Analysis on aerospace software health metrics based on multi-factor reliability growth model

Weiqi Xie[1]*, Yuanwen Cai[2], Long Cheng[2]

[1]*Department of Graduate Management, Equipment Academy, Beijing, 101416, Beijing, China*

[2]*Department of Space Equipment, Equipment Academy, Beijing, 101416, Beijing, China*

Received 1 June 2014, www.cmnt.lv

Abstract

Software health management technology main includes the processes of real-time detection, fault diagnosis, health metrics and taking mitigation measurement, and health metrics is the important basis for taking mitigation measurement. An integrated software health metrics is put forward in this paper, and the software health is measured from the layer of task, function and resource. By constructing a multi-factor reliability growth model, the health of task is measured by reliability; the health of resource is measured by the usage of various resources; and the health of function is measured by the risk of failure models and their propagation distance. Finally, the integrated health metrics is built on the basis of the three areas, and the status of software health is divided by the number of health. Then, the status of software health provides theoretical basis for which mitigation measurement should be taken

Keywords: aerospace software, software health, health metrics, task health, resource health, function health

1 Introduction

With the software widely used in the fields of aerospace, the impact of software errors to the system is increasing. Although the software has strict verification and validation in the process of design and implementation, some defects may not be detected in the process of verification and validation [1]. How to deal with these potential software defects that induced to failure in software runtime has become a hot research topic in the field of software engineering. Software Health Management Technology is one of the effective measurements to handle the faults during software running. Some work has been done on the concept and framework of the software health management by some researchers [2-5].

According to the definition of software health management in [5], software health metric is an important part of software health management. Current research on health metrics are focused on the mechanical and electronic equipment, such as launch vehicle [6], satellite platforms [7], aircraft engines [8], unmanned aerial vehicles [9] and so on. However, little research of software health metrics has been done. Currently, only literature [10] studied the framework of software health metrics, but specific methods are not given.

According to the characteristics of aerospace software development and use process, a software health integrated metrics is put forward in this paper, which measures the health of aerospace software from the three layers of task, functional and resource. The result of metrics will provide a theoretical basis for the mitigation measures taken in the software running and is important for the aerospace software to complete task more reliability.

2 Software health integrated metrics

Aerospace software is a complex system that involves many aspects of software, so we have to find some characteristic parameters that can integrate evaluate the overall performance of the software to judge the health status of the software. Literature [10] hold the opinion that the software health is an integrated measure based a variety of quality factors, and can be regarded as a weighting function of variety attributes, such as follows:

$$Health(S) = Func(prop_1(S), prop_2(S), \cdots prop_n(S)), \quad (1)$$

where, $Health(S)$ is the health of the S system; $prop_1(S), prop_2(S), \cdots prop_n(S)$ represents the quality factors used to measure the health of the S system. According to the concept of software health [9], we choose reliability, functionality and availability as the quality factors.

In this paper, we use SWHI (software health index) to quantitatively describe the health status of the software, and it is calculated using integrated weighted method:

$$SWHI(t) = \sum_{i=1}^{r} \varphi_i x_i(t), \quad (2)$$

where r is the number of health-related quality and in this paper they are reliability, functionality and availability, $x_i(t)$ is the function value of one property at time t. It is

* ***Corresponding author's*** e-mail: xieweiqi480@163.com

the health number of sub-attribute value that has been normalized, and s.t. $0 \le x_i(t) \le 1$. When the property is in the best health status, $x_i(t)=1$ and $x_i(t)=0$ when the property is failure; φ_i is the weight of $x_i(t)$ which reflect the effect of sub-attribute on the software health, and it satisfied $0 \le \varphi_i \le 1, \sum_{i=1}^{r}\varphi_i = 1$.

3 Reliability-based TASK health

3.1 MULTI-FACTOR RELIABILITY MODEL

NHPP (Non-Homogeneous Poisson Process) class of software reliability growth model is the most appropriate and the simplest model to evaluate the software reliability. In order to improve the accuracy of evaluation and prediction of NHPP class software reliability growth model, some scholars try to build better models through improvement the NHPP assumptions, such as considerations related failures [11], fault debug process [12], fault detection rate [13], test efficiency [14], and environmental differences [15]. The more integrated factors considered, the higher accuracy of the evaluation and prediction of the model. Combined the features of aerospace software development, testing and operation processes, we build a multi-factor reliability growth model based on NHPP by consider the fault detection rate, the fault exclude process, test efficiency and the differences of test environment and operating environment.

The assumptions of the multi-factor reliability growth model based on NHPP are followed as:

1) Troubleshooting process comply with the non-homogeneous Poisson process;

2) The faults detected in the time interval $[t_0,t_1),[t_1,t_2)\cdots[t_{m-1},t_m)$ are independent;

3) In aerospace software, it's more and more difficult to find fault with the growth over time. Thus, the fault detection function is not a constant but a function of diminishing growth over time, denoted by $b(t)=\alpha e^{-\beta t}$ [14].

4) In the troubleshooting process, sometimes new failure may be introduced. The probability of troubleshooting is denoted as p and the ration of new failure is denoted as q, and then q is the ratio of the number of faults newly introduced and the number of fault excluded.

5) The test efficiency is a decreasing function when consider the effect of the test time, testers, energy and other factors during the test, and can be denoted as $\varphi(t)=\lambda te^{-\gamma t}$ [14].

6) Because the differences between testing profile and operational profile, we take the effect of environmental differences into account and the environmental factor can be defined as $\phi(t)=1+(a+ct)e^{-bt}$ [15].

According to the assumption 4-6, we get the failure intensity function:

$$\lambda(t)=\frac{dm(t)}{dt}=b(t)(a(t)-pm(t))\varphi(t)\phi(t)\ . \tag{3}$$

According to the assumption 3, the total number of software failures is:

$$a(t)=a_0+pqm(t)\ . \tag{4}$$

a_0 is the total number of errors in the software at the beginning of test.

With the equations of $b(t)$, $a(t)$, $\varphi(t)$ and $\phi(t)$,then the equation (3) is

$$\begin{aligned}&\frac{dm(t)}{dt}=\alpha e^{-\beta t}[a_0+pqm(t)-pm(t)]\cdot\\&\lambda te^{-\gamma t}[1+(a+ct)e^{-bt}]=\\&\alpha\lambda te^{-(\beta+\gamma)t}[1+(a+ct)e^{-bt}][a_0+p(q-1)m(t)]\end{aligned} \tag{5}$$

Make $A=\beta+\gamma$, and with the initial condition $m(0)=0$, the solution of Equation (5) is obtained:

$$m(t)=\frac{a_0}{p(q-1)}\left(1-e^{\frac{\alpha\lambda p(q-1)}{A^2}+\frac{1}{(A+b)^2}(a+\frac{2c}{A+b})+f_1(t)+f_2(t)}\right). \tag{6}$$

where:

$$f_1(t)=\alpha\lambda p(1-q)\frac{e^{-At}}{A^2}(At+1)$$

$$f_2(t)=\alpha\lambda p(q-1)\frac{e^{-(A+b)t}}{(A+b)^2}[-(A+b)ct^2+$$

$$(a(-A-b)-2c)t-\frac{2c}{A+b}-a].$$

With the maximum likelihood method, the unknown parameters of the Equation (6) can be calculated. With the time interval $[t,t+x)$, the reliability of the software is the THI (Task Health Index):

$$THI=R(x\,|\,t)=P\{N(t+x)-N(t)=0\}=e^{-[m(t+x)-m(t)]}. \tag{7}$$

3.2 EXPERIMENTAL AND SIMULATION

The fit ability of the model can be measured with SSE (Sum of Squares Error) and the regression index R-Square. The predictive ability of the mode can be measured by AE. The closer of SSE and R-Square to 1, the better fit ability of the model is, and the smaller of AE, the stronger the predictive of the model is.

To test the performance of the multi-factor reliability model, we applied the model on the published data. We select the 'Release1'data in the literature [16] as the test data. The multi-factor reliability model is compared with G-O model, Generalized G-O model [17], Delayed S-shaped model and inflected S-shaped model [18] by analyzed their performance. For the 'Release1'data, the selected data of 16-20 is used to predict. The results are shown in Table 1 and Figures 1 and 2.

TABLE 1 The results of compared models

Model	Parameters
G-O	a = 102.7787, b = 0.1671
Generalized G-O	a = 102.5189, b = 0.1161, c = 1.1290
Delayed S-shaped	a = 100.2045, b = 0.3038
Inflected S-shaped	a = 100.3726, b = 0.2554, β = 1.8210
Model in this paper	a_0 = 97.511, p = 0.9984, q = 0.0088, α = 0.12, λ = 0.1008, A = 0.0995, a = 0.000023, b = 0.009, c = 0.00009

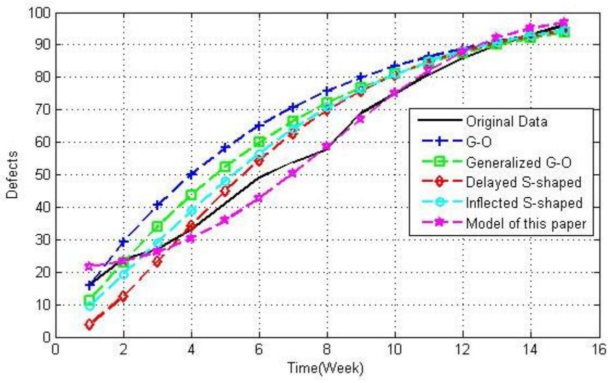

FIGURE 1 The fitting figure

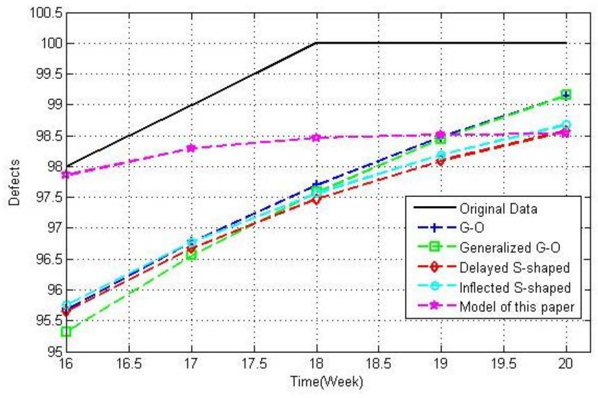

FIGURE 2 The predictive figure

Through the experiment, we can see that the multi-factor reliability model has better fitting and predictive performance. With this model, we can get more accuracy software reliability at any time. For the number of software reliability is between 0-1, so we can use this value as the RBTH (Reliability Based Task Health). For the multi-factor reliability model is based on the data has been detected, if a new fault is detected, the data needs to be added to the model and rebuilt the model.

4 Resource usage based health

In time series analysis, AR(p) model is one of the most widely application model [19].The value x_t at any time t can be expressed as a linear combination of last p values $x_{t-1}, x_{t-2} \cdots x_{t-p}$ with the white noise at time t :

$$x_t = \sum_{i=1}^{p} \beta_i x_{t-i} + \varepsilon_t, \varepsilon_t \sim NID(0, \sigma^2), \quad (8)$$

β_i is the autoregressive parameters, ε_t is the unobservable random error and σ^2 is the variance of ε_t. In complex application environments, it's not easy to build an accurate time series model, then it's impossible to make accurate multi-step ahead forecast. To solve these problems, literate [20, 21] studied the multi-step rolling time series, and applied in wind speed forecast. The results show that the accuracy has been greatly improved.

For the actual value at time x_{t+m-1} is unknown, we instead the actual value x_{t+m-1} with the predicted value $\hat{x}_{t+m-1}$ to predict new value in the rolling m step prediction. Therefore, at time t, the m step prediction equation can be expressed as:

$$x_{t+m} = \begin{cases} \sum_{i=1}^{p_m} \beta_{mi} x_{t-i+1} & (m=1) \\ \sum_{i=1}^{m} \beta_{mi} \hat{x}_{t+m-i+1} + \sum_{i=m+1}^{p_m} \beta_{mi} x_{t+p_m-i+1} & (1 < m \le p_m) \\ \sum_{i=1}^{p_m} \beta_{mi} \hat{x}_{t+p_m-i} & (m > p_m) \end{cases} \quad (9)$$

p_m and β_{mi} are the order and the regression coefficient AR(p) model.

For the time series $\{x_{t-N+1}, x_{t-N+2} \cdots x_t\}$ in the fixed window at time t, δ is the standard deviation of this time series, if $|x_{t+i} - \hat{x}_{t+i}| > \delta$, then the data observed at time $t+i$ is called an abnormal data. If k observed data are abnormal continuously, then an abnormal event has occurred.

The health of resource usage can be measured with RHI (Resource Health Index), if we detected n abnormal data continuously, then the resource health index can be expressed as:

$$RHI = 1 - \frac{n}{k}, \quad (10)$$

If there are m type's resources, the total resource health index can be expressed as:

$$RHI = \frac{1}{m} \sum_{i=1}^{m} \left(1 - \frac{n_i}{k_i}\right), \quad (11)$$

n_i denotes the number of abnormal data of i type resource, and k_i denotes the abnormal event threshold of i type resource.

5 Fault risk based function health

5.1 INTEGRATED RISK MODEL BASED ON DATA

In QJ3050A-2011 "Aerospace product failure modes, effects and criticality analysis Guide", RPN (Risk Priority

Number) is used to analysis the critical of fault. And RPN can be calculated as:

$$RPN = SN \times ON \times DN,$$

where:

SN: (Severity Number) indicates the fault severity;

ON: (Occurrence Number) indicates the occurrence of a fault;

DN: (Detection Number) indicates the degree of fault detection.

In this paper, thus information can be got by analysis the case sets of aerospace software. These cases include the information of the failure model, failure causes, and the introduction and testing stages and other information. This information can serve as aerospace software PDM (Product Data Management).

5.1.1 SN

According to the effect of software failure on the function of the software and the system security, the severity level can be divided into five levels, as shown in Table 2. By analysis 'Fault overview' of each case, we can get the information of the main tasks of the software and the information of the software failure.

Take the average of all error severity in the cases contained in the fault class as the severity of the error class. That means, if the error class ET_i contain n cases, and the severity of each case is ET_i, then the severity of error class can be calculated as:

$$SN = \frac{\sum_{i=1}^{n} SN_i}{n} \tag{12}$$

TABLE 2 Severity classification

Level	Description	Severities	Weights
Disaster	Software failure and affect the security of the system, task fails.	V	10
Fatal	Degrade run and affect the safe of the system, mission failure	IV	8
Medium	Software failure, task delayed	III	5
Ordinary	Some functions of software failures, task postpone	II	2
Slight	Running normal, but data is incorrect, task completion in degrade	I	1

5.1.2 ON

The occurrence of the software fault is the relative incidence. For the number of occurrences of a set of fault class $e = \{e_1, e_2 \cdots e_i\}$, if $e_i = max(e), i \in (1,n)$, then the occurrence of fault class e_t can be calculated by the following equation:

$$ON(e_j) = \frac{e_j}{e_i} \times 10 \quad i, j \in (1,n). \tag{13}$$

5.1.3 DN

DN can be measured by the stage of defect introduced and the stage of defect detected. From the description of 'Fault Analysis' of each case, we can get the information of the defect introduced, and also by analysis the 'Fault overview', we can know when the defect is detected. The longer the defect stays, the more difficult to detect, so we can built the detection coefficient matrix, as shown in Table 3.

TABLE 3 Detection coefficient matrix

	Defects Detected							
Defects Introduced	System Requirements Analysis	Requirements Analysis	Design	Unit Testing	Integration Testing	Confirmation test	Joint test system	Operational
System Requirements	1	2	4	6	8	10	10	10
Requirements Analysis					7	9	10	10
Design					6	8	10	10
Code					5	7	10	10
Integrated management					1	3	10	10
					5	8	10	10
	Weighted							

If there are *n* cases in the error class, and DN_i is the detection of each case, and then the DN_i of error class ET_i can be calculated by the following equation:

$$DN = \frac{\sum_{i=1}^{n} DN_i}{n}. \tag{14}$$

5.2 FAILURE MODE SEVERITY

The software failure mode can be divided into the following categories during running:

1) Code Error: including calculation and arithmetic error, initialization and reset error, programming error.

2) Validation error: mainly are the validity error of event and data, including the timing error.

3) Reasonable error: mainly the interface error.

According to the statistics, the RPN of each error class can be calculated and the results are shown in Figures 3 and 4.

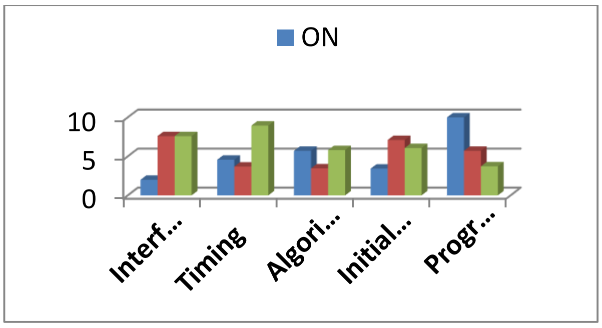

FIGURE 3 SN, ON and DN of each error type

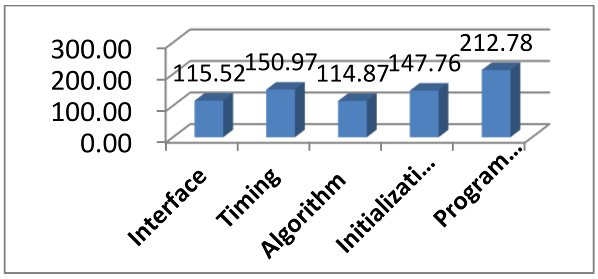

FIGURE 4 RPN of each error type

For the code error contains the errors of calculation and algorithm, initialization and reset, programming, so we chose the maximum RPN from the three types as the RPN of code error. RPN characterize relative severity of the failure mode, the normalized severity of the three failure modes are as follows:

TABLE 4 Normalized RPN of Failure Model

Failure Model	Normalized RPN
Validation error	0.543
Reasonable error	0.71
Code Error	1

5.3 FAULT PROPAGATION DISTANCE INDEX

Usually, the severity of fault will increase after propagation. To describe this situation, fault propagation distance (FPD) is defined as follows:

Definition 1: Fault Propagation Distance is the distance of the failure mode has propagated in the fault propagation directed graph. It's a description of the fault propagation depth.

According to the basic nature that with the fault spread, the severity of fault model will increase, we set the severity of the failure model to 0. And the severity wills plus one if the failure model propagates to a new node. However, some nodes are intersection node and they are critical node, we set the maximum distance of the multiple paths as the severity of the node. Then the FPD of typical TFPG [22] is shown as following:

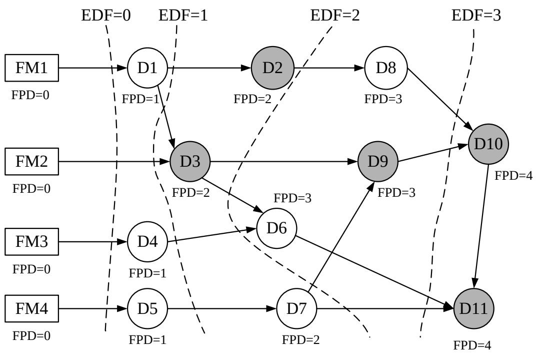

FIGURE 5 Fault propagation distance distribution diagram

According to FPD algorithm, fault propagation distance has the following properties:

Property 1 For $\forall v \in F, FPD(v) = 0$.

Property 2: For $\forall v, v' \in V$, if $(v, v') \in E$, then $FPD(v) \le FPD(v')$.

For D is a non-empty set in TFPG, we can get property3 according property1 and property2.

Property 3 For $\forall v \in D, FPD(v) \ge 1$.

According to these properties, the FPD is an increase function when the fault propagates along any given path in TFPG, and the nodes with the same propagation distance form an EDF (Equal Distance Front). The set of EDF is $\{n \in N\}$ and the maximum number of the set is called the maximum propagation max(EDF). We can find that TFPG is divided into several areas by EDFs, as shown in Figure 5.

In order to limit the value of FPD to [0,1], FPD is normalized; Then we define the FPDI (Fault Propagation Distance Index) of node v ($v \in V$) as follows:

$$FPDI = \frac{FPD(v)}{max(EDF)}, \tag{15}$$

FPDI describes the depth of the fault has been propagated in TFPG when propagated to node v. Greater *FPDI* indicating the greater distance the fault has spread, and the greater harm to the software.

5.4 FUNCTION HEALTH

After the malfunction occurred, the severity of impact will be more and more serious with the fault spread. The severity of fault model means the impact to software, and the FPDI means the distance of the fault has spread. Thus, the FHI (Function Health Index) of software can be measured by the following equation:

$$FHI = 1 - RPN \times FPDI. \tag{16}$$

6 Software health statuses

The software health index is the weighting function of reliability, functional and validity, thus SWHI can be measured by the following formula according Equation (1):

$$SWHI = \varphi_1 \times THI + \varphi_2 \times RHI + \varphi_3 \times FHI \,. \tag{17}$$

$\varphi_1, \varphi_2, \varphi_3$ are the weighting coefficients of the three attributes, and they obey $s.t\ \varphi_1 + \varphi_2 + \varphi_3 = 1$.

THI, RHI, FHI indicate the health status from three different aspects. THI reflects the software health status from the overall quality of the software; RHI describes the health status from the resource usage, and FHI reflects the health status from the function implementation. For the different starting point and different level of description, the important to reflect the health of the software is different. Therefore, the weighting coefficient is different. According to the important to the health of software, $\varphi_1 = \frac{1}{6}, \varphi_2 = \frac{1}{3}, \varphi_3 = \frac{1}{2}$ in this paper. Thus Equation (17) becomes:

$$SWHI = \frac{1}{6} \times THI + \frac{1}{3} \times RHI + \frac{1}{2} \times FHI \,. \tag{18}$$

According to the value of SWHI, health status of the software can be divided as Table 5. According to the health status of software, the software health system will decide which mitigation will be taken.

TABLE 5 Software Health Status

SWHI	Healthy	Running status	Failure rate
[0.8,1.0)	Health	Software running in good condition, does not appear obvious abnormalities	Failure is unlikely
[0.4,0.8)	Sub-health	Software running deterioration	Greater probability of failure
[0,0.4)	Abnormal	Software running abnormal	Less likelihood of major failure

6 Conclusions

According to the needs of software health metrics of software health management system, a software health integrated metrics is put forward based on the layers of task, resource and function for the aerospace software.

1) According to the characteristics of development and usage of aerospace software, a multi-factor reliability growth model is build. By comparison with other models, the results show that this model has high accuracy and strong predictive ability;

2) According to the multi-step rolling model based on the AR (p) model, the resource used by the software is on-line monitored and predicted. Then, a model is built to measure the software resource health with the results of prediction.

3) By accounted and analysed the failure data of aerospace software, the fault severity is measured with the integrated risk model. Fault propagation distance is defined based on the TFPG model. Finally, the software function health is measured with the index of failure model severity and the fault propagation distance.

The software health status is divided based on the three health metrics, and then provides important theoretical basis for the software health management system of what mitigation measurements should be taken.

References

[1] Zhao C, Dong W, Wang J, Sui P, Qi Z 2009 Software Active Online Monitoring Under Anticipatory Semantics *Proceedings of the 1st International Workshop on Software Health Management 2009*

[2] Riecks J, Storm W, Hollingsworth M 2011 Concept Development for Software Health Management *NTIS: NASA/CR-2011-217150* 2011

[3] Schumann J, Mbaya T, Mengshoel O 2011 Bayesian software health management for aircraft guidance, navigation, and control *Annual Conference of the Prognostics and Health Management Society*

[4] Mengshoel O J, Schumann J 2011 Software Health Management with Bayesian Networks *Proceedings of the 2nd International Workshop on Software Health Management*

[5] Xie W, Dang C, Shi J, Cai Y 2013 Study on the application of health management techniques in aerospace testing software *Proceedings of 2013 International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering* Sichuan China 1928-33

[6] Pendse R, Thanthry N 2009 Aircraft health management network: A user interface *IEEE Aerospace and Electronic Systems Magazine* **24**(7) 4-9

[7] Zhang G, An H, Hu Y, Yuan Y 2013 Health Condition Evaluation for a LEO Satllite Based on Improved FMECA *Jouranl of Astrodynamics* **3**(3) 118-21

[8] Wang W-W, Shi Y, Shang Q-K, Liu F, Hou J-J 2013 Application of Smooth Support Vector Machines to Diagnosis of Deterioration Fault of Aeroengine Performance *Fire Control＆Command Control* **38**(7) 1176-9

[9] Li Y, Chen H, Zhang Q-J, Zhao K-R 2011 Assessment method of health condition for UAV systems *Systems Engineering and Electronics* **33**(3) 562-7

[10] Chen G, Bai X, Liu Y, Zhou L 2013 Survey on Health Management of Service-Based Software System *Journal of Frontiers of Computer Science and Technology* **7**(7) 577-91

[11] Zhao J, Zhang R, Gu G 2007 Study on Software Reliability Growth Model Considering Failure Dependency *Chinese Journal of Computers* **30**(10) 1713-20 *(in Chinese)*

[12] Teng X 2001 A Non-Homogeneous Poisson Process Software Reliability Growth Model for N-Version Programming Systems *Dissertation for the Doctoral Degree of Philosophy of the State University of New Jersey* 33-117

[13] Huang C Y, Kuo S Y 1997 Analysis of a Software Reliability Growth Model with Logistic Testing-Effort Function *Proceedings of the Eighth International Symposium on Software Reliability Engineering* 378-88

[14] Tan W-X, Shen Y-L 2011 Research of Software Reliability Model with Test Efficiency *Computer Technology and Development* **21**(8) 67-70

[15] Han X, Lei H 2011 Software reliability growth model considering environmental difference *Journal of Computer Applications* **31**(7) 1759-61

[16] Wood A 1996 Software Reliability Growth Models Tandem Computer Inc *Corporate Information CenterCuper* Cupertina Calif

[17] Huang C-Y, Lyu M R, Kuo S-Y 2003 A unified scheme of some non-homogenous Poisson process models for software reliability estimation *IEEE Transactions on Software Engineering* **29**(3) 261-9

[18] Yamada S, Ohba M, Osaki S 1983 S-shaped software reliability growth modeling for software error detection *IEEE Transactions on Reliability* **32(**5) 475-84

[19] Cheng B-L 2003 Talking about AR(P) Model and Application. *Journal of Chongqing Vocational& Technical Institute* **3**(12) 117-8 *(in Chinese)*

[20] Liu H, Tian H-Q, Li Y-F 2010 Short-term forecasting optimization algorithm for wind speed from wind farms based on wavelet analysis method and rolling time series method *Journal of Central South University (Science and Technology)* **41**(1) 370-5

[21] Yang S-Y, Wang L-X, Niu T-W, Deng F 2012 Multi-step prediction of rolling time series based on particle filter optimization *Systems Engineering and Electronics* **34**(6) 1097-101

[22] Xie W-Q, Cai Y-W, Xing X-C, Xin C-J 2014 Algorithm Design of Failure Diagnosis Reasoner for Software Health Management System. *Measurement & Control Technology* **33**(5) 119-23

Authors

Weiqi Xie, May 1987, China.

Current position, grades: doctor candidate at the Equipment Academy.
University studies: master's degree in Armament Launch Theory and Technology at The Academy of Equipment Command & Technology, China in 2011.
Scientific interests: theory of aerospace test, launch and control, aerospace software testing.

Yuanwen Cai, September 1967, China.

Current position, grades: professor at the Equipment Academy, China.
University studies: master's degree at National University of Defence Technology, in 1990. PhD degree at Beijing University of Aeronautics and Astronautics in 2011.
Scientific interests: test launch and control technology of flight vehicle, computer simulation technology.

Long Cheng, March 1981, China.

Current position, grades: researcher at the Equipment Academy, China.
University studies: PhD degree at The Academy of Equipment Command & Technology, China in 2010.
Scientific interests: aircraft control test, automatic control and system simulation.