ISSN 1673-9418 CODEN JKYTA8
Journal of Frontiers of Computer Science and Technology
1673-9418/2008/02(04)-0431-08
DOI:10.3778/j.issn.1673-9418.2008.04.009

E-mail:fcst@public2.bta.net.cn
http://www.ceaj.org
Tel:+86-10-51616056

Research on Internetware Reliability Measurement Model Based on Service Update*

DU Yugen+, LI Yonggang
Software Engineering Institute, East China Normal University, Shanghai 200062, China
+ Corresponding author: E-mail: ygdu@sei.ecnu.edu.cn

基于服务更新的网构软件可靠性模型研究*

杜育根+,李永钢
华东师范大学 软件学院,上海 200062

摘 要:网构软件是在开放、动态和多变的Internet环境下软件系统基本形态的一种抽象。这种新的软件系统,它的构建依赖于开放环境中各异构、自治的软件服务实体之间的有效协同。其可靠性不单取决于拥有自主性的软件服务实体本身,还取决于外部环境的动态变化,主要表现为开放环境下服务实体元素的更新,所以传统的软件可靠性的度量模型已不能适应这种新的软件形态。能否在网构软件形态下建立一个好的可靠性度量模型成为其中一个较为核心的问题。文章以服务更新过程中失效数(failure counts)为基础,将服务更新强度引入Musa-Okumoto(M-O)模型中,作为M-O模型在新的软件形态下的一个推广。最后讨论了网构软件退化的情形和退化条件,为开放环境下网构软件可靠性研究提供一种思路。

关键词:网构软件;服务更新;可靠性模型;开放环境
文献标识码:A **中图分类号:**TP311

DU Yugen, LI Yonggang. Research on internetware reliability measurement model based on service update. Journal of Frontiers of Computer Science and Technology, 2008,2(4):431-438.

Abstract: Internetware is an abstract of software system basic paradigm under an open, dynamic and ever-changing Internet environment. This new software system is architected on an effective collaboration of heterogeneous and autonomous software service entities under an open environment. Its reliability depends not only

* the National Natural Science Foundation of China under Grant No.90718013 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2007AA01Z189 (国家高技术研究发展计划(863)).
Received 2008-03, Accepted 2008-06.

on autonomous software entity itself, but also on dynamic changes of outside, such as changes of service entity elements under an open environment. The current reliability measurement model for traditional software is no longer suitable for this new software paradigm. It becomes a more urgent issue whether a good reliability measurement model can be built under internetware software paradigm. A concept of failure counts during service update process is brought forward. Service update intensity is applied into Musa-Okumoto model, which is one of the most widely used software reliability models, therefore this model is expanded under new software paradigm. Finally internetware deterioration and its condition are also discussed.

Key words: internetware; service update; reliability model; open environment

1 Introduction

Internetware is a new paradigm of software fit for a collaborated, open and dynamic environment. Its operation depends on all kinds of heterogeneous and autonomy service entities' collaboration[1]. It is a natural extension of traditional software structure, but has its own unique characters such as open structure, dynamic cooperation, online evolution, environment apperception and self-adaptation[2-3]. Internetware will span from information web to software web.

Internetware has a service-oriented architecture and its component is a service entity (Fig.1). Here, service entity, in forms of distributed, autonomous and heterogeneous components, is independent, spontaneous and self-adaptive. Entity interaction is inter-connection, inter-communication, collaboration and alliance by manners of several static connection and dynamic cooperation.

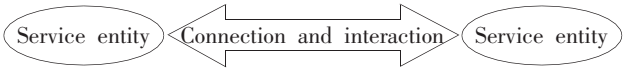

Fig.1 The basic paradigm of service-oriented software

图 1 面向服务软件的基本形态

However, internetware development process is under an open, dynamic and ever-changing Internet environment with rich basic software resources (see Fig.2). Its basic system is assembled by resources of basic software entities which is self-independent and can not be managed and collaborated directly by developer. What's more, in the total internetware lifecycle, the structure of internetware is still

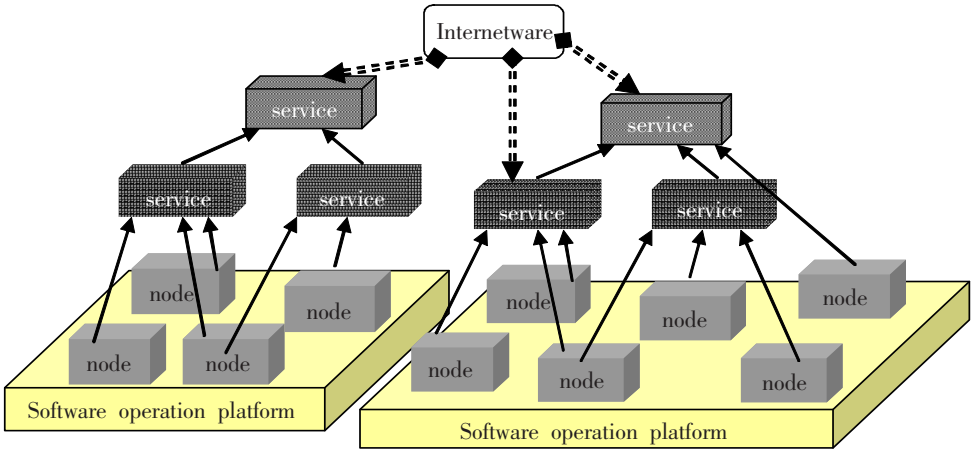

Fig.2 Internetware paradigm

图 2 网构软件形态

under dynamic evolution after internetware has been submitted for operation. For example, by system requirements changes, participant service entities may dynamically leave or join the system, or by surroundings changes, service entities functions will be re-configured and updated[4].

Therefore service update includes two levels of meanings: one level is service update by the composition changes of its service entities, the other level is the update by single service entity itself.

Traditional reliability models only discuss these failures caused by software fault while ignoring the problems caused by internetware dynamic evolution. We'd like to introduce service updating process into the model to adapt to such new software paradigms. In the 3rd section, we will discuss in details about some changes after we impose service updating intensity on Musa-Okumoto model. Considering side effects that each update will bring, we also define side effect intensity in the 5th section to depict deterioration of software.

2 Internetware Reliability Static Model

Traditional software reliability models can be categorized as time-based model, which is based on software failure time and failure intervals, such as Jelinski-Moranda models; failure-based model, which is based on the sum of failures and failure rate, such as Goel-Okumoto and Musa-Okumoto model[5].

According to hypothesis of comprehensive model, testing environment and statistics methods presented in Reference [6], it can be also grouped as random process model, such as Markov process model and Poisson process model and non-random process model.

Among process models, if supposed failure rate is a constant in software unchanged intervals and drops along with failure counts, those models belong to Markov process type; If cumulative failure counts during debugging process is taken as time function $N(t)$, under some conditions it can be approximate to a non Poisson process, such models belong to non-Poisson Process type. Under such a way of categorizing, Jelinski-Moranda model belongs to Markov process, Goel-Okumoto model to non-Poisson process. While Musa's execution time model shall fall into Markov process type, that is to say, future distribution condition of a process is independent to the past. And Musa-Okumoto's logarithmic-Poisson-execution-time-model shall be in non-homogeneous Poisson process type. Failure refers to many faults or bugs that reside in the code. When an input to the software activates a module where a fault is present, a failure can occur.

Cumulative failure function $N(t)$: cumulative number of failures experienced by time t. It is a random process. The expected value of $N(t)$ is called the "Mean value function":

$$\mu(t)=E[N(t)]$$

Failure intensity function $\lambda(t)$ represents the rate of change (derivative) of the mean value function $\mu(t)$:

$$\lambda(t)=\frac{\mathrm{d}}{\mathrm{d}t}\mu(t)$$

Failure rate, also called hazard rate, is the rate at which failures occur in the interval $[t,t+\Delta t]$:

$$r(t)=\lim_{\Delta t\to 0}\frac{p(t\leqslant T\leqslant t+\Delta t|T>t)}{\Delta t}$$

where T represents time when failure occurs.

$$r(t)=\frac{f(t)}{R(t)}$$

Unreliability $F(t)$ is the probability that a failure will occur in the interval $[0,t]$. And $f(t)$ is the density function of $F(t)$.

Reliability $R(t)$ is the probability of failure-free operation for a specified period under stated conditions. $R(t)$ is probability for no failure in the interval $[0,t]$.

$R(t)=1-F(t)$, $R(t)=\int_t^{\infty} f(x)\mathrm{d}x$

2.1 Software Service Entity Reliability Analysis

Suppose bugs of component software are independent to each other, based on cumulative failure function, we can use Goel-Okumoto model to give a system reliability exponent growth model (NHPP). Here, a is the expectation of number of bugs found; b is bugs found rate at time t, then failure intensity function and mean value function of each service software entity can be expressed as below:

Failure intensity function $\lambda(t)=abe^{-bt}$

Mean value function $\mu(t)=a(1-e^{-bt})$

The reliability of failure-free operation in $(t_i,t_i+\Delta t)$ (t_i is the time when the ith failure has occurred):

$R(\Delta t|t_i)=\exp(-ae^{-bt_i}(1-e^{-b\Delta t}))$

2.2 Reliability Analysis on Connection Structure of Service Entities

The reliability of service assembled system depends not only on different reliability of each service component, but also on relationship of the connection structures of each component. The relationship can be grouped as below[7-9]:

(1)Serial system $C_1\rightarrow C_2\rightarrow\cdots\rightarrow C_n$, Failure of any component results in the failure of the whole system.

Failure intensity: $\lambda_{sys}=\sum_{i=1}^{n}\lambda_i$

System reliability: $R_{sys}(t)=\prod_{i=1}^{n}R_i(t)$

where λ_i is the Failure intensity and $R_i(t)$ is the reliability of the ith component C_i.

(2)Parallel system $C_1\parallel C_2\parallel\cdots\parallel C_n$, Failure of all components results in the failure of the whole system.

System reliability: $R_{sys}(t)=1-\prod_{i=1}^{n}(1-R_i(t))$

(3)Circulate service system, one or more component circulated for N times use $[C_i]^n$.

System reliability: $R_{sys}(t)=R_i(t)^n$

(4)Parallel-Series system

$(C_{11}\rightarrow C_{12}\rightarrow\cdots\rightarrow C_{1n})\parallel(C_{21}\rightarrow C_{22}\rightarrow\cdots\rightarrow C_{2n})\parallel\cdots\parallel(C_{m1}\rightarrow C_{m2}\rightarrow\cdots\rightarrow C_{mn})$

then single path reliability of the system is $R_i=\prod_{j=1}^{m}R_{ij}$;

System reliability is $R_{sys}=1-\prod_{i=1}^{m}(1-\prod_{j=1}^{n}R_{ij})$.

(5)Series-Parallel system

$(C_{11}\parallel C_{21}\parallel\cdots\parallel C_{m1})\rightarrow(C_{12}\parallel C_{22}\parallel\cdots\parallel C_{m2})\rightarrow\cdots\rightarrow(C_{1n}\parallel C_{2n}\parallel\cdots\parallel C_{mn})$

then single path reliability is $R_j=1-\prod_{i=1}^{m}(1-R_{ij})$ and

System reliability is $R_{sys}=\prod_{j=1}^{n}\left[1-\prod_{i=1}^{m}(1-R_{ij})\right]$.

(6)M/N system

Suppose there are total N same components with reliability $R(t)$ in a parallel system, at least M components need to work correctly for the system to function properly ($m\leqslant n$), then system reliability is

$$R_{sys}(t)=1-\sum_{i=0}^{m-1}\binom{n}{i}R(t)^i(1-R(t))^{n-i}$$

Obviously when $M=1$, it deteriorates to parallel system and when $M=N$, series system.

2.3 Reliability Analysis on Connection of Service Entity

The service interface of internetware components can be realized on different platforms and can be accessed and interacted by different communication protocols such as Java RPC, SOAP, etc, on the other hand, a uniform service interface described in WSDL is provided to the outside. Therefore, the realization and access of components service is separated. Developer only needs to know the components service interface, ignoring its realization and accessing methods. Developer has much easier access to these component resources. The protocol of mid-

dleware components of internetware determines the reliability of its connection.

SOAP provides an interchange mechanics based on XML for different components under internet environment. Different from other protocols, SOAP arranges transferred message to text form, which is designed for combination with all kinds of protocols.

The reliability of SOAP depends on its binding with transport protocol. If UDP related transport protocol is adopted, groupings may be lost due to some factors such as the band width, time extension, size of the grouping and so on[10]. If bound with common HTTP or SMTP protocols, its reliability is guaranteed by TCP.

3 Reliability Model Based on Service Update

Service oriented, internetware possesses the paradigm shown as Fig.2. Smooth running of software depends on reliability of service entity. Failure count is a random process, its cumulative failure function is determined by two other random processes.

(1)Cumulative failure function $N(t)$, where the failures are caused by service component entity fault;

(2)Cumulative failure function $M(t)$, where the failures are caused by service updates under a dynamic evolution environment or system requirements changes.

While service oriented failure count is also a random process, it is two dimension distribution of $N(t)$ and $M(t)$. Model hypotheses with reference to [11-13] are given as below:

(1)When time t is 0, failure count is 0 and service update count is 0.

(2)Since service failure caused by service entity's fault or bugs will reduce along with progress of testing, thus failure intensity $\lambda(t)$ decays exponentially:

$$\lambda(t)=\lambda_0 e^{-\theta\mu_1(t)}$$

(λ_0 is initial failure intensity, θ is failure intensity decay parameter and $\mu_1(t)=E[N(t)]$)

(3)Suppose failure increase caused by service changes is stable which is a homogeneous Poisson process and service update intensity $\delta(t)$ is a constant $\tilde{\delta}$, and then there exist

$$\delta(t)=\frac{\mathrm{d}\mu_2(t)}{\mathrm{d}t}=\tilde{\delta},\ \mu_2(t)=E[M(t)]$$

(4)Suppose no new bugs occur after service entity's malfunction has been repaired; suppose if service updates happen, component service entity's original failure rate and failure decay rate after service updates are equal to their values before service updates. (Of course, this supposition is ideal. We will discuss later in the 5th section, the situation where every service update will incur new bugs).

(5)Within a small interval $(t,t+\Delta t)$, probability of one service failure is $\lambda(t)\Delta t+o(\Delta t)$, probability of more than one failure is $o(\Delta t)$, $\lim\limits_{\Delta t\to 0}\frac{o(\Delta t)}{\Delta t}=0$.

From supposition (2), differential equation can be deducted as

$$\frac{\mathrm{d}[e^{\theta\mu_1(t)}]}{\mathrm{d}t}=\lambda_0\theta$$

Its integral is $e^{\theta\mu_1(t)}=\lambda_0\theta t+C$, C is integral constant.

From supposition(1), $\mu_1(0)=0$, so $C=1$. The mean function is

$$\mu_1(t)=\frac{\ln(\lambda_0\theta t+1)}{\theta} \tag{1}$$

From supposition (3), differential equation $\frac{\mathrm{d}\mu_2(t)}{\mathrm{d}t}=\tilde{\delta}$, integral is $\mu_2(t)=\tilde{\delta}t+C$, C is integral constant, and from supposition (1), $C=0$ and

$$\mu_2(t)=\tilde{\delta}t \tag{2}$$

Since service update random process and component software entity failure process are independent to each other, from supposition (4), the mean value function of service failure can be calculated

$$w(t)=\mu_1(t)+\mu_2(t)=\frac{1}{\theta}\ln(\lambda_0\theta t+1)+\tilde{\delta}t \quad (3)$$

From supposition (1) and (5), service failure is a Poisson process. Therefore, probability of $w(t)=m$ is $Pr\{w(t)=m\}=\frac{[w(t)]^m}{m!}e^{-w(t)}$, service reliability distribution is

$$R(\Delta t|t_{i-1})=e^{-[w(\Delta t+t_{i-1})-w(t_{i-1})]} \quad (4)$$

By formula (3) and (4)

$$R(\Delta t|t_{i-1})=\exp\{-[\frac{\ln\lambda_0\theta(\Delta t+t_{i-1})+1}{\theta}+\tilde{\delta}(\Delta t+t_{i-1})-\frac{\ln\lambda_0\theta(t_{i-1})+1}{\theta}-\tilde{\delta}t_{i-1}]\}=[\frac{\lambda_0\theta t_{i-1}+1}{\lambda_0\theta(\Delta t+t_{i-1})+1}]^{1/\theta}e^{-\tilde{\delta}\Delta t} \quad (5)$$

Let $\xi=\frac{1}{\theta}\ln\frac{\lambda_0\theta t_{i-1}+1}{\lambda_0\theta(\Delta t+t_{i-1})+1}$, the formula (5) is simplified as

$$R(\Delta t|t_{i-1})=\exp\{\xi-\tilde{\delta}\Delta t\} \quad (6)$$

In actual application, parallel and series structure systems are most commonly used, such as an example in the coming section. Suppose the base protocol of component connection is reliable and the reliability function of service k is

$$R_k(\Delta t|t_{i-1})=\exp\{\xi_k-\tilde{\delta}_k\Delta t\}$$

then when two services are series structured, their joint reliability function is calculated

$$R_{serial}(\Delta t|t_{i-1})=\exp\{\xi_1+\xi_2-(\tilde{\delta}_1+\tilde{\delta}_2)\Delta t\} \quad (7)$$

When two services are parallel structured, their joint reliability function is calculated

$$R_{parallel}(\Delta t|t_{i-1})=R_1(\Delta t|t_{i-1})+R_2(\Delta t|t_{i-1})-R_{serial}(\Delta t|t_{i-1})=\exp\{\xi_1-\tilde{\delta}_1\Delta t\}+\exp\{\xi_2-\tilde{\delta}_2\Delta t\}-\exp\{\xi_1+\xi_2-(\tilde{\delta}_1+\tilde{\delta}_2)\Delta t\} \quad (8)$$

4 Example

In this section, the above model based on internet service changes is applied to analyze its service reliability, then an overall reliability measurement is given by analysis on connections between internetware entities.

An example quoted from reference [14] with some modification is shown as Fig.3. A composite service TravelPlan, presented by a travel agency combined with services of its partners such as airlines, train ticket agencies, banks, express companies and so on, can tailor-make a travel plan for its customers according to their different needs.

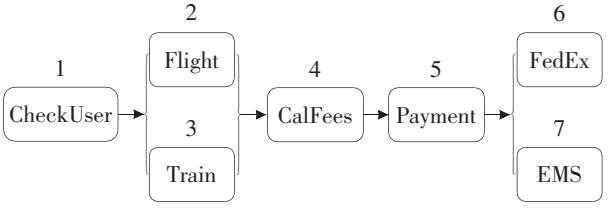

Fig.3 Travel plan

图3 旅行计划

Each service function is shown as below:

CheckUser: check if flow requestor is registered user of the travel agency and book hotel at destination;

Flight: book the flight tickets to destination;

Train: book train ticket to destination;

Calfees: calculate total expense;

Payment: pay the expense;

FedEx, EMS: finally book FedEx, UPS or EMS to deliver notes.

Among them, services of booking air tickets or train tickets are parallel. The flow will continue if only one booking service is successful. Another parallel structure is to book service from two express companies. Actual delivery only requires one, without any orientation.

Suppose the base protocol of the connection of component is reliable, the formula of the above section of (7) and (8) can be directly applied.

The reliability of the whole system is distributed as below:

$$R_{system}(t)=\exp\{\xi_1+\xi_4+\xi_5-(\tilde{\delta}_1+\tilde{\delta}_4+\tilde{\delta}_5)\Delta t\}\cdot$$
$$(\exp\{\xi_3-\tilde{\delta}_3\Delta t\}+\exp\{\xi_2-\tilde{\delta}_2\Delta t\}-\exp\{\xi_3+\xi_2-(\tilde{\delta}_3+\tilde{\delta}_2)\Delta t\})\cdot$$
$$(\exp\{\xi_6-\tilde{\delta}_6\Delta t\}+\exp\{\xi_7-\tilde{\delta}_7\Delta t\}-\exp\{\xi_6+\xi_7-(\tilde{\delta}_6+\tilde{\delta}_7)\Delta t\})$$

5 Internetware Deterioration Discussion on the Model

In total life cycle of internetware, its idealized curve shows the software will not wear out, but deterioration does exist. New faults may occur in every software service changes, which will dramatically increase failure intensity (failure rate) as Fig.4 shows below. Before the curve can return to original steady-state failure rate, another change is requested, causing the curve to spike again. Slowly, the minimum failure rate level rises gradually. In other words, software is deteriorating due to side effects caused by continuous changes[15].

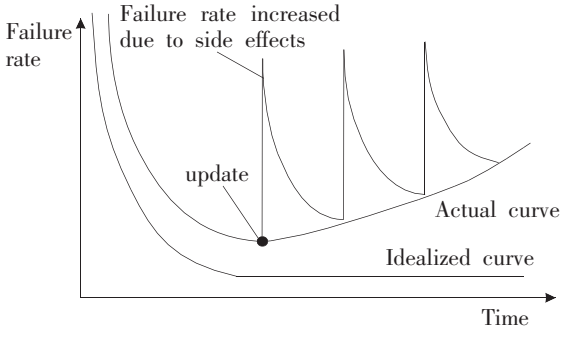

Fig.4 Software failure rate curve

图4 软件失效率曲线

So the above model hypothesis shall be modified as such: if at time t_i service update happens, after i times changes, the failure rate is changed from λ_i to λ_{i+1}. Suppose failure decay rate θ is the same after each change, and service update side effect index is $\frac{\ln\lambda_{i+1}-\ln\lambda_i}{\theta}$, When

$$\lambda_{i+1}e^{-\theta\mu(t_i+\frac{1}{\tilde{\delta}})}>\lambda_i e^{-\theta\mu(t_i)}\Rightarrow$$
$$\frac{1}{\theta}(\ln\lambda_{i+1}-\ln\lambda_i)>\mu(t_i+\frac{1}{\tilde{\delta}})-\mu(t_i)$$

That is to say, when side effect index is bigger than the failure counts in the interval time $1/\tilde{\delta}$, software deterioration happens.

Therefore, we can test internetware under above conditions and judge if software deteriorates. We can enhance software reliability by improving quality of service update and reducing service update intensity.

6 Conclusion

This article, on the basis of failure counts in service changing process, introduced service changing intensity into Musa-Okumoto model as an expansion under internetware paradigms. From formula (5) it can be found that service reliability is under minus exponential relationship with service change intensity. If our requirements change frequently and internet service is not stable, then service reliability will be reduced due to increased update intensity. The model proposes a reliable measurement for internetware under an open, dynamic and evolved environment. Finally the article also discussed some problems that may occur when the model is applied to practical use. The model is still under preliminary stage and its precision will be tested in practice. It can be improved by some new approaches such as internetware response and its lifecycle.

References:

[1] Mao Xiaoguang, Deng Yongjin. A general model for component-based software reliability[J]. Journal of Software, 2004,15(1):27-32.

[2] Yang Fuqing. Thinking on the development of software engineering technology[J]. Journal of Software, 2005,16(1):1-7.

[3] Yang Fuqing, Mei Hong, Lv Jian, et al. Some discussion on the development of software technology[J]. ACTA Electronica, 2002,30(12A):1901-1907.

[4] Poladian V, Sousa J P, Garlan D, et al. Dynamic confi-

guration of resource-aware services[C]//Proc of the 26th Int'l Conf on Software Engineering (ICSE), Edinburgh, 2004: 604-613.

[5] Singpurwalla N D, Wiilson S P. Statistical methods in software engineering: reliability and rick[M]. New York: Springer-Verlag Inc, 1999.

[6] Xu Renzuo, Xie Min, Zheng Renjie. Software reliability models and applications[M]. Beijing: Tsinghua University Press, 1994.

[7] Benatallah H R. A Petri Net-based model for web services composition[C]//the 14th Australasian Database Conference, Adelaide, Australia, 2003.

[8] Hamlet D, Mason D, Woit D. Theory of software reliability based on components[C]//the 3th International Workshop on Component-based Software Engineering. Toronto: IEEE Computer Society, 2001:361-370.

[9] Yang W L, Wu Y, Chen M H. An architecture-based software reliability model[C]//Proceedings of Pacific Rim International Symposium on Dependable Computing. Hong Kong, China: IEEE Press, 1999.

[10] Wang Ping, Sun Changsong, Li Lijie. Primary research on internetware reliability technology[C]//Proceedings of the First International Multi-Symposiums on Computer and Computational Sciences (IMSCCS'06), College of Computer Science and Technology, Harbin Engineering University, Harbin, 2006.

[11] Lu Daquan. Stochastic processes with applications[M]. Beijing: Tsinghua University Press, 1986.

[12] Musa J D, Iananino A, Okumoto K. Software reliability: measurement, prediction, application[M]. New York: McGraw-Hill, 1987.

[13] Musa J D. Software reliability measurement[J]. Journal of Systems and Software, 1980,1:223-224.

[14] Huang Tao, Ding Yongjin, Wei Jun. Research on the relaxed transaction models for applied semantics-based of internetware[J]. Science in China Ser E: Information Sciences, 2006,36(10):1170-1188.

[15] Pressman R S. Software engineering: a practitioner's approach[M]. 6th ed. New York: McGrill-Hill Company Inc, 2005:5-6.

附中文参考文献:

[1] 毛晓光,邓勇进.基于构件软件的可靠性通用模型[J].软件学报,2004,15(1):27-32.

[2] 杨芙清.软件工程技术发展思索[J].软件学报,2005,16(1):1-7.

[3] 杨芙清,梅宏,吕建,等.浅论软件技术发展[J].电子学报,2002,30(12A):1901-1907.

[6] 徐仁佐,谢文旻,郑人杰.软件可靠性模型及应用[M].北京:清华大学出版社,1994.

[11] 陆大铨.随机过程及其应用[M].北京:清华大学出版社,1986.

[14] 黄涛,丁晓宁,魏峻.基于应用语义的网构软件松驰事物模型研究[J].中国科学 E 辑:信息科学,2006,36(10):1170-1188.

DU Yugen was born in 1966. He received the M.S. degree in Operational Research and Control Theory from Shanghai University of Science and Technology in 1992. He is a professor at East China Normal University. His research interests include software reliability measurement, SOA, software modeling and engineering.

杜育根(1966-),1992 年于上海科技大学(现为上海大学)获运筹学与控制论专业硕士学位,华东师范大学软件学院副教授,研究兴趣包括软件建模和工程、软件可靠度量、面向服务体系结构。

LI Yonggang was born in 1985. He is a graduate student at East China Normal University. His research interests include software reliability measurement and Web service, etc.

李永钢(1985-),华东师范大学软件学院硕士研究生,研究兴趣包括软件可靠度量、Web 服务等。