectively.

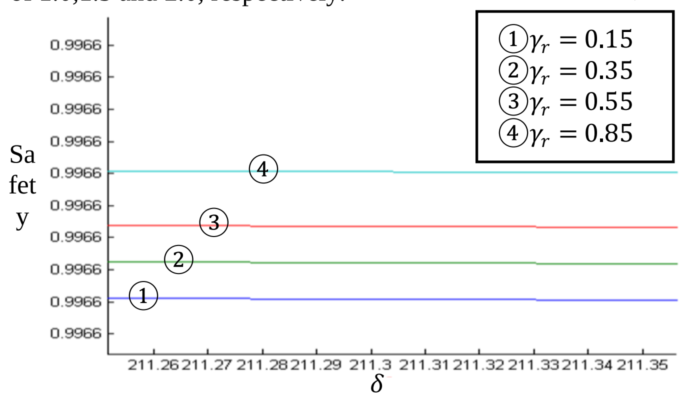

Figure 9.Safety under policy II where γ_r varies

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

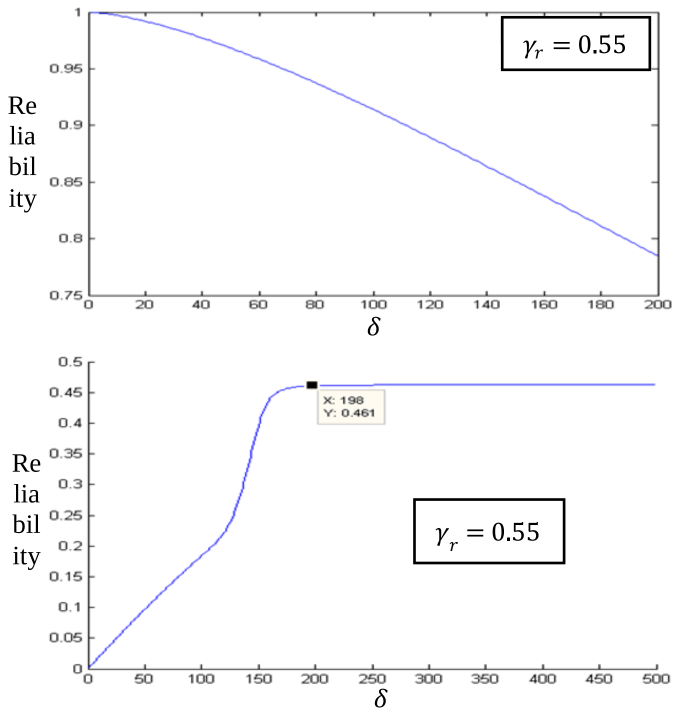

Figure 10.Reliability under policy I and II whereγ_r = 0.35

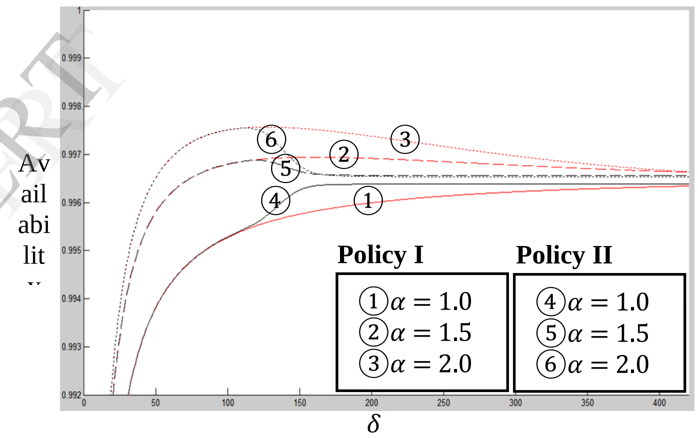

Figure 11.Availability under two policies where αvaries

Figure 12and Figure 13show that the higher the failure density α is, the higherthe value of safety will be in the low value domain ofδ.

On the other hand, when α increases, the ability in whichsystem in the state A will decreases, so the reliability ofsystem 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 δ,and then stops and be kept relatively stable. Meanwhile, the value of parameter δ does not influence the reliabilityof the system any great deal.

C. Experimental results

In this section we will show some experimental result of the proposed method in a real application such as Online Judge system.

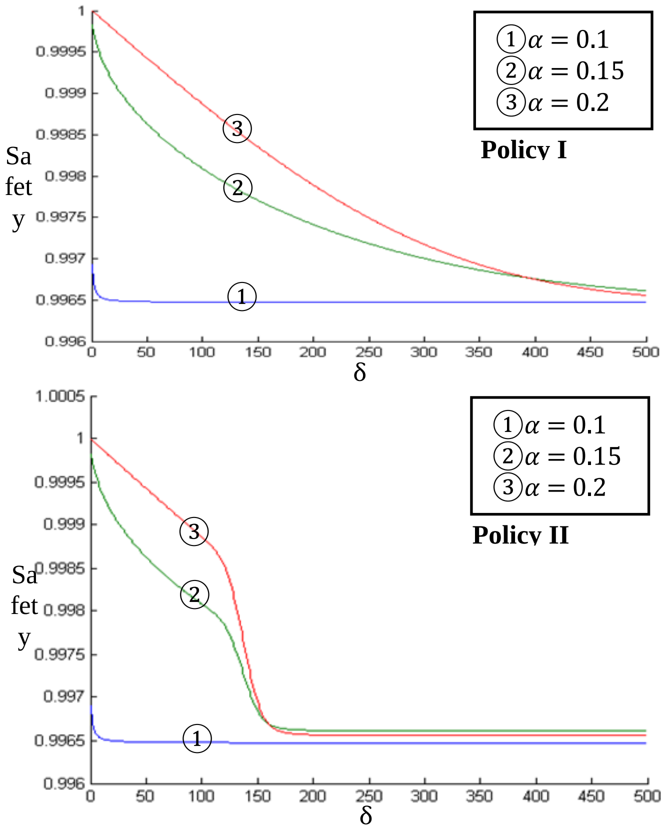

Figure 12.Safety under two policies where α varies

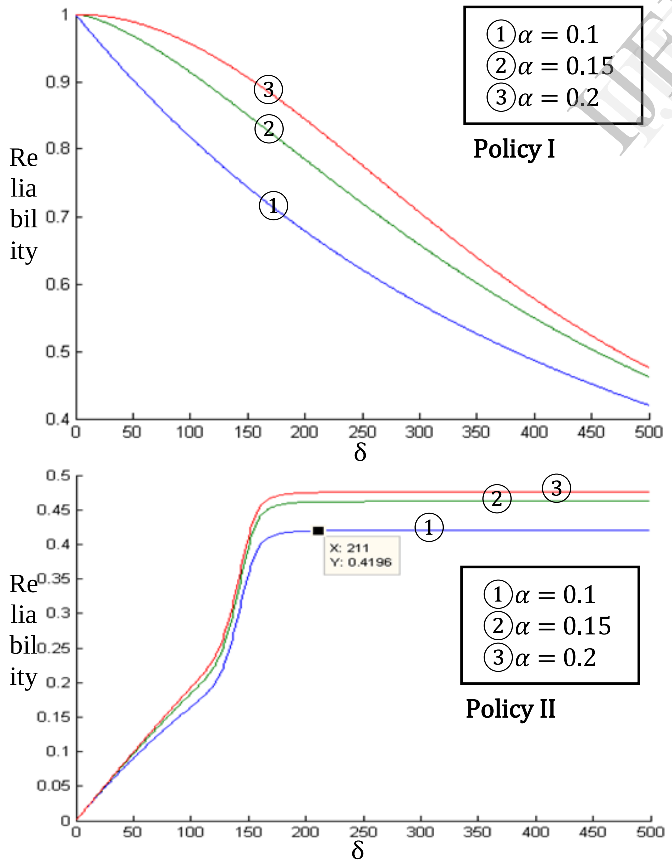

Figure 13.Reliability under two policies where αvaries

1) Online Judge

An online judge is an automated judge which checks a submitted solution for an existed problem and generates the output. It checks if the generated output was correct with respect to the output set that is saved as a full proof judge output set for that particular problem thus generate the result for the user such as Accepted, Wrong Answer, Runtime Error, etc.

An online judge is in general a server, which contains descriptions of problems from different contests, as well as data sets to judge whether a particular solution solves any of theseproblems. A user from anywhere in the world can register himself (or herself) with an online judge for free and solve as many problems as he likes. He can send as many solutions as he want till receiving satisfactory information, not only about the verdict, but also about the time that the code takes to run after improving the program and/or the algorithm used to solve the selected challenge. One of the main distinctive trait of the online judges is that they allow the users this self-competitive behavior to learn informatics, not only algorithms but also programming.

There are several existing popular online judges all over the World Wide Web. Here mentioned some of them:UVA Online Judge, Sphere Online Judge, and BKOJ Online Judge.

2) BKOJ Online Judge

BKOJ is an online judge of Ha Noi uninversity of Science and Technology. It was built with the primary purpose of being used as a training tool for ACM /ICPC teams of the university.

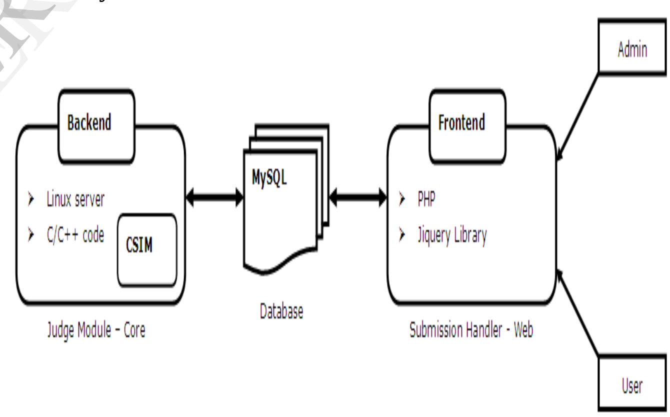

Figure 14.Block diagram of BKOJ

BKOJ system consists of two major parts: Web (as Frontend) and Core (as Backend). The Web part plays as distributed information management system, managing information such as user registration, problem submission, solution submission, problem modification,etc.

The Core part as kernel of BKOJ system provides a method to judge all solutions, which were submitted by any users. In this paper, we only focus on the functions of the Core part.

BKOJ system has been installed in BKCloud platform as a software service running from 2012. Figure 15 shows its deployment in the platform.

The Core part as kernel of the system provides a method to judge all solutions, which were submitted by any users. In this disscussion, we only focus on the functions of the Core part.

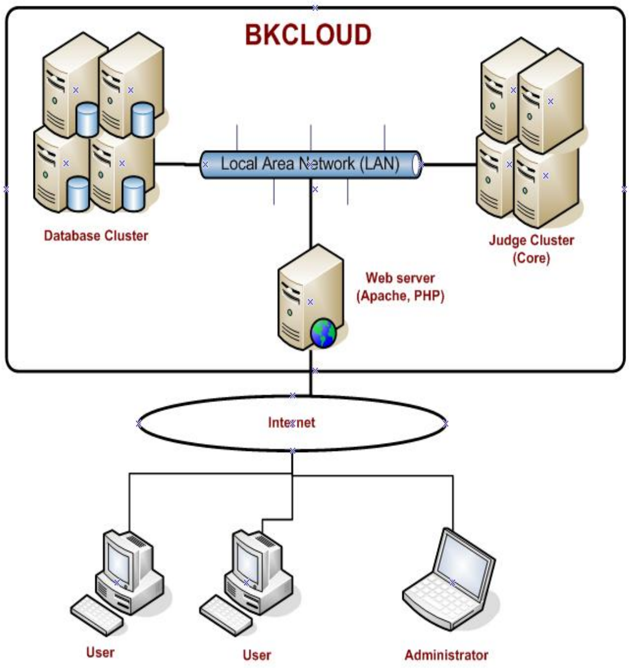

Figure 15. BKOJ deployment in the BKCloud system

3) *BKOJ - Core working process*

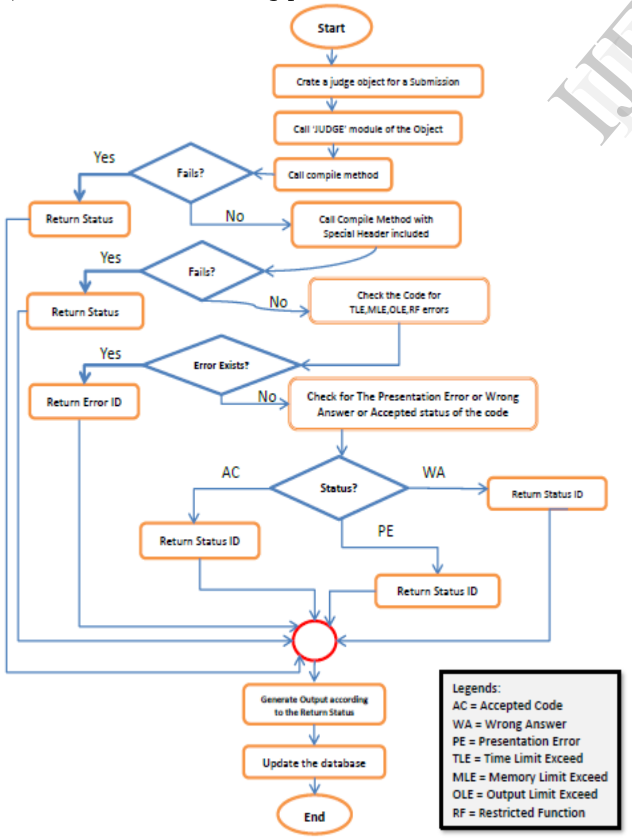

Figure 166.The judgment process of BKOJ - Core

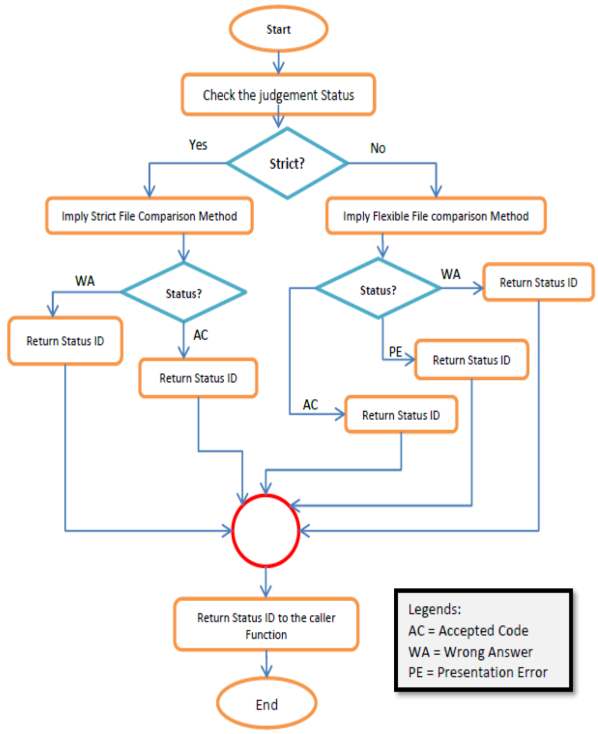

Figure 177.The probabilites of checking outputs

4) *The experimental results in detail*

In this experiment, we only consider applying the Policy I to BKOJ. We createdavirtualcontestwiththe simultaneous participation of 500 virtual contestants during 8hours. The contestusingthe data,collectedfrommany private contest of our university – HUST - from 2012 to 2013.

Somebasicinformation about the contest is showninthe following table:

Server Software	Apache/2.2.14		
Document Path	/JudgeOnline/status.php		
Concurrency Level	500		
Time taken for tests	32,239 seconds		
Complete requests	1000		
Failed requests	207 (Connect: 0, Receive: 0, Length: 207, Exceptions: 0)		
Total transferred	7262294 bytes		
HTML transferred	6956196 bytes		
Requests per second	31.02[#/sec](mean)		
Time per request	16119.251[ms](mean)		
Time per request	32.239[ms] (mean, across all concurrent requests)		
Connection Times(ms)			
	min	median	max
Connect	1	229	3000
Processing	296	2009	32200
Waiting	0	1002	15725

Basedon theinformation mentionedinthe above tablewedrew thecontinuos theoreticalcurvesof the Availability, the Safety and the Reliability of BKOJasin Figures 18,19 and

20respectively. In contrast, the black discretepoints representthe actual valueofthe relating properties ofBKOJ. Model parameters: $\gamma_r = 0.5$ (h);$\gamma_f = 0.5$(h); $\lambda = 6.0(\text{h}^{-1})$; $k = 30$; $MTTF = 240$(h); $\mu(\cdot) = \mu(t)$ and $\rho(\cdot) = \rho(t)$ are the same as (24)(25)(26);α is fixed at 1.0; δ is a assigned values of 96(h), 144(h), 192(h), 384(h) respectively (where h = hours)

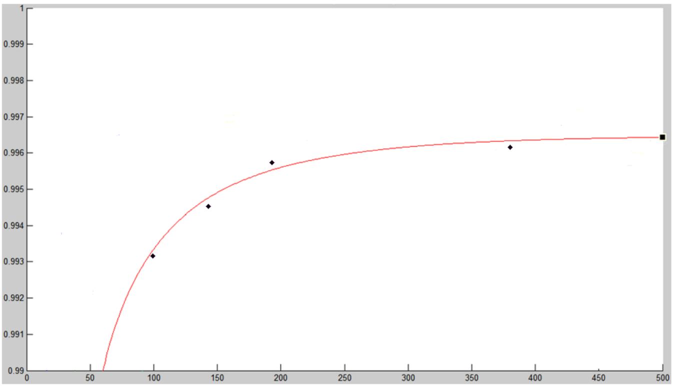

Figure 188. Availability of BKOJ under policy I

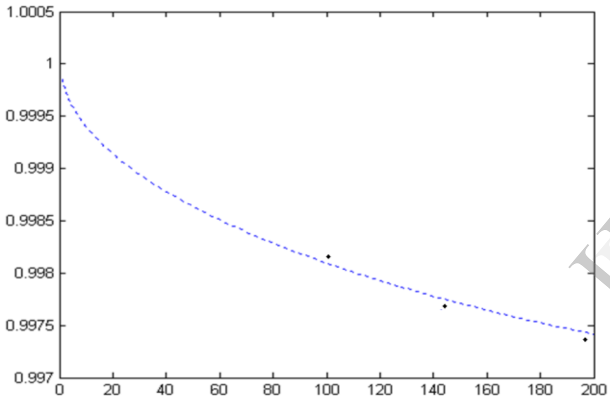

Figure 199. Safety of BKOJ under policy I

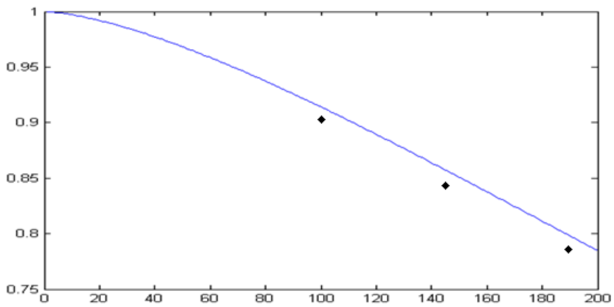

Figure 20. Reliability of BKOJ under policy II

The experimental resultsshow thatthe theoreticalcurvesfitquitewell withlimitedpracticalvalue, which confirmedthe practical valueofthe method for evaluating the quality properties in a rejuvenation system using Markov model.

VI. CONCLUSIONS

Applying the theory of mathematical Markov model, the theory of rejuvenation, we have built a model to evaluate the software attributes of rejuvenation systems. The proposed approach used two Markov chain models with corresponding policy I and II. We showed expanding math calculation of model of this method by using the Matlab. The experiments with BKOJ SaaS on BkCloud system are confirmed its worth.

In the future, based on the relationship between these software attributes and fault tolerance techniques on cloud environment, the research will be further developed. From the evaluation of the software attributes of fault-tolerant software in cloud environments, 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.

ACKNOWLEDGMENT

This research is sponsored by the research Grant KC.01.01/10-15 by Ministry of Science and Technology Vietnam

VII. REFERENCES

[1] M. Larsson, "Predicting quality attributes in a component-based software systems". Malardalen University Press, 2004, ISBN: 91-88834-33-6; ISSN: 1651-4238.

[2] X. CHENGJIE, “Availability and reliability analysis of computer software systems considering maintenance and security issues”, PhD Thesis 2011. http://scholarbank.nus.edu.sg/handle/10635/25829

[3] R. Roshandel, “Calculating architectural reliability via modeling and analysis,” in Proceedings of the 26thInternational Conference on Software Engineering, pp. 69–71, IEEE Computer Society, 2004.

[4] H. Pham, System software reliability. Springer, 2006.

[5] M. Grottke, L. Li, K. Vaidyanathan, and K. S. Trivedi,“Analysis of software aging in a web server,” Reliability, IEEE Transactions on, vol. 55, no. 3, pp. 411–420, 2006.

[6] T. Dohi, K. Goseva-Popstojanova, K. Vaidyanathan, K. S. Trivedi, and S. Osaki, “Software rejuvenation: modeling and applications,” in Handbook of ReliabilityEngineering, pp. 245–263, Springer, 2003.

[7] Michael R. Lyu. Software Reliability Engineering: A Roadmap. International Conference on Software Engineering 2007 Future of Software Engineering, pp. 153-170, ISBN:0-7695-2829-5

[8] Pham Thanh Trung, Huynh Quyet Thang. Building the Reliability Prediction Model of Component-Based Software Architectures. International Journal of Information Technology, Volume 5, No. 1, 2009, pp. 17-25.