A New Method for Abnormal Behavior Propagation in Networked Software

Cheng Peng[1,2,3], Ming Liu[1,3], Xin-Pan Yuan[1,3], Long-Xin Zhang[1,3], Jun-Feng Man[1,2,3]

[1] School of Computer Science and Technology, Hunan University of Technology, China
[2] School of Information Science and Engineering, Central South University, China
[3] Key Laboratory of Hunan Province for Intelligent Information Perception and Processing, China
Chengpeng@csu.edu.cn, lihijjie@sina.com, yipingsoong@sina.com, jianiingzhao@sina.com, mjfok@qq.com

Abstract

In general, bugs in software are inevitable. The study on the abnormal behavior propagation mechanism trigged by bugs in networked software provides the way for people to grasp the execution rule and to adopt corresponding pinning measurements. Based on the situation of abnormal behavior propagation at different granularity software entities, three factors - propagation probability, interactive frequency and connection rate which affects the abnormal behavior propagation, are proposed, and corresponding definition and computation method are also investigated. The process model for abnormal behavior propagation is constructed in reference to the compartment model, individual models and three factors mentioned above, which improves the expression ability of model and enhances its completeness and accuracy. Then, the analytical method for abnormal behavior propagation is applied to the online electronic shopping system, and the results testify the correctness and feasibility of the propagation mechanism.

Key words: Networked software, Abnormal behavior, Propagation model, System bug

1 Introduction

Abnormal behaviors refer to defects or bugs in the software system itself, which result in deviation between system behavior at runtime and normal behavior, and eventually cause failure. Generally, bugs are inevitable and cost much to detect and repair. The outbreak of them will cause incalculable loss and damage. The networked software based on the Internet [1], whose behaviors are complex [2], uncontrollable and continuously evolutionary [3], integrates various heterogeneous resources and services. The small errors within elements may cause the collapse of the whole system, which brings great challenge to data security and the quality of service. Accordingly, it is indispensable to control and eliminate the abnormal behavior propagation for networked software before the outbreak of the disaster, and it is also urgent to maintain the normal and stable software system operation, and to provide continuous and reliable services for users. At present, there have much beneficial achievement on software bug classification [4], distribution [5], testing [6], modeling and repairing mechanisms [7-8], but there are still some limitations: (1) the analysis object is significantly stand-alone system software, which leads to the analysis deficiency of distributed large-scale software system under the network environment; (2) the exist model can only detect and verify known errors by adopting the error injection, but can not express and analyze the potential unknown errors and propagation rules; (3) the analytical model is constructed in reference to the computer virus propagation model and the biological virus propagation model, which cannot be totally adapted to the networked software and can not realistically portray or reveal propagation characteristics of its abnormal behaviors; (4) the granularity of these models are large, and mainly consider internal error propagation more at system-level and component-level than at program-level; (5) incorrect behaviors break away from the system execution environment, which are treated as a range of fixed modes, so it is very weak for the scalability and universality of models; (6) the analytical model is mainly for low-level language system with the hardware support which costs very much.

Based on preliminary work [9], the abnormal behavior propagation model for networked software is constructed in reference to the compartment model [10] and individual model [11]. From the analysis of coarse-granularities such as components and classes to fine-granularities such as functions and attribute, we propose three factors— error propagation probability, connection rate and interactive frequency, which affect the abnormal behavior propagation in the networked software. These provide strong basis for future work like abnormal behavior pinning control to find the

*Corresponding Author: Jun-Feng Man; E-mail: mjfok@qq.com
DOI: 10.3966/160792642018031902017

impact of various factors on the abnormal behavior propagation by solving the model differential equations, and to apply the model to typical networked software system to testify the correctness and availability of the theory by adopting empirical methods.

The rest of the paper is organized as follows: in Section 2, the current studies on software abnormal behavior are briefly reviewed; the mechanisms for abnormal behavior propagation are described in detail in Section 3; in Section 4, the experimental results and corresponding analysis are shown; finally, the conclusion is given in Section 5.

2 Related Work

In recent years, many researchers have made many attempts for the problem of abnormal behavior from the design phase, coding phase [12], and testing phase [13-14] to the system reliability evaluation phase [15]. They, from different perspective, constructed the analytical model to detect and locate abnormal behaviors based on system-level or component-level that played a certain role in improving the stability and reliability of the system. In addition, Avizienis pointed out that multi-layer faults should be considered when constructing the abnormal behavior model [16], and a single logical value (with/without faults) could not fully express some vital abnormal behaviors. The author also stressed the importance of the error propagation between components and the system. However, most modeling methods did not pay attention to the error propagation of components and the system [17]. As far as we know, it is seldom mentioned that constructing the abnormal behavior propagation model from the perspective of multiple granularities. Based on other factors, Elmqvist and Nadjm-Tehrani [18] proposed a formal modeling method with security interface and provided combinational reasoning on the system security. Grunske and Neumann [19] constructed a system error behavior model by adopting the troubleshooting and conversion symbol FDTN (Failure Diagnosis and Transformation Notation), which could deduct the CFT (Component Fault Trees) to analyze the security. Rugina adopted error model accessories of AADL (Architecture Analysis and Design Language) to construct composite system error behavior models, and transformed these models into stochastic Petri nets or Markov chains for analyzing [20]. Joshi also made a further study on converting these error behavior models to fault trees [21]. These efforts mainly focused on models of known error behaviors, so there is still considerable research scope on factors affecting error behavior propagation and improvements of models' completeness.

3 The Mechanism for Abnormal Behavior Propagation

3.1 The Abnormal Propagation Model for Networked Software

Figure 1 depicts the multiple-granularities error propagation model for networked software system. Each component consists of several classes which contain a variety of functions. Calls between functions, relationships between classes and messages passing between components are represented by directed edges. Components, classes and functions are presented by different granularity nodes.

The error rate int $f(i)$ inside the component *i* refer to the implicit error probability of component *i.* However, errors inside components will not necessarily propagate to other components and result in the system error. Therefore, we need to introduce the parameter $ep(i)$, which represents the error propagation probability inside the component *i*. We assumed that the system is composed of C numbers of interactive components, and we adopted $p(i,j)$ $(1 \le i,j \le C)$ to represent the error propagation probability between components. It is clear that $(\forall i)\sum_j p(i,j) = 1$, and models allow self-conversion, so $p(i,i) \ne 0$.

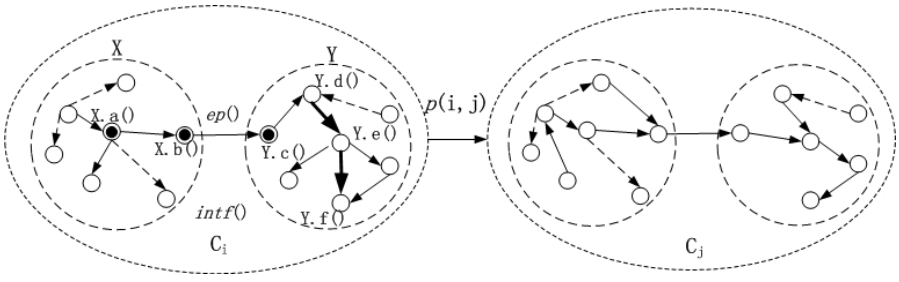


Figure 1. The model for abnormal propagation and related parameters

Relationships between function calls directly reflect the connection rate. The execution process of the software system constitutes a time-varying network N, in which nodes are relatively fixed and edges are time-varying. Where, V is a node set and $E(t)$ is a dynamic directed edge set. Directed solid lines at Figure 1

indicate function calls at some time and directed dotted lines at Figure 1 represent possible call relationships. Circles with black dots in Figure 1 represent implied error nodes.

The interactive frequency refers to the calling and called number between two nodes during a certain period. Some implied error nodes in a handful of execution could not propagate errors to other nodes which interact with them, but when the interactive frequency reaches the upper limitation, it may cause other nodes calling the node to generate errors. The interactive frequency also reflects the execution frequency and the importance of some operation in the networked software. In Figure 1, directed solid lines in bold represent frequent interaction between nodes.

3.2 Model Construction

3.2.1 Related Definition

Definition 1. (The Error Probability inside Components) the error probability inside components $\text{int}\, f(i)$ is a static estimate value.

$$\text{int}\, f(i) = \frac{FD \cdot l_i}{1000} \tag{1}$$

Here, l_i is the number of codes lines and *FD is the error density.*

Definition 2. (The Error Propagation Probability inside Components) the error propagation probability inside components *ep*(*i*) is derived based on FPIC (Failure Proneness Index of Class) which can be presented by CK six standard value [22]:

$$FPIC_X = \alpha \frac{CBO_X}{CBO_i} + \beta \frac{WMC_X}{WMC_i} + \gamma \frac{RFC_X}{RFC_i} \tag{2}$$

CBO_X, WMC_X and RFC_X are the attribute value of CBO, RFC and WMC of class X, CBO_i, WMC_i and RFC_i are the sum of attribute value of CBO, RFC and WMC of all classes in component *i*, α, β and γ are the corresponding weight of these three attributes. According to the effect and the index of failure proneness, the error propagation probability inside components can be described as follows:

$$ep(i) = \frac{1}{M_C} \sum_{X=1}^{M_C} FPIC_X \tag{3}$$

Here, M_C is the sum of classes inside the component *i.*

Definition 3. (The Error Propagation Probability between Components) the error propagation probability between components $p(i,j)$ can be defined as follows:

$$p(i,j) = P\left(\left[C_j\right](p) \neq \left[C_j\right](q) \mid p \neq q\right) \tag{4}$$

Here, $\left[C_j\right]$ is the function of the component *j*, which contains the node state transition that is generated by function calls in the component *j* and the output when component *j* is executed. *p* and *q* are passing messages between the component *i* and *j*. The propagation probability given by formula (4) can be described as the following aspects: the state generated in the component *i*, the transition probability between states and the entropy of the message propagation between component *i* and component *j*.

$$p(i,j) = \frac{1 - \sum_{p \in S_{C_j}} P_{C_j}(p) \sum_{q \in S_{C_j}} P_{C_i,C_j} \left[F_p^{-1}(q)\right]^2}{1 - \sum_{m \in M_{C_i,C_j}} P_{C_i,C_j}[m]^2} \tag{5}$$

Here, S_{C_j} is the states in component *j*, P_{C_j} is the state probability distribution, P_{C_i,C_j} is the probability distribution of passing message M_{C_i,C_j} between component *i* and *j*, the conversion function $F_p^{-1}(q)$ is the message *m* needed to be inputted when the state *p* transfers to the state *q*. $\sum_{q \in S_{C_j}} P_{C_i,C_j} \left[F_p^{-1}(q)\right]^2$ and $\sum_{m \in M_{C_i,C_j}} P_{C_i,C_j}[m]^2$ are the second-order Renyi entropy index. We assume that both the state S_{C_j} of the component *j* and the message M_{C_i,C_j} that the component *i* propagates to the component *j* are equal probability distributions, so the formula (5) can be simplified as follows:

$$p(i,j) = \frac{1 - \frac{1}{\left|S_{C_j}\right|\left|M_{C_i,C_j}\right|^2} \sum_{p \in S_{C_j}} \sum_{q \in S_{C_j}} \left|F_p^{-1}(q)\right|^2}{1 - \frac{1}{\left|M_{C_i,C_j}\right|}} \tag{6}$$

When the state transition function *F* is unknown, it is assumed that the number of messages which every initial state triggered to transfer to another new state is the same, so, the upper limitation can be obtained as follows:

$$p(i,j) \leq \frac{1 - \frac{1}{\left|S_{C_j}\right|}}{1 - \frac{1}{\left|M_{C_i,C_j}\right|}} \tag{7}$$

Definition 4. (Multi-step Propagation Probability) Multi-step propagation probabilities is $err^{(k)}(i,j)$. It is given that execution starts from the component *I*, reaches the component *j* after *K* steps control transfer

and generates error output probability. The probability $err^{(k)}(\ )$ can be associated with $err^{(k-1)}(\ )$ by the following recursion formula,

$$err^{(k)}(i,j)=p^{(k)}(i,j)\cdot \operatorname{int} f(j)+ep(j)\cdot\left(1-\operatorname{int} f(j)\right)\cdot \sum_{h=0}^{C} err^{(k-1)}(i,h)p(h,j) \quad \textbf{(8)}$$

When $k<0$, we can know $err^{(k)}(i,j)=0$, $err^{(0)}(i,j)=\operatorname{int}\ f(j)(\forall i=j)\ err^{(0)}(i,j)=0(\forall i\neq j)$.

Definition 5. (The connection rate) Considering the directed graph $G(V,E(t))$, the connection rate can be expressed as follows,

$$c(t)=\frac{|E(t)|}{|V|\cdot(|V|-1)} \quad \textbf{(9)}$$

$|E(t)|$ is the edge number of the directed graph *G* as the software system interaction formed at time *t* and $|V|$ is the node number. The connection rate describes the degree of connectivity between nodes. When the connection rate is greater, the degree of connectivity between nodes becomes higher, thus the abnormal behavior propagation possibility is larger.

Definition 6. (The Interactive Frequency $\mu(t)$). Software system interaction forms directed graph *G* which can be represented by an adjacency matrix $A(t)$, in which the connected edges between nodes are changing with time. At time *t*, the connection between node *i* and its neighbors can be described by the neighborhood of *i*, $V_i(t)=\{a_{ij}(t)|a_{ij}(t)\in A(t),j=1,2,...,N\}$, which represents the *i-th* row vector of the matrix. And there is a connection between the node *i* and the node *j*. $a_{ij}(t)=1$, on the contrary, $a_{ij}(t)=0$, multiple time slices can form multiple adjacency matrices $A(t)$, which can be added up to constitute the interactive frequency matrix $B=\sum_{t=1}^{T}A(t)$ - between nodes during a certain period *T*, while, the neighborhood $V_i(T)$ of the node *i* in the interactive frequency matrix *B* represents the number that *B* interacts with neighbor nodes at time *t*, so the interactive frequency between the node *i* and node *j* can be further represented as $\mu_{ij}(t)=b_{ij}/\sum_{i=1}^{N}\sum_{j=1}^{N}b_{ij}$, and the interactive frequency of *G* is $\mu(t)=\sum_{i=1}^{N}\sum_{j=1}^{N}d_i\mu_{ij}(t)$, d_i being the degree of the node *i*.

3.2.2 Model Construction

The software abnormal behavior propagation is analyzed by the SI model of the two-state compartment and the individual model [23]. $S(t)$ indicates the number of healthy nodes, $I(t)$ represents the number of implied error nodes. Figure 2 shows the two-state compartment model and α is the infection intensity.

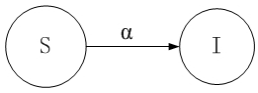


Figure 2. SI compartment model diagram

As what we have considered is the network software system, whose interactive behaviors are complex and the node number is relatively large, so deterministic compartment propagation models are adopted and the balance equation of each compartment is,

$$\begin{cases}\dfrac{dS(t)}{dt}=-\alpha S(t)\\ \dfrac{dI(t)}{dt}=\alpha S(t)\end{cases}$$

At time *t*, as the connecting rate is known, the implied error node *i* can connect with $c(t)(N-1)$ nodes of *G* at most and the proportion that these nodes are not infected by errors is $(1-I(t)/N)$, so $c(t)(N-1)(1-I(t)/N)$ is the number of healthy nodes $S(t)$ connected with the node *i*. the interactive frequency between nodes is $\mu(t)$, the propagation probability is β. Here, different methods are adopted to calculate the propagation probability according to different granularity nodes as the above definition. Thus, the ratio of health nodes which are infected is $\beta\mu(t)S(t)I(t)/N$ and the infection intensity is

$$\alpha=\frac{\beta\mu(t)}{N}I(t)$$

The infection rate is

$$a=\frac{\beta\mu(t)}{N} \quad \textbf{(10)}$$

So we can conclude that

$$\frac{dI(t)}{dt}=aS(t)I(t) \quad \textbf{(11)}$$

$S(t)=c(t)(N-1)(1-I(t)/N)$ is known, so the abnormal behavior propagation analysis model is,

$$\frac{dI(t)}{dt}=ac(t)(N-1)(1-I(t)/N)I(t) \quad \textbf{(12)}$$

When N is large, $N-1\approx N$, then the above formula can be simplified as,

$$\frac{dI(t)}{dt}=Nac(t)I(t)-ac(t)I^2(t) \quad \textbf{(13)}$$

The general solution is

$$I(t)=\frac{N}{1+CNe^{-ac(t)Nt}} \quad \textbf{(14)}$$

Here, C is constant. It is assumed that there is only one implicit error node, that is $I(0)=1$, when the initial time is $t=0$.

$$I(0)=1\Rightarrow\frac{N}{1+CN}=1\Rightarrow C=\frac{N-1}{N}$$

The final solution is

$$I(t)=\frac{N}{1+(N-1)e^{-ac(t)Nt}}$$

The diffusion rate of error nodes can be represented as

$$i(t)=\frac{I(t)}{N}=\frac{1}{1+(N-1)e^{-ac(t)Nt}} \quad \textbf{(15)}$$

4 Experiments and Analysis

The model for abnormal behavior propagation can be testified based on the networked software system - Web Shop. Figure 3 is the architecture of Web Shop platform, which mainly constitutes components C1-C8. Multiple users log in or register to enter the shopping platform through the different geographical location of the GUI at the same time, interact with Identifier and DBMS of cloud data center, are triggered to confirm the task, then simultaneously enter the Market to buy and order goods and form the core of the cloud data center with the Account Manager. The latter uses the data which the database system provides and manages the user's operation request by interacting with all other components (such as Messenger, Verifier and Payment Platform).

Figure 4 shows the dynamic model which is formed by partly components interaction and the state transition diagram inside all components. The message sets between components can be obtained by Rose-RT and other similar tools. The input message is Msg1 (C3_C4), output messages are Msg1 (C4_C5), Msg2(C4_C5), Msg3(C4_C5), Msg4(C4_C5), Msg5(C4_C5). Therefore, upper limitation values of the error propagation probability between the component C4 and component C5 can be calculated according to the formula (7). Here, Ci=C4, Cj=C5, $S_{C_5}=2$, $M_{C_4,C_5}=5$, $P(4,5)\le(1-0.5)/(1-0.2)=0.625$. The error propagation probability between any two components of the system can be obtained based on the above reasoning. In addition, according to the formula (1), the error rate *intf*( ) inside components can be also gained. As is shown in the Table 3.

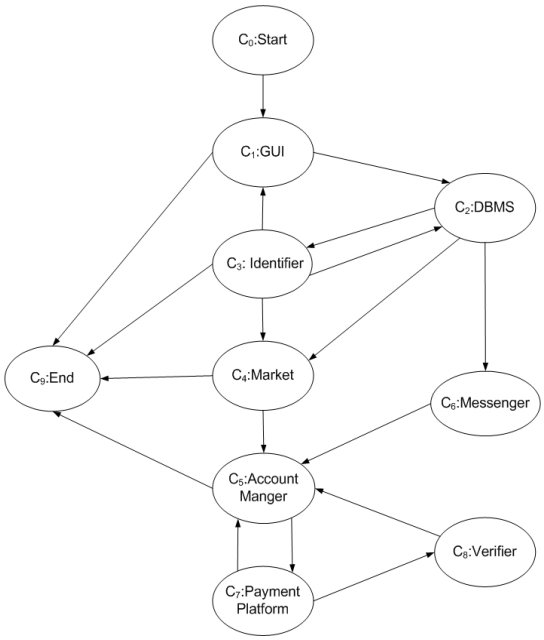


Figure 3. The architecture of Web Shop platform

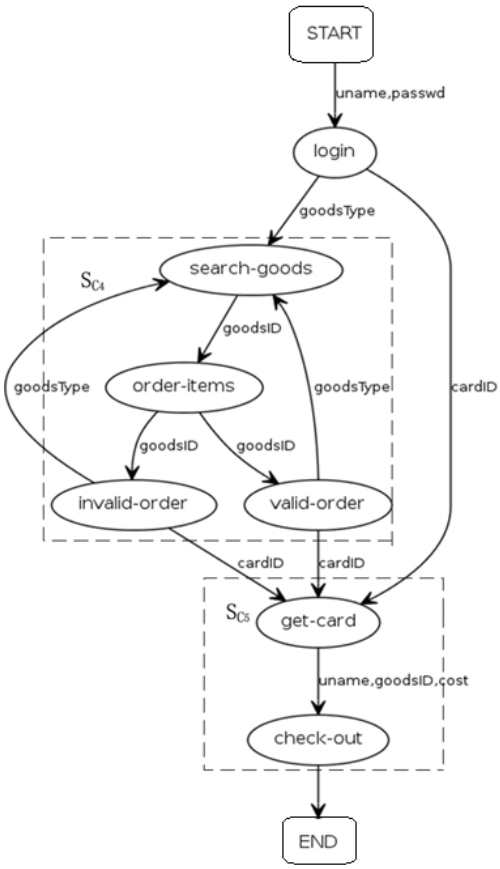


Figure 4. dynamic model [9] and state conversion diagram inside all components

4.1 Error Propagation Probability

Here, taken component GUI as an example, which realizes the user log in, registration and sending verification E-mail. Table 1 gives all classes CK standard attribute values of GUI, which are obtained by the open source software CKJM and standard plug-in Eclipse. The error propagation probability *ep*( ) inside corresponding components can be calculated according to the formula (3). We can see it from the Table 2.

Table 1. All classes CK standard attribute values inside the GUI

Class name	RFC	CBO	WMC	FPIC
javax.mail.Properties	16	15	2	0.13896
javax.mail.Session	7	4	1	0.05193
javax.mail.Transport *ep*( )	21	18	6	0.21087
javax.mail.MimeMessage	5	3	2	0.05103
javax.mail.Internet Address	3	7	4	0.08042
javax.mail.Internet.MimeMultpart	2	5	4	0.06883
javax.mail.Internet.MimeBodyPart	6	4	2	0.05868
javax.activation.DataHandler	17	16	6	0.18826
javax.activation.FileDataSource	14	12	5	0.15099

Table 2. The error propagation probability inside the GUI

C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9
0	0.11	0.56	0.25	0.74	0.62	0.08	0.43	0.23	0

Table 3. The error propagation probability between components

	C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	*intf*( )
C_0	0	1	0	0	0	0	0	0	0	0	0
C_1	0	0	0.999	0	0	0	0	0	0	0.001	0.019
C_2	0	0	0	0.235	0.667	0	0.098	0	0	0	0.032
C_3	0	0.056	0.026	0	0.917	0	0	0	0	0.001	0
C_4	0	0	0	0	0	0.625	0	0	0	0.375	0.007
C_5	0	0	0	0	0	0	0	0.9377	0	0.0623	0.005
C_6	0	0	0	0	0	1	0	0	0	0	0
C_7	0	0	0	0	0	0.01	0	0	0.99	0	0
C_8	0	0	0	0	0	1	0	0	0	0	0.1021
C_9	0	0	0	0	0	0	0	0	0	1	0

4.2 Interaction Frequency

During a certain period T, we can divide the execution process of the software, which can be treated as the addition of multiple-time-slices interactive behavior diagram. We also can take the component C_4 as an example. It contains 4 states - *search-goods, order-items, invalid-order* and *valid-order*. The call relationships among them constitute the interactive behavior diagram $G_4\left(V,E(t)\right)$, $V=\{V_1,V_2,V_3,V_4\}$, in which total call relationships are $\left|E(t)\right|=9$. The corresponding execution footprints during this period are shown in Figure 5. Then, according to definition (6), the interactive matrix B between nodes inside the component C_4 can be obtained at *T*=200s. It is assumed that there is interaction between the node V_1 and other external nodes. If $b_{11}=1$, the interactive frequency $\mu(t)$ between nodes of the component C_4 can be calculated.

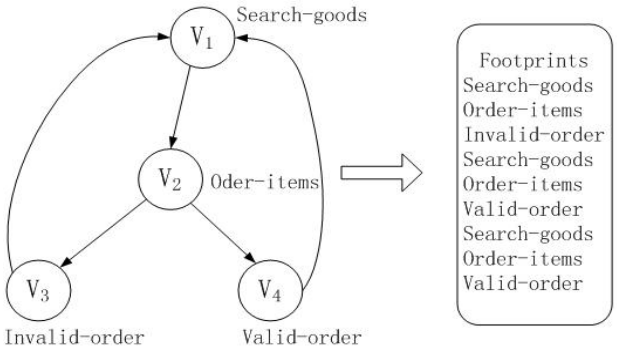


Figure 5. Interactive behavior diagram and corresponding execution footprints

$$B=\begin{bmatrix}1&3&0&0\\0&0&1&2\\1&0&0&0\\1&0&0&0\end{bmatrix}\quad[\mu_{ij}]=\begin{bmatrix}1/9&1/3&0&0\\0&0&1/9&2/9\\1/9&0&0&0\\1/9&0&0&0\end{bmatrix}$$

$[\mu_i]=[4/3\quad 1\quad 2/9\quad 2/9]$, thus, at T=200s, the

interactive frequency of the component C_4 is $\mu(t) \approx 2.99$. About 18,000 execution footprint logs of the component C_4 are collected during 7×24 hours in the experiment. According to the statistical analysis, we can know that the interactive frequency follows the Poisson distribution, that is, $\mu(T=t) = \frac{e^{-\lambda}\lambda^t}{t!}$ $(t = 0,1,2,...)$.

4.3 Connection Rate

It was found that the connection rate of the interactive behavior diagram set $\{G_1, G_2, G_3, \cdots, G_t\}$ met power-low distribution during different period, that is $c(t) = Ht^{-\theta}$, $\theta = 1.53846$, $H = 345.167$, and the correlation coefficient is 98.9605%. Figure 6 shows the connection ratio curve: due to the abnormal behavior propagation in the software system, the connection ratio gradually decreases as time goes on, thereafter the connection ratio is stable at a certain value, which explains the normal execution of the program is disturbed by errors, calls between functions significantly reduce and the task of these components can not be normally completed, which make the whole software system in paralysis or half-paralysis condition. We can also know the linear equation obtained by fitting connection ratio is similar to $c(t)$, when t is large ($t > 600$), the relative deviation is less than 10%, as shown in Figure 7.

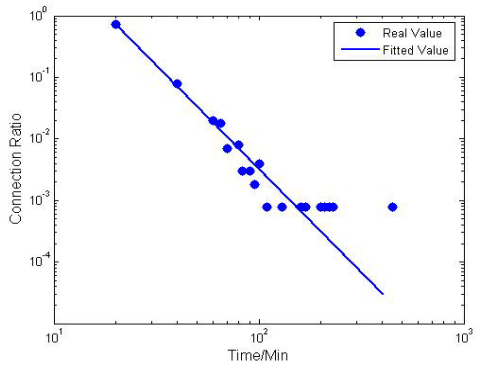


Figure 6. Connection ratio

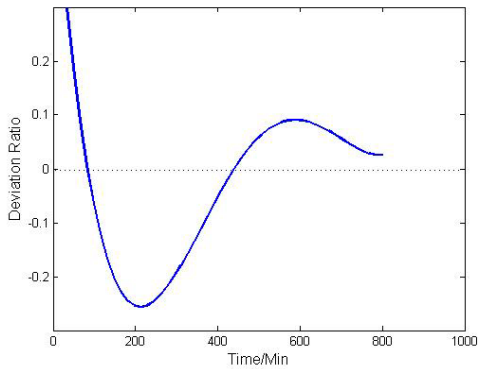


Figure 7. Relative deviation

4.4 Diffusion Rate and Infection Rate

For Web Shop system at runtime, 63 time points within 90 minutes are randomly selected as test samples to generate the interaction behavior diagram, which is used to obtain propagation probability, connection ratio and interactive frequency. Thus, the distribution and fitting curve of spreading ratio is calculated, according result is shown in Figure 8. At time T = 90 min, the spreading ratio reaches its maximum value, close to 94%, which indicates that most nodes in the system are affected by abnormal behavior propagation. Similarly, according to the formula (10), the infection rate affected by abnormal behaviors within 400 minutes also can be described as in Figure 9. In a relatively short time, once errors inside the software are triggered, implied error nodes will accelerate the infection to healthy nodes, and the infection rate will also rapidly increase. At time T=96 min, the infection rate reaches a peak to 0.40, which corresponds to the spreading ratio. In addition, the infection rate is positively related to the interactive frequency, which follows the Poisson distribution. The variation tendency of infection rate also testifies experimental results.

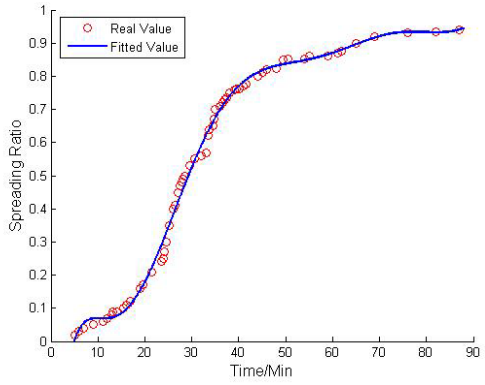


Figure 8. Spreading ratio

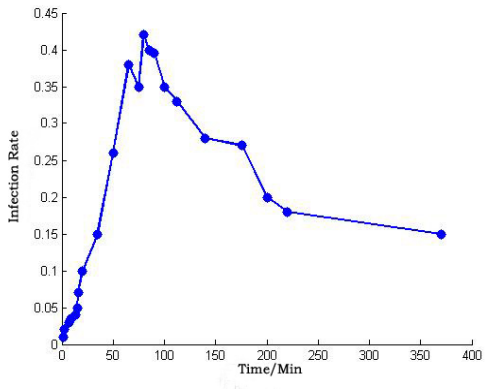


Figure 9. Infection rate

4.5 Accuracy of Model Comparison

The accuracy rate of the model can be defined as the

ratio of the error node number which is contained in propagation path of abnormal behavior and the average value of error nodes which results from the spread of abnormal behavior by injecting errors. First, we adopted PROPANE to inject errors, traced the propagation of abnormal behavior and calculated error node sum until abnormal behaviors propagation results into system failure. As is shown in Figure 10, when injecting error number toward the Highest out-degree inside the component C_4, through two intermediate nodes, abnormal behavior begins to spread; after 10 intermediate nodes, the error ratio of the system reaches the maximum of 100%; when injecting errors toward the Lowest out-degree inside the component C_4, through four intermediate nodes, abnormal behavior begins to spread; after 13 intermediate nodes, the error ratio of the system reaches the maximum of 100%, finally to system standstill. We repeat above experiment, and the whole data is the average of 100 experiments. Finally, the number of average error node of the system is 12.

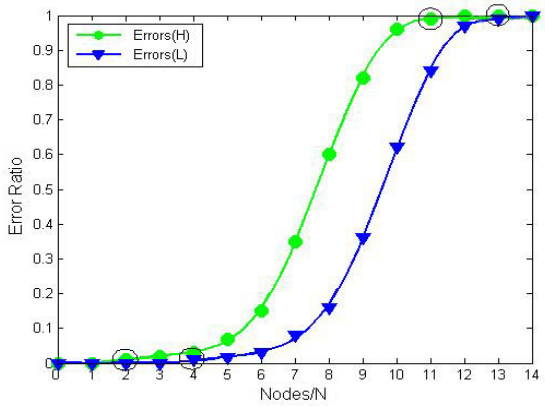


Figure 10. Error nodes analysis

To compare with the method presented in the paper, we adopted DTMP, SPN and BN to analyze abnormal behavior propagation. The result is the average of 100 times and the accuracy rate comparison is shown in Figure 11. It is indicated that the ABPM considers three factors affecting abnormal behavior propagation, the analysis ability of this model is less disturbed by the system scale and the accuracy ratio is still high, about 99%. DTMP assumes that the system execution follows Markov property, that is, at any given time, only one component can be executed, concurrency and synchronization can not be handled and the failures of components are also independent. Such assumption is not fit for the actual operation of the networked software, so the accuracy of model is affected. SPN has strong analytical abilities for concurrency, synchronization and uncertainty, but it has low accuracy, because state-space exponentially increases with the model scale. Although BN is able to deal with association of the abnormal propagation between implicit error nodes and healthy nodes, the assumption bases on the independence of node failures is sensitive to the system scale, so the accuracy of this model needs to be improved.

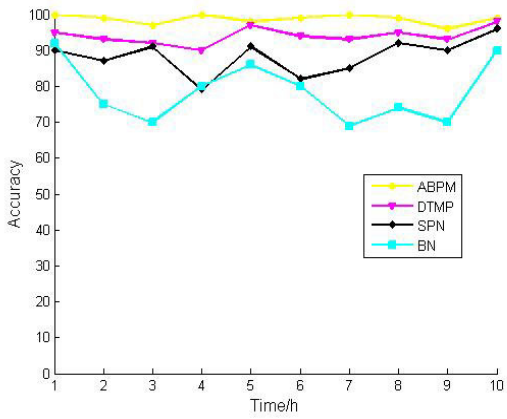


Figure 11. Accuracy compare

5 Conclusion

The study on the abnormal behavior propagation mechanism for networked software plays a significant role in improving the stability and reliability of such new software system. Based on the insufficiency of the existing model, factors affecting the abnormal behavior propagation for networked software are firstly proposed, then the detailed definitions and calculation methods are given and finally the abnormal behavior propagation model is derived. Compared to other models, the abnormal behavior propagation model for networked software comprehensively takes into consideration the dynamic process of software system execution from different granularities, which has stronger expressing abilities. Especially in the analysis of the abnormal behavior of unknown errors of the system, the model is more accurate and reasonable. However, there is still a long way to go as for how to find other factors affecting abnormal behavior propagation, how to further improve the model completeness, and how to effectively control abnormal behavior to improve the stability of the system.

Acknowledgement

This paper is supported by Natural Science Foundation of China (No. 61503131, No. 61702178), The Natural Science Foundation of Hunan Province (No.2016JJ5035, 2016JJ5036, 2015JJ2046, 2017JJ3065, 2018JJ4063), The Major Project of Hunan Province Education Department (No.16A059, 17A052), The Excellent Youth Project of Hunan Province Education Department (No. 16B071), State Key Laboratory of Software Engineering, Wuhan University Open Issue (No. SKLSE2014-10-06).

References

[1] Y.-T. Ma, K.-Q. He, B. Li, J. Liu, Empirical Study on the Characteristics of Complex Networks in Networked Software, *Journal of Software*, Vol. 22, No. 3, pp. 381-407, March, 2011.

[2] J. Q. Fang, X. F. Wang, Z. G. Zheng, Research of Dynamical Complexity of Nonlinear Networks, *Complex Systems and Complexity Science*, Vol. 7, No. 2-3, pp. 5-9, 2014.

[3] C. W. He, L. J. Zhang, H. Zhang, An Approach to Aspect-oriented Software Evolution Based on Metadata and Reflection, *Acta Electronica Sinica*, Vol. 39, No. 8, pp. 1771-1777, August, 2011.

[4] A. Sureka, Learning to Classify Bug Reports into Components, *International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, Prague, Czech Republic, 2012, pp. 288-303.

[5] H. Y. Zhang, On the Distribution of Software Faults, *IEEE Transactions on Software Engineering*, Vol. 34, No. 2, pp. 301-302, March, 2008.

[6] P. Chen, H. Han, X. B. Shen, X.-C. Yin, B. Mao, L. Xie, Detecting Integer Bugs Based on Static and Dynamic Program Analysis, *Acta Electronica Sinica*, Vol. 38, No. 8, pp. 1741-1747, August, 2010.

[7] C. Lin, Y. Z. Wang, Y. Yang, Y. Qu, Research on Network Dependability Analysis Methods Based on Stochastic Petri Net, *Acta Electronica Sinica*, Vol. 34, No. 2, pp. 322-332, February, 2006.

[8] J. S. Monson, M. Wirthlin, B. Hutchings, A Fault Injection Analysis of Linux Operating on an FPGA-embedded Platform, *International Journal of Reconfigurable Computing*, Vol. 2012, Article ID 850487, 2012.

[9] C. Peng, L. M. Yang, J. F. Man, Dynamic Modeling of Networked Software Interactive Behavior, *Acta Electronica Sinica*, Vol. 41, No. 2, pp. 314-320, February, 2013.

[10] Y. Nakata, P. Getto, A. Marciniak-Czochra, T. Alarcón, Stability Analysis of Multi-compartment Models for Cell Production Systems, *Journal of Biological Dynamics*, Vol. 6, No. 1, pp. 2-18, January, 2012.

[11] S. M. Balakrishnan, A. K. Sangaiah, Aspect-oriented Middleware Framework for Resolving Service Discovery Issues in Internet of Things, *IJIPT*, Vol. 9, No. 2-3, pp. 62-78, February, 2016.

[12] B. Yang, D. Liu, W. Zhang, Information Dissemination in Mobile Social Networks with Gossip Algorithms, *International Journal of Ad Hoc and Ubiquitous Computing*, Vol. 21, No. 4, pp. 259-265, January, 2016.

[13] P. R. Su, D. G. Feng, An Anomaly Intrusion Detection Model Based on Nonhierarchical Clustering, *Acta Electronica Sinica*, Vol. 34, No. 10, pp. 1809-1811, October, 2006.

[14] J. J. Xu, Q. P. Tan, L. Xiong, J. Ye, A Quantitative Approach for Program Reliability Analysis of Soft Errors, *Acta Electronica Sinica*, Vol. 39, No. 3, pp. 675-679, March, 2011.

[15] J. Dou, Z. Zhang, J. Dang, L. Wu, Y. Wei, C. Sun, Properties and Achievable Data Rate of a Cyclic Prefix Based Imperfect Reconstruction Filter Bank Multiple access System, *IET Communications*, Vol. 10, No. 17, pp. 2427-2434, November, 2016.

[16] A. Avizienis, J.-C. Laprie, B. Randell, C. Landwehr, Basic Concepts and Taxonomy of Dependable and Secure Computing, *IEEE Transactions on Dependable and Secure Computing*, Vol. 1, No. 1, pp. 11-33, January-March, 2004.

[17] M. Hiller, A. Jhumka, N. E. Suri, EPIC: Profiling the Propagation and Effect of Data Errors in Software, *IEEE Transactions on Computers*, Vol. 53, No. 5, pp. 512-530, May, 2004.

[18] J. Elmqvist, S. Nadjm-Tehrani, Safety-oriented Design of Component Assemblies Using Safety Interfaces, *Electronic Notes in Theoretical Computer Science*, Vol. 182, pp. 57-72, June, 2007.

[19] L. Grunske, R. Neumann, Quality Improvement by Integrating Non-functional Properties in Software Architecture Specification, *EASY*, Vol. 2, No. 1, pp. 23-32, 2013.

[20] A. E. Rugina, K. Kanoun, M. Kaâniche, An Architecture-based Dependability Modeling Framework Using AADL, *10th IASTED Int. Conf. on Software Engineering and Applications*, Dallas, TX, 2006, pp. 222-227.

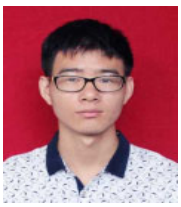
[21] P. Popic, D. Desovski, W. Abdelmoez, B. Cukic, Error Propagation in the reliability Analysis of Component-based Systems, *Proceedings of the 16th IEEE International Symposium on Software Reliability Engineering*, Chicago, IL, 2005, pp. 53-62.

[22] S. R. Chidamber, C. F. Kemerer, A Metrics Suite for Object Oriented Design, *IEEE Transactions on Software Engineering*, Vol. 20, No. 6, pp. 476-493, June, 1994.

[23] Y. Wang, J. Wang, L. Zhang, Cross Diffusion-induced Pattern in an SI Model, *Applied Mathematics and Computation*, Vol. 217, No. 5, pp. 1965-1970, November, 2010.

Biographies

Cheng Peng He received the M.E. and the Ph.D. degree in School of Information Science and Engineering, Central South University, Chang Sha, China. His current research interests include networked software, computer network and software engineering.



Ming Liu was born in Yangzhou, Jiangsu 1993. He is pursuing for master's degree in Hunan University of Technology. His research interests include Big Data, Crowd Computing etc.

Xin-Pan Yuan received the Ph.D. degree in Information Science and Engineering, Central South University in 2012. His current research interests include information retrieval.

Long-Xin Zhang received the Ph.D. degree in Computer Science from Hunan University, Changsha, China. His research interests include real-time systems, power aware computing and fault-tolerant systems, modeling and scheduling for distributed computing systems.

Jun-Feng Man Corresponding author. He received Ph.D. degree in School of Information Science and Engineering, Central South University in 2010, ChangSha, China. His research interests include networked software, big data technology.