Abnormity control design and performance analysis of real-time data exchange software based on Petri net

Zhang Weimin

Abstract: In many spaceflight measure and control software systems, varieties of measure data are exchanged between different software. Qualities of measure and control software systems are influenced by the performances of data exchange software greatly. Many problems that appear during the running process of real-time measure and control software and are difficult to be located are caused by data exchange software. So, it is necessary to analyze the performances of data exchange software while designing measure and control software systems. In this article, the Petri net model of the real-time data exchange software is established first. Then the model is simplified and analyzed. The design of abnormity control for buffer overflow is given. Finally, using the Petri net method, the performances of the real-time data exchange software are analyzed and discussed.

Key words: data exchange, performance analyze, abnormity control, Petri net

1 The Petri net model of real-time data exchange software

In many spaceflight measure and control software systems, varieties of measure data are exchanged between different software. Qualities of measure and control software system are influenced by the performances of data exchange software greatly. Many problems that appear during the running process of real-time measure and control software and are difficult to be located are caused by data exchange software. So, it is necessary to analyze the performances of data exchange software while designing measure and control software systems. In this article, the performances of data exchange software are analyzed quantitatively by using Petri net method. In the meantime, the abnormity control design of buffer overflow is presented.

In a Petri net figure, a circle (called a position) denotes the state of a matter, a rectangle (called a transfer) denotes the transition between states. Positions determine whether transfers can occur. Transfers change states of matter. The dependences between states and transfers are expressed by arrows. Dots in circles are called tokens. A transfer is effective and can startup if there are dots in every position, between which and the specific transfer there is a arrow and the arrow points to the transfer. After a transition, the number of dots in every position previous to the transfer decreases by one and the number of dots in every position posterior to the transfer increases by one.

The Petri net model of data exchange between two software processes is showed in figure 1.

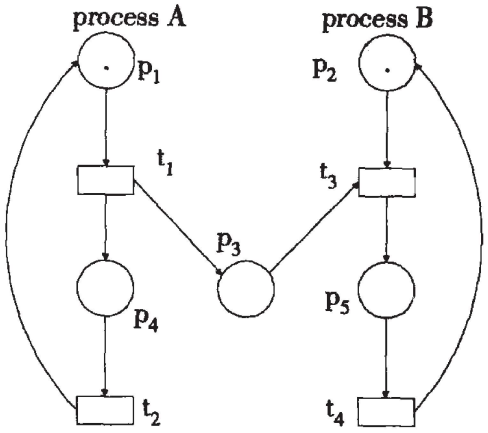

Figure 1. The Petri net model of data exchange between two processes

In figure 1, process A collects or generates data and process B processes or consumes data. Position p_1 and p_2 denote process A and process B being in ready state respectively. Position p_3 is a data buffer, which can be a common data buffer or a message queue in a specific software system. Position p_4 denotes process A being in the state of having generated a group of data. Position p_5 denotes process B being in the state of having obtained a group of data. Transfer t1 denotes the data generating process of process A. After generating a group of data, process A changes its state from p_1 to p_4 and puts the data to buffer p_3. Transfer t_2 denotes the inner processing process of A . After finishing its inner process, process A changes its state from p_4 to p1. Transfer t_3 denotes the data obtaining process of B from buffer p_3. After obtaining a group of data, process B changes its state from p_2 to p_5. Transfer t_4 denotes the data processing process of B. After finishing the data process, process B changes its state from p_5 to p_2.

2 Model simplification and overflow control design

In order to make the analysis and calculation easier, we simplify the Petri net model in Figure 1. Because transfer t_2 consumes the processing time of process A only, the time consumed by transfer t_2 can be added to the time consumed by transfer t_1. So transfer t_2 and t1 can be combined to one transfer. Similarly, transfer t_4 and t_3 can be combined to one transfer. Thus, we get the simplified data exchange Petri net model as showed in Figure 2.

The coverage tree of the Petri net in Figure 2 is

基金资助:北京航天指挥控制中心试验技术研究项目
项目编号:2002SY6104001

showed in Figure 3. The start state of the software system is $M_0(110)$ with one token in p_1, one token in p_2, no token in p_3. In state M_0, transfer t_1 can startup. But transfer t_2 cannot startup in state M_0 because there is no token in p_3. After transition of transfer t_1 the system state changes to (111). In state (111), transfer t_1 and t_2 can startup. After transition of transfer t_1 or t_2 the system state changes to (112) or (110) respectively. Go on like this. The system can reach the states of (110), (111), (112), etc. We use ω to denote arbitrary number of tokens in the coverage tree.

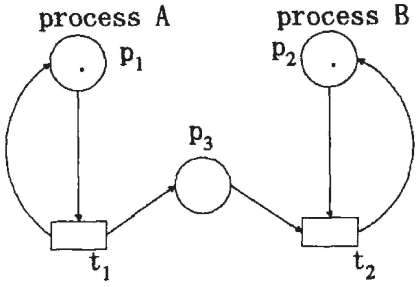

Figure 2. Simplified data exchange Petri net model

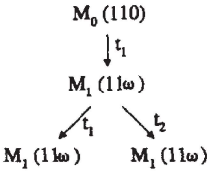

Figure 3. Coverage tree of the Petri net in Figure 2

From the previous analysis we can know that the number of tokens in position p_3 can be arbitrary. That is to say that, no matter how large its capacity is, buffer p_3 may be overflowed. In practical software systems buffer overflow may cause system abnormity. Thus, we must take measures to prevent the occurrence of buffer overflow when we design software.

Tokens in position p_3 are generated by transfer t_1. In order to prevent position p_3 from overflow transfer t_1 must be designed to a controlled transfer. Therefore, position p_4 is introduced to the model. In the initial state, the token number in position p_4 is the size of buffer p_3. The Petri net model with overflow control is showed in Figure 4. The letter n in position p_4 denotes that there are n tokens in p_4. That is to say that buffer p_3 can store n groups of data. In the system showed in Figure 4, the sum of token numbers of position p_3 and p_4 always equals n. The number of data groups in position p_3 cannot exceed n. So, buffer p_3 can never be overflowed.

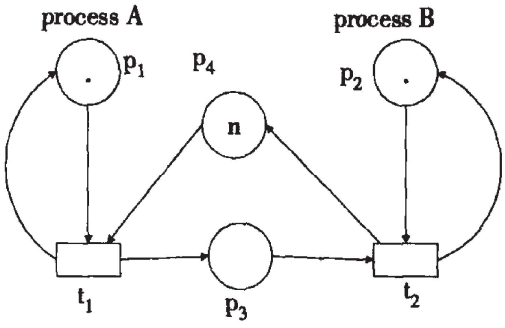

Figure 4. Petri net model with overflow control

3 Data Exchange software performance analysis

In the following part, the performances of the software system showed in Figure 4 are analyzed in the cases of n=2 and n=4.

(1)Case n=2

When n=2, the coverage tree of the Petri net model of the software system is showed in Figure 5.

The startup rate of a transfer is the average number of transitions in a time unit of the specific transfer under the effective condition. The reciprocal of the startup rate of a transfer is the delay time or average service time of the transfer. Suppose the startup rates of transfer t_1 and t_2 are λ_1 and λ_2 respectively. The Markov chain of the system state space is showed in Figure 6.

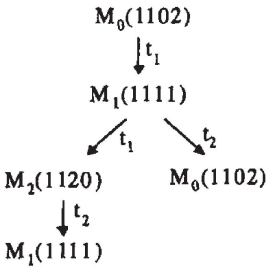

Figure5. Coverage tree of case n=2

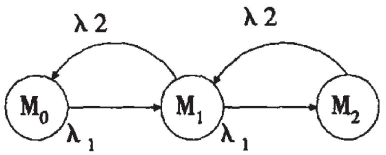

Figure6. Markov chain of case n=2

Now, the transfer rate matrix of the Markov chain $Q=(q_{ij})$ is

$Q=[(-\lambda_1, \lambda_1, 0), (\lambda_2, -\lambda_1-\lambda_2, \lambda_1), (0, \lambda_2, -\lambda_2)]^T$

Suppose the probabilities of the system being in states M_0, M_1 and M_2 are π_0, π_1 and π_2 respectively.

Let $\pi=(\pi_0, \pi_1, \pi_2)$. Then we can get π through solving the next linear equations.

$\pi Q=0$, $\pi_0+\pi_1+\pi_2=1$ ..... ①

With the value of π some performances of the system can be obtained.

The actual average transition times per unit time of transfer t_1 and t_2 are

$F_1=\lambda_1[-\pi_0(\lambda_1/q_{00})-\pi_1(\lambda_1/q_{11})]$ and $F_2=\lambda_2[-\pi_1(\lambda_2/q_{11})-\pi_2(\lambda_2/q_{22})]$.

The probabilities of there being 0, 1 and 2 groups of data in buffer p_3 are $E_0=\pi_0$, $E_1=\pi_1$ and $E_2=\pi_2$.

The average number of data groups (the expectation value) in buffer p_3 is $E=\pi_1+2\pi_2$.

The usage rate of the buffer is f=E/2.

Now we see some examples by giving different values to λ_1 and λ_1.

(a) Suppose the average processing times of transfer t_1 and t_2 are 10ms and 3.333ms respectively. That is $\lambda_1=100$, $\lambda_2=300$. By solving equations ① we can get

$\pi_0=9/13$, $\pi_1=3/13$, $\pi_2=1/13$, $F_1=75$, $F_2=75$, $E=5/13$ and $f=5/26$.

(b) Suppose the average processing times of transfer

t_1 and t_2 are 3.333ms and 10ms respectively. That is λ_1= 300, λ_2=100. By solving equations ① we can get

π_0=1/13, π_1=3/13, π_2=9/13, F_1=75, F_2=75, E=21/13 and f=21/26.

(c) Suppose the average processing times of transfer t_1 and t_2 both are 6.666ms. That is λ_1= λ_2=150. By solving equations ① we can get

π_0= π_1= π_2=1/3, F_1=75, F_2=75, E=1 and f=1/2.

By the above calculations we can get that the system' s average throughputs are all 75 in three different cases. So, in the data exchange system showed in Figure 4, different task distributions between t_1 and t_2 do not affect the system' s average throughput only if the total processing time of t_1 and t_2 is certain.

But the buffer usages are different greatly in three different cases.

In case (a), t_1 processes data slower than t_2. The average number of data groups in buffer is 5/13. The probability of the buffer being full is E_2= π_2=1/13. The buffer is rarely full in this case.

In case (b), t_1 processes data faster than t_2. The probability of the buffer being full is E_2= π_2=9/13. The buffer is often full. In this case, the probability of t1 being waiting buffer space is very high. So some data may be lost because of the buffer' s fullness.

It appears that case (a) is better than case (b). But in an actual real-time software system process A usually collects or receives data from another system. Under the condition that the data rate generated by the outside system is certain, if process A processes data very slowly, the data exchange between the outside system and process A becomes the case (b) and the data may be lost frequently as well. Thus we can come to the conclusion that it is better to divide the processing task between process A and B equally. So the data exchange system should be designed to case (c) .

(2)Case n=4

When n=4, the coverage tree of the Petri net model of the software system is showed in Figure 7.

The Markov chain of the system state space is showed in Figure 8. Now the performances in three cases are given directly. Here π=(π_0, π_1, π_2, π_3, π_4).

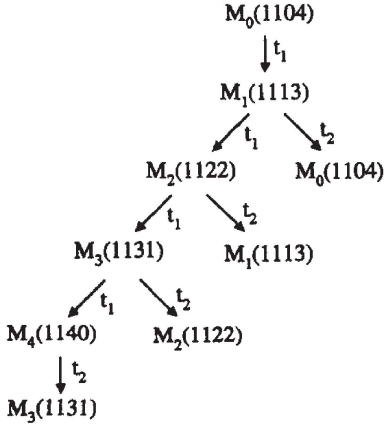

Figure 7. Coverage tree of case n=4

(a) Suppose λ_1=100, λ_2=300, then

π_0=81/121, π_1=27/121, π_2=9/121, π_3=3/121, π_4=1/121, F_1=75, F_2=75, E=58/121 and f=29/242.

(b) Suppose λ_1=300, λ_2=100, then

π_0=1/121, π_1=3/121, π_2=9/121, π_3=27/121, π_4=81/121, F_1=75, F_2=75, E=426/121, and f=213/242.

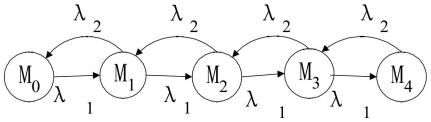

Figure 8. Markov chain of case n=4

(c) Suppose λ_1=150, λ_2=150, then

π_0= π_1= π_2= π_3= π_4=1/5, F_1=75, F_2=75, E=2 and f=1/2.

When n=4 the buffer capacity is doubled. However in case (b) the probability of the buffer being full, E_4= π_4=81/121, is very high. E_4 when n=4 is almost the same as E_2 when n=2 in case (b). From this we can get that if data are lost because process A processes data faster than process B the data losing can not be decreased only by enlarging the buffer size. In order to decrease the data losing the processing task balance between process A and B must be considered first.

4 Conclusions

In many real-time measure and control software systems, data exchange software is important component. In these systems, communications software receives or collects data from outside the system and stores them in data buffer. Data processing software gets data from the buffer and processes them. In order to prevent data from losing some software designers decrease the communications software' s workload and distribute most of the processing task to the data processing software. Meanwhile they set the communications software at a higher priority level to increase its processing speed further more. From the above performance analysis we know that these methods can not decrease data losing but may increase data losing. In some cases enlarging the buffer capacity can not decrease data losing remarkably. Hence when real-time software is designed the system' s performances must be analyzed thoroughly and the process function must be distributed between communications software and data processing software properly. The software' s priorities and the buffer size must be set properly also. When data is lost frequently in a real-time software system we can not solve the data losing problem simply by increasing the communications software' s priority level and/or enlarging the buffer' s capacity. We must analyze the software system quantitatively and find out the real reason to the data losing. The real reason may be that the buffer size is too small, or may be that the processing function distribution is not proper, or may be that the software' s throughput can not meet the system' s demands, etc. Only by knowing the problem' s real reason can we solve the system problem to the core.

References

[1]Yang Wenlong. Software Engineering. Beijing, Publishing(见 163 页)

技术创新

的工作强度。

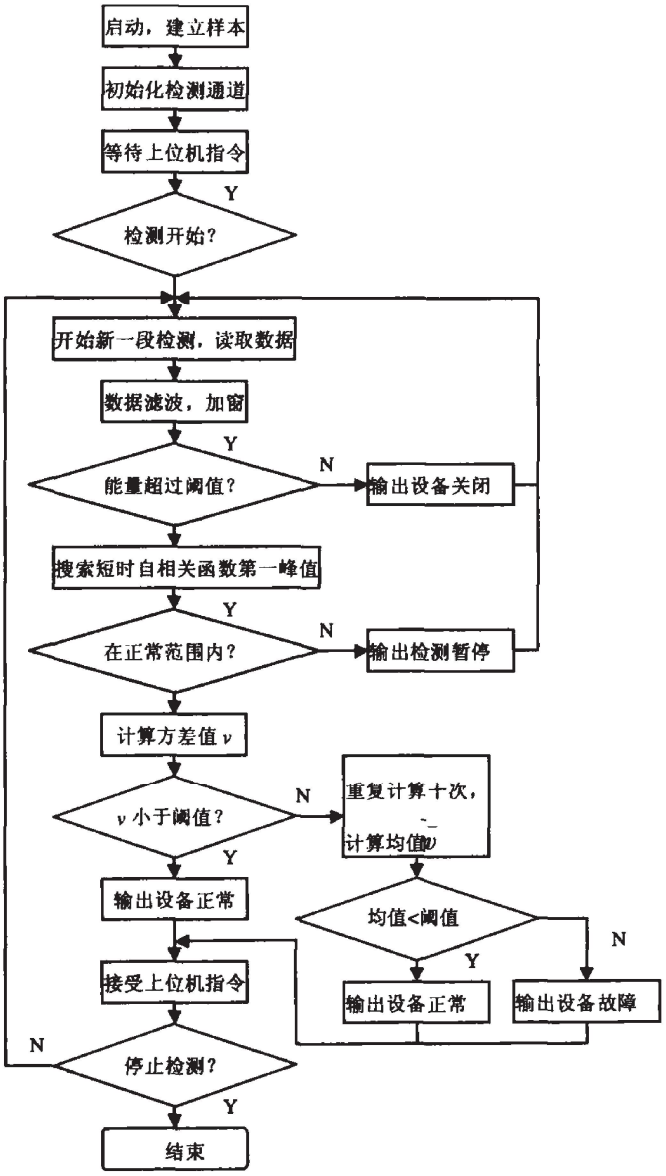

图 1 检测过程流程图

参考文献
[1]于文征.柱塞泵泄漏故障智能诊断系统研究[XL].南京：南京理工大学，2002
[2]徐惠钢，薄煜明，杜国平.基于特征频率功率谱方差的故障实时检测[J]. 东南大学学报，vol.33, 2003
[3]Alan V.Oppenheim. 信号与系统 [M]. 西安：西安交通大学出版社，2000.10
[4]王世一.数字信号处理[M].北京：北京理工大学出版社，1997.7
[5]徐惠钢.基于 DirectX 开发库的多路声音信号实时采集[J].电子工程师，2003，10
[6]孙守迁，王剑，杨勤等译.DirectX 7 速成教程[M].北京:机械工业出版社,2002

作者简介：徐惠钢，1969 年，男，汉，常熟理工学院讲师，南京理工大学自动化系博士生，主要研究数据通信、随机控制；电子邮箱：x_hg@sohu.com or xuhuigang@cslg.cn;薄煜明 1965 年，男，汉，南京理工大学自动化系教授，主要研究控制理论应用、分布式系统;杜国平，1959 年，男，汉，南京理工大学自动化系高级工程师，主要研究控制理论应用、集散控制系统、计算机应用。

(215500 江苏常熟 常熟理工学院物理系) 徐惠钢
(210094 江苏南京 南京理工大学自动化系) 徐惠钢 薄煜明 杜国平
(Physics Department of Changshu Institute of Technology, Changshu 215500, China) Xu,Huigang
(Automation Department of Nanjing University of Science and Technology, Nanjing 210094, China) Xu, Huigang Bo,Yuming Du,Guoping
通讯地址：(210094 南京理工大学自动化系 1001 教研室)徐惠钢

(投稿日期:2005.1.8) (修稿日期:2005.1.23)

(接第 46 页)

参考文献
[1]杨文龙等,软件工程,北京：电子工业出版社,1998.
[2]蒋昌俊,Petri 网的行为理论及其应用,北京：高等教育出版社,2003.

作者简介：张卫民，男，1962 生，博士研究生，硕士，研究员。研究方向为软件工程、软件质量保证、航天测控软件研制与开发。E_mail：zwm1962@sina.com
Author Resume: Zhang,Weimin: Male, burn in 1962, doctor graduate student, master, researcher. Research direction: software engineering, software quality assurance, aerospace measurement & control software developing. E_mail：zwm1962@sina.com
(100083 北京航空航天大学计算机学院)张卫民
(100094 北京航天指挥控制中心)张卫民
(College of computer, Beijing University of Aeronautics and Astronautics, 100083,China)Zheng, Weimin;
(Beijing Astronautic Command & Control Centor, 100094, China)Zheng,Weimin
通信地址：
(100094 北京市 5130 信箱 106#)张卫民

(投稿日期:2005.3.1) (修稿日期:2005.3.15)

(接第 49 页)House of Electronics Industry, 1998

[2]Jiang Changjun. Behavior Theory and Applications of Petri Net. Beijing, Higher Education Press, 2003

Author Resume: Zhang Weimin, Male, burn in 1962, doctor student, master, researcher. Research direction: software engineering, software quality assurance, aerospace measure & control software developing. E_mail：zwm1962@sina.com
(College of computer, Beijing University of Aeronautics and Astronautics, 100083,China)Zheng, Weimin
(Beijing Astronautic Command & Control Centor, 100094, China)Zheng,Weimin

(投稿日期:2005.3.1) (修稿日期:2005.3.15)

技术创新