ISSN 1330-3651 (Print), ISSN 1848-6339 (Online)
DOI: 10.17559/TV-20160704190047

FAULT LOCATION IN CNC SYSTEM SOFTWARE BASED ON THE ARCHITECTURE EXPANSION

Yan Gu, Yiqiang Wang, Jieqiong Lin, Xiuhua Yuan

Preliminary communication

There are currently no appropriate methods to find CNC system software defects and eliminate hidden dangers. In order to improve CNC system reliability, the architecture expansion-based fault location method in CNC system software was proposed in this paper. The failure of CNC system software was 619 analysed, the expansion method of CNC system software architecture was proposed and the expansion component was established. The software data morphology information and running path were monitored and recorded, the failure pathway was obtained and a similar path set algorithm was adopted to generate the similar pathway set of the fault path. A least squares SVM-based suspicion model was established to determine the fault statement, eliminate faults and position the software fault in the level of the CNC system structure. Fault location experimentation was conducted in the multi-axis movement control card PCI-7344. The experiment's result shows that the method proposed avoided the repeated testing and debugging by programmers. Without being limited by artificial factors and levels, it is a reliable method of CNC system software fault location.

Keywords: *architecture expansion; CNC system; Least Square Support Vector Machine (LS-SVM); similar path set; software fault location*

Lokacija greške u softveru CNC sustava na osnovu širenja arhitekture

Prethodno priopćenje

Trenutno ne postoje odgovarajuće metode kojima bi se pronašla greška u softveru CNC sustava i otklonile skrivene opasnosti. U svrhu poboljšanja pouzdanosti CNC sustava, u radu je predložena metoda lokacije greške u softveru CNC sustava zasnovana na širenju arhitekture. Analizirana je greška u softveru CNC sustava, predložena je metoda širenja arhitekture softvera CNC sustava i ustanovljena je komponenta širenja. Pratili su se i bilježili izvršna putanja i informacije o morfologiji podataka softvera, dobivena je putanja greške i prihvaćen algoritam slične putanje kako bi se generirala putanja slična putanji greške. Postavljen je model zasnovan na potpori vektora najmanjim kvadratima (Least Square Support Vector Machine - LS-SVM) kako bi se odredila naredba za grešku, eliminirale greške i greška softvera stavila u strukturu CNC sustava. Eksperimentiranje s lokacijom greške provedeno je u kartici za nadzor višeosnog gibanja PCI-7344. Rezultat eksperimenta pokazuje da se predloženom metodom izbjeglo ponovljeno testiranje i otklanjanje grešaka od strane programera. Neograničena umjetnim faktorima i nivoima, to je pouzdana metoda za lociranje greške u softveru CNC sustava.

Ključne riječi: *CNC sustav; lokacija greške u softveru; potpora vektora najmanjim kvadratima (LS-SVM); slična putanja; širenje arhitekture*

1 Introduction

As the function of CNC systems becomes increasingly more powerful and the software size expands continuously it is hard to guarantee the reliability of the software. The software of CNC systems is required to carry out a large quantity of real-time online calculation therefore, ensuring stable operation is the primary focus to improve working accuracy and efficiency. The software failure of a CNC system is mainly caused by design error. In the process of software design, some defects are not discovered in the testing stage. In processing, those defects are activated which result in occurrence of failure and subsequently can seriously influence production. Through research on CNC system software fault location techniques, it is possible to discover software defects, eliminate hidden dangers, and ensure stable operation.

The CNC system fault analysis and fault location technique work to good effect in terms of solving hardware fault [1÷3] although it cannot be easily applied in CNC system software fault finding. Currently, most programmers and users discover the fault via debugging the program, which is quite an expensive and time consuming process. Due to the influence and limitation of human's ability and many factors in the debugging process, it is very hard to realize quick and accurate fault location of CNC system software.

Zhou discovered that function and memory correspond to each other and he monitored data access memory to complete the diagnosis and fault location [4]. Harrold created software programs running track spectra and obtained the difference between the system software running track [5]. Renieris ran a program based on the path to locate the software failure by comparing the difference between failure and success paths [6]. Wang found the code fault location by the extent of the path shape to the software [7]. Other methods have also been proposed to solve the software fault location [8÷13]. The fault location methods above require a large quantity of testing data and a regular result can be obtained only when the capacity of samples reaches certain degree.

The focus of previous work on software fault location of CNC system was solved by using similar path sets and artificial neural networks [14]. These methods do not combine the operation behaviour of software with the overall software architecture to improve the overall level of reliability from a systematic perspective. This paper locates CNC system software faults through architecture expansion and begins from a systematic level to improve fault location of CNC software. The expansion component of CNC system software architecture is established, the software data morphology information and running path are monitored and recorded through the expansion component.

The remainder of the paper is organized as follows: Section 2 established CNC system software fault location methods on the basis of software architecture expansion; Section 3 set up the software architecture expansion of CNC system; Section 4 presented the similar path method to generate the similar pathway of the fault path and; Section 5 introduced the LS-SVM algorithm to calculate

code degree-of-suspicion. Finally, an application example was given in Section 6 to verify the method proposed in this paper.

The system structure expansion based on the overall structure characteristics of CNC system software and the fault positioning method avoid the repeated tests and debugging of programmers. Without being limited by artificial factors and levels, it is a reliable CNC system software fault location method.

2 Software fault location method based on architecture expansion

The reliability of CNC system software is different from the reliability of the hardware, in essence: a hardware fault or failure is caused due to physical abrasion, and its reliability reduces with the increase of system operation time. However, a software fault is caused due to a design defect. In the case of software, initially the program defect is caused due to designer error or other objective reasons. The software fault just refers to the state under which the incorrect, abnormal, or non-standard execution happens in software. If the software fault is not timely discovered, a serious accident will occur. When CNC system software operates to the defect path, the defect will be activated and then the fault will happen. Therefore, this paper starts from software architecture to research CNC system software fault location techniques through analysis on program operating path.

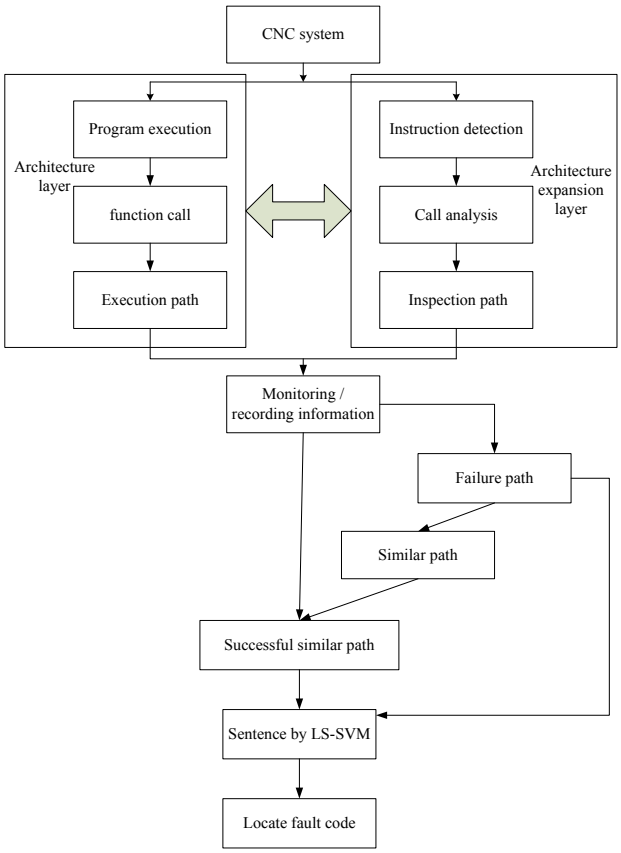

Figure 1 Fault location of CNC system software

The CNC system software architecture expansion method proposed is to carry out software fault location from the structure level of a CNC system. Firstly, this paper designs expansion components of CNC system software architecture. In the operation of a CNC system, the interaction is made between CNC system components and expansion components through semantic load and semantic perception to generate a monitoring task list and monitoring instruction list. The data form and operating path of program is also detected and recorded. Then, the similar path set of the failure path is generated according to a similar path algorithm. Through comparison, it is possible to obtain a feasible path with a concentrated similar path and construct a successful similar path. The next stage is to construct a code degree-of-suspicion calculation model based on LS-SVM to locate fault code. Finally, this paper provides a method for eliminating the fault.

This paper establishes a CNC system software fault location method based on architecture expansion, as shown in Fig. 1.

3 CNC system software architecture expansion

Through constructing the expansion of CNC system software architecture, the relevant components and algorithm were implanted into the system so as to carry out CNC software fault location and improve reliability of CNC systems. According to the software architecture of CNC systems, the expansion was made beyond the overall structure. The relevant functional components of fault location were added to obtain data form of the code and the operating path of the program by monitoring and recording operation information of the relevant program. The software architecture expansion of CNC system proposed in this paper is shown in Fig. 2.

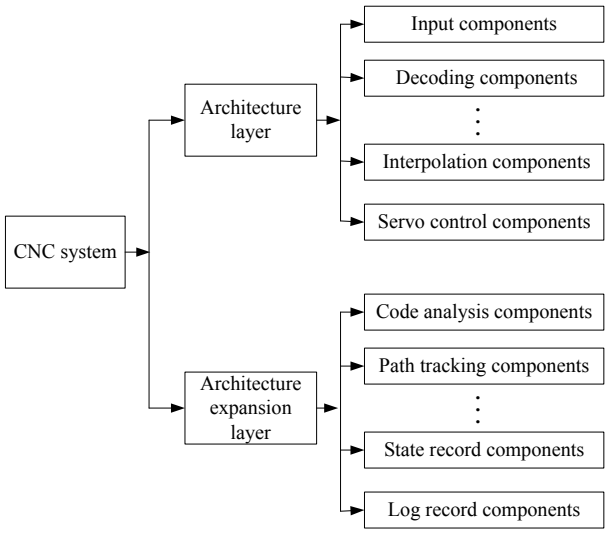

Figure 2 CNC system software architecture expansion

According to the method of CNC system software architecture expansion proposed in this paper, we added monitoring and recording components to obtain related data and operation information of the path. This data included excavation analysis on tracking the moving trajectory of the CNC system software, analysis of source code and the state skipping record of the code and other functions. From an operation process of CNC system

software perspective, the dynamic behaviour of software mainly consisted of data forms and instructions, and it referred to the relevance and interactive function between the operating path of the software program and the data. The CNC system software architecture expansion established in this paper is shown below:

(1) Data form analysis: describe the state of dynamic change in data form of the software, and carry out analysis from the data, including data structure layout, storage arrangement, and change of data state.

(2) Instruction analysis: monitor and analyse the operating path of program and track and record the operating path of program (including sequence, cycle, skip, and other basic structure). When the operating path passes through defect code of software, the corresponding logics, cycle, call or other errors will happen. The software defect is then activated and the software fault is caused.

(3) Relevance and interaction: carry out relevance and interaction between data and instruction, and then carry out mutual constraints between the operating path of software and data consumption, data utilization from instruction to data, or from instruction to instruction.

(4) Semantic load: due to expansion of relevant functional components in the original CNC system software architecture, the interrupting registration needs to be added if the system has access to newly added functional components. The software interrupting signal is reserved in the interrupting subsystem of the existing CNC system, and the semantic load is made in those reserved positions so as to realize the interaction between the software system result and the expanded functional components.

(5) Semantic perception: the relevant functional components expanded on architecture need to perceive the functional components of the original CNC system for convenience of monitoring and recording. Through adding monitoring on software instruction and execution tasks, it is possible to perceive the semantic information operated in program. The structure of functional components of the CNC system architecture expansion fault location is shown in Fig. 3.

Code input: program executed by CNC system software.

Instruction input: operating path of CNC system software, including sequence, cycle, skip, and other basic structure.

Monitoring task list: including data structure layout, storage arrangement, and change of data state, etc.; interaction and matching made between CNC system components and expansion components via semantic load and semantic perception to generate monitoring task list.

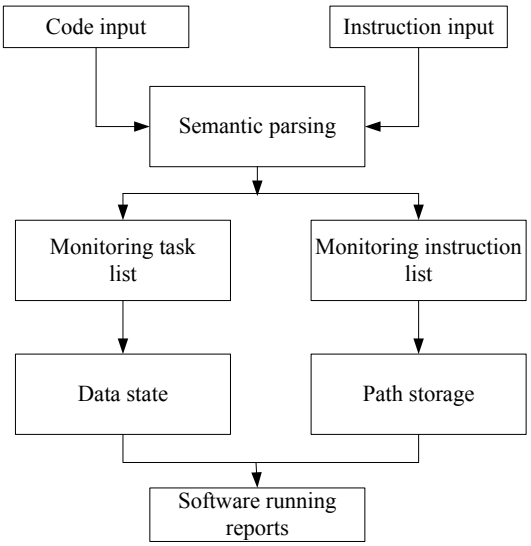

Figure 3 Structure of functional component of CNC system fault location

Monitoring instruction list: the software has a giant operating path, and it is unnecessary to record all operating instructions, only the call of path, skip, and information interruption need to be recorded.

4 Generation of similar path set of CNC system

Firstly it was required to generate the software path according to recorded information about the operation of architecture expansion components and then generate a similar path set of the failure path according to the similar path algorithm; by contrast, we can obtain a feasible path in the similar path set and construct a successful similar path.

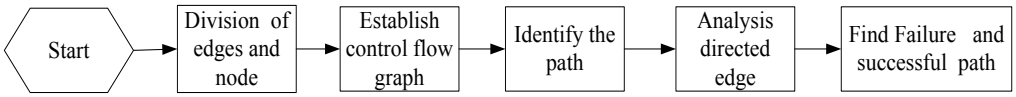

Figure 4 Flow chart of control flow method

As the software scale continuously expanded and the layer became more complex, the systematic concurrency was continuously enhanced. The research and analysis on software code can enhance our knowledge of systematic function, and also lay a firm foundation for successive analysis on software faults. Due to the fact that the code of CNC systems is complicated and difficult to understand, it was very difficult to realize timeliness by use of manual analysis. Based on the features and processes of CNC systems, we adopted a control flow method to analyse the software program, as shown in Fig. 4.

Through the analysis on skip node between failure path and execution process, the path space was calculated out. Through the comparison between space and node and replacement, we obtained the similar path. Then, we needed to look up the information obtained from architecture expansion components, obtain a feasible path in the similar path set by contrast, and then construct a successful similar path.

The realization process is shown below:

(1) Generate control flow diagram according to software code, and obtain failure path according to monitoring information on architecture expansion components

(2) Find out branch predicate

(3) Determine and mark the unrestraint side

(4) Calculate the distance between the failure path and similar path

(5) Add all obtained similar paths into the similar path set

(6) Make comparison between path information of the similar path set and record the architecture expansion component monitoring information to obtain successful similar path set.

5 Code degree-of-suspicion model based on LS-SVM

The mapping from the operating path of CNC system software to fault was used as the calculation mode of code degree-of-suspicion, the similar path and defect path were taken as input and the fault was taken as output. The solution was then calculated by use of LS-SVM. LS-SVM is evolved from SVM and the solution was made mainly from original space mapping to higher space according to the mapping relation between input and output [19]. The formula is shown below:

$$\mathrm{y}(\mathrm{x})=\omega\cdot\varphi(x)+b, \tag{1}$$

where: $(x_k,y_k), x_k\in R^n, y_k\in R^n, k=1,2,\cdots,N$, $\omega\in R^n$ refers to the weight vector, $b\in R$ refers to the deviation value, and $\varphi(x)$ refers to nonlinear mapping.

The solution was made according to formula above, as shown below:

$$J(\omega,R_{emp})=\frac{1}{2}\|\omega\|^2+\gamma\cdot R_{emp}, \tag{2}$$

where $\|\omega\|^2$ refers to the degree of complexity, γ refers to the error penalty function, and R_{emp} refers to the loss function.

There was a difference between selecting R_{emp} and traditional SVM as for LS-SVM, and the secondary norm of error e was selected.

$$\min_{\omega,\mathrm{b},e} J(\omega,e)=\frac{1}{2}\omega^T\omega+\gamma\frac{1}{2}\sum_{k=1}^{l}e_k^2, \tag{3}$$

where the constraint condition is: $y_k=\omega^T\varphi(x_k)+b+e_k$, $k=1,\cdots,N$.

Introducing the Lagrange function into formula (3):

$$L(\omega,b,e,\alpha)=\frac{1}{2}\omega^{\mathrm{T}}\omega+\gamma\frac{1}{2}\sum_{k=1}^{N}e_k^2-\sum_{k=1}^{N}\alpha_k\left(\omega^{\mathrm{T}}\varphi(x_k)+b+e_k-y_k\right) \tag{4}$$

where: e_k refers to the error corresponding to the k item, and α_k refers to the multiplier of Lagrange $(\alpha_k)\in R$.

As for formula above, the following formula was obtained via $\partial L/\partial\omega=0$, $\partial L/\partial b=0$, $\partial L/\partial\alpha_k=0$.

$$\left.\begin{aligned}
&\frac{\partial L}{\partial\omega}=0\rightarrow\omega=\sum_{k=1}^{N}\alpha_k\varphi(x_k)\\
&\frac{\partial L}{\partial b}=0\rightarrow\sum_{k=1}^{N}\alpha_k=0\\
&\frac{\partial L}{\partial e_k}=0\rightarrow\alpha_k=\gamma e_k\\
&\frac{\partial L}{\partial\alpha_k}=0\rightarrow y_k=\omega^{\mathrm{T}}\varphi(x_k)+b+e_k-y_k
\end{aligned}\right\} \tag{5}$$

The following formula was obtained via eliminating ω and e in the formula above:

$$\begin{pmatrix}0 & U^{\mathrm{T}}\\ U & \Omega+\gamma^{-1}I\end{pmatrix}\cdot\begin{pmatrix}b\\ a\end{pmatrix}=\begin{pmatrix}0\\ y\end{pmatrix} \tag{6}$$

where: I refers to unit matrix, $y=[y_1,y_2,\cdots,y_N]^{\mathrm{T}}$, $\alpha=[\alpha_1,\alpha_2,\cdots,\alpha_N]^{\mathrm{T}}$, $U=[1,1,\cdots,1]^{\mathrm{T}}$, $k=1,\cdots,N$, $\Omega=\varphi(x)\varphi(x)$ .There exists a kernel function $K(x_k,x_i)$:

$$K(x_k,x_i)=\varphi^{\mathrm{T}}(x_k)\varphi(x_i) \tag{7}$$

where: k = 1,..., N. Thus the regression model of LS-SVM is:

$$y(x)=\sum_{k=1}^{N}\alpha_k K(x_k,x_i)+b, \tag{8}$$

According to the similar path method proposed in the above section, the similar path of the failure path was generated. Then, we made conversion for the operating path of the CNC system program, by corresponding the fault position of the failure path and the similar path of the code to the input and output of LS-SVM according to actual operation situation. The sample of LS-SVM was (a^i,b^i), and a^i and b^i corresponded to input and output of LS-SVM respectively.

$$(a^i)_j=\begin{cases}1, & pass\ by\ edge\\ 0, & else\end{cases} \tag{9}$$

$$b^i=\begin{cases}0, & pass\ to\ success\\ 1, & pass\ to\ failure\end{cases} \tag{10}$$

where: $0<i<m+1$, $0<j<n$.

6 Examples of CNC system software fault location

This paper carried out fault location experiments on a programmable multi-axis controller PCI-7344 of NI. Firstly, we injected fault code into a NURBS interpolating program, and then generated a monitoring task list and monitoring instruction list according to the CNC system architecture expansion components designed in this paper in order to operate the interpolating program. We checked and recorded the data form of the program code and the

operating path of program, and then generated similar path set of the failure path according to the similar path algorithm. By contrast, we obtained a feasible path in the similar path set, and constructed a successful similar path. Finally, we calculated the degree of suspicion and located the fault code.

The purpose of the fault injection was to inject a fault into a specific operating program by hand so as to accelerate the occurrence of an error in the system and cause the program to fail. We adopted the CNC system architecture expansion components established in this paper to monitor the response after the fault was injected into system, and also recorded the information to provide an effective basis for the following calculation and analysis. The CNC system software fault injection is an effective means used to locate faults, accelerate the occurrence of faults and confirm the methodology. Through injecting error code into the NURBS interpolating program, this paper realized fault injection. The fault injection process of the interpolating program designed in this paper is discussed as below. A k NURBS curve can be defined as:

$$p(u)=\frac{\sum_{i=0}^{n} w_i d_i N_{i,k}(u)}{\sum_{i=0}^{n} w_i d_i N_{i,k}(u)}, \quad (11)$$

where: k_i refers to the control vertex, $(i=0,1,\ldots,n)$ refers to the weight factor, $N_{i,k}(u)$ refers to the primary function of k B spline, and $U=(u_0,u_i,\ldots,u_{n+k+1})$ refers to the node vector.

$$N_{i,0}(u)=\begin{cases}1 & u\in[u_i,u_{i+1}]\\ 0 & else\end{cases} \quad (12)$$

$$N_{i,0}(u)=\frac{(u-u_i)}{(u_{i+k+1}-u)}N_{i,k-1}(u)+\frac{(u_{i+k}-u)}{(u_{i+k}-u_{i+1})}N_{i+1,k-1}(u), \quad (13)$$

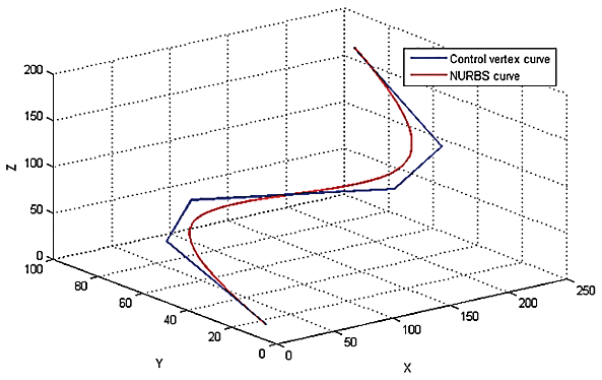

Figure 5 ANURBS interpolating curve

We took 3 parameters: control vertex d_i, weight factor w_i, and node vector U, as one part of the CNC program instruction.The NURBS curve is then formed within the CNC system. This paper adopted a cubic NURBS curve, and the set parameters are shown below:

Control vertex d_i:{(0, 0, 0), (−100, −100, 0), (−100, 100, 0), (0, 0, 0), (100, −100, 0), (100, 100, 0), (0, 0, 0) }. Weight factor w_i : {1, 1, 1.2, 0.88, 1, 0.9, 1 }. Node vector $\boldsymbol{u}_j$: {0, 0, 0, 0, 0.32, 0.64, 1, 1, 1, 1, 1}. Interpolation cycle: T_S = 1 ms. Feed rate: V = 50 mm/s. The maximum feed acceleration: a_{max} = 200 mm/s^2. The maximum chord error: ε_{max} = 3 μm. Acceleration: A = 500 mm/s^2. The NURBS curve is shown in Fig. 5, and the curve of interpolating speed is shown in Fig. 6.

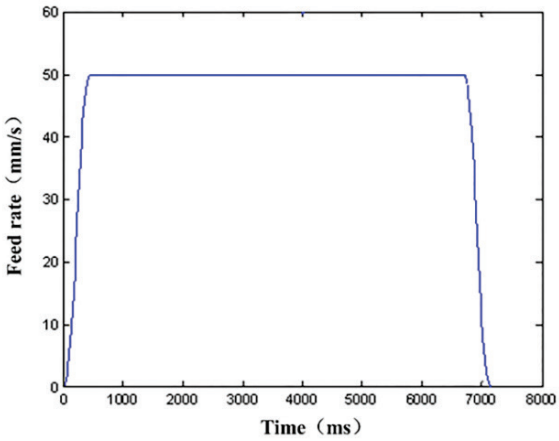

Figure 6 The curve of interpolating speed

When the fault code was injected and the interpolating NURBS operated, the fault occured in the CNC system. The fault was reflected by the fact that the obtained error in height of bow ε was larger than the setting value, as shown in Fig. 7.

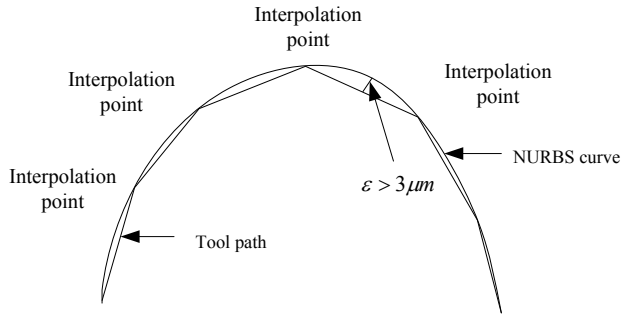

Figure 7 Schematic diagram of interpolation fault

To analyse the code, we firstly divided the program into the directed edge. The relationship between code, directed edge and node is shown in Tab. 1. The control flow diagram was formed next, as shown in Fig. 8.

Table 1 Division of directed edges

Code	1-4	5	6-8	9-13
Element	e_0	n_1	e_1	e_2
Code	14-18	19-21	22	23
Element	e_3	e_4	n_2	e_5
Code	24	25-26	27-29	30-31
Element	n_3	e_7	e_8	e_6
Code	32	33-34	35-37	38-39
Element	n_4	e_9	e_{10}	e_{11}
Code	40	41-42	43	
Element	n_5	e_{12}	e_{14}	

According to the monitoring components and recording components of the CNC system software

architecture expansion, the failure path was $\pi_f = e_0e_1e_4e_5e_7e_{11}e_{12}e_{14}$ when the fault occurred. $\{n_1, n_2, n_3, n_4, n_5\}$ refers to the predicate node of the path branch. The similar path generated according to the similar path algorithm is shown in Tab. 2. By contrasting with the information of the operating path, we analyzed the execution result of the similar path and obtained a successful similar path set $\{\pi_1\ \pi_3\ \pi_7\ \pi_9\ \pi_{11}\ \pi_{14}\ \pi_{17}\ \pi_{18}\ \pi_{20}\ \pi_{23}\}$.

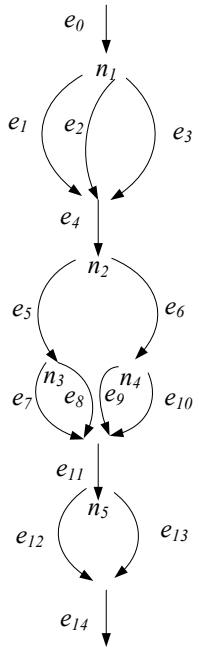

Figure 8 NURBS control flow diagram

After obtaining a failure path and similar path set, this paper adopted code degree-of-suspicion, calculated according to LS-SVM to determine fault sentence. The directed edge of the failure path and similar path was taken as the input of LS-SVM, and the path execution result obtained via architecture expansion was taken as the output. It can be deduced by the interpolating control flow diagram analyzed above that the directed edge of the failure path and similar path consists of 8 paths. Therefore, this paper established an 8 input and 1 output system. Through calculation, the parameters of the SVM model were obtained. Following this, the directed edge of the CNC system software fault code was taken as the input to calculate the degree of suspicion. The result is shown in Tab. 3.

The degree of suspicion of all directed edges can be seen from Tab. 3, the highest degree of suspicion is 0.8235, and the corresponding directed edge is e_7. According to the corresponding relation between code and directed edge, it can be known that the source code corresponding to directed edge e_7 is sentence 25 and sentence 26. Through checking sentence, we can know that the fault sentence is code 25, and this sentence is consistent with initially injected fault. To sum up, the CNC system software fault location method based on architecture expansion proposed in this paper can be used to successfully find out fault code in CNC system software.

Table 2 The similar path

Node	Similar path
$\langle n_1 \rangle$	$\pi_1 = e_0e_2e_4e_5e_7e_{11}e_{12}e_{14}$
$\langle n_1, n_5 \rangle$	$\pi_2 = e_0e_2e_4e_5e_7e_{11}e_{13}e_{14}$
$\langle n_1, n_3, n_5 \rangle$	$\pi_3 = e_0e_2e_4e_5e_8e_{11}e_{12}e_{14}$
$\langle n_1, n_3, n_5 \rangle$	$\pi_4 = e_0e_2e_4e_5e_8e_{11}e_{13}e_{14}$
$\langle n_1, n_2 \rangle$	$\pi_5 = e_0e_2e_4e_6e_9e_{11}e_{12}e_{14}$
$\langle n_1, n_3, n_5 \rangle$	$\pi_6 = e_0e_2e_4e_6e_9e_{11}e_{13}e_{14}$
$\langle n_1, n_2, n_4, n_5 \rangle$	$\pi_7 = e_0e_2e_4e_6e_{10}e_{11}e_{12}e_{14}$
$\langle n_1, n_2, n_4, n_5 \rangle$	$\pi_8 = e_0e_2e_4e_6e_{10}e_{11}e_{13}e_{14}$
$\langle n_1, n_2 \rangle$	$\pi_9 = e_0e_3e_4e_5e_7e_{11}e_{12}e_{14}$
$\langle n_1, n_5 \rangle$	$\pi_{10} = e_0e_3e_4e_5e_7e_{11}e_{13}e_{14}$
$\langle n_1, n_5 \rangle$	$\pi_{11} = e_0e_3e_4e_5e_8e_{11}e_{12}e_{14}$
$\langle n_1, n_3, n_5 \rangle$	$\pi_{12} = e_0e_3e_4e_5e_8e_{11}e_{13}e_{14}$
$\langle n_1, n_2 \rangle$	$\pi_{13} = e_0e_3e_4e_6e_9e_{11}e_{12}e_{14}$
$\langle n_1, n_2, n_5 \rangle$	$\pi_{14} = e_0e_3e_4e_6e_{10}e_{11}e_{12}e_{14}$
$\langle n_1, n_2, n_4, n_5 \rangle$	$\pi_{15} = e_0e_3e_4e_6e_{10}e_{11}e_{13}e_{14}$
$\langle n_1, n_2, n_4, n_5 \rangle$	$\pi_{16} = e_0e_3e_4e_6e_9e_{11}e_{13}e_{14}$
$\langle n_1 \rangle$	$\pi_{17} = e_0e_1e_4e_5e_7e_{11}e_{13}e_{14}$
$\langle n_3 \rangle$	$\pi_{18} = e_0e_1e_4e_5e_8e_{11}e_{12}e_{14}$
$\langle n_3, n_5 \rangle$	$\pi_{19} = e_0e_1e_4e_5e_8e_{11}e_{13}e_{14}$
$\langle n_1, n_2 \rangle$	$\pi_{20} = e_0e_1e_4e_6e_9e_{11}e_{12}e_{14}$
$\langle n_1, n_2, n_5 \rangle$	$\pi_{21} = e_0e_1e_4e_6e_9e_{11}e_{13}e_{14}$
$\langle n_1, n_2, n_4, n_5 \rangle$	$\pi_{22} = e_0e_1e_4e_6e_{10}e_{11}e_{13}e_{14}$
$\langle n_1, n_2, n_4, n_5 \rangle$	$\pi_{23} = e_0e_1e_4e_6e_{10}e_{11}e_{12}e_{14}$

Table 3 The degree of suspicion of all directed edges

Directed edges	Degree of suspicion
e_0	0,0819
e_1	0,2031
e_4	0,3258
e_5	0,5069
e_7	0,8235
e_{11}	0,1219
e_{12}	0,2825
e_{14}	0,0819

7 Conclusion

This paper analysed the fault mechanism of CNC system software, proposed a CNC system software architecture expansion method, and established a similar path set algorithm and degree-of-suspicion calculation model based on LS-SVM to carry out fault location in software from the CNC system structure level. Firstly, this paper analysed software fault mechanism of CNC system, and found that the software fault was caused due to the fact that the executive path of the program activates a defect in the software, and thus the CNC system fails. This paper then proposed a CNC system software architecture expansion method, and designed CNC system software architecture expansion components. Through semantic load and semantic perception, the interaction between CNC system components and expansion components was made to generate a monitoring task list, monitoring instruction list. The data form of program code and the operating path of the program were detected

and recorded. Next, this paper proposed a similar path algorithm to generate a similar path set of an operating failure set; by contrast, we obtained a feasible path in the similar path set, and constructed a successful similar path. Furthermore, this paper established a code degree-of-suspicion calculation model based on LS-SVM, and carried out fault location experiments on a programmable multi-axis controller PCI-7344, successfully locating fault code. The experimental result showed that the method of architecture expansion based on the overall structure of CNC system software and fault location through similar path and LS-SVM avoids the need for repeated testing and debugging by programmers. It is not limited by human factors and knowledge, and it is a feasible CNC system software fault location method.

8 References

[1] Yiqiang, Wang; Yazhou, Jia; Weiwei, Jiang. Early failure analysis of machining center: a case study. // Reliablity Engineering & System Safety. 72, 1(2001), pp. 91-97. DOI: 10.1016/S0951-8320(00)00100-9

[2] Yiqiang, Wang; Yazhou, Jia;Junyi, Yu;Shangfeng, Yi. Field failure database of CNC lathes. //International Journal of Quality & Reliability Management. 16, 4(1999), pp. 330-343. DOI: 10.1108/02656719910266532

[3] Yiqiang, Wang; Xiaoqiang, Wang; Shunming, Hua. Reliability analysis and improvement of ATCs of CNC lathes. // Applied Mechanics and Materials, 2010, pp. 939-943.

[4] Pin, Zhou; et al. iWatcher:Efficient architectural support for software debugging. // Proceedings of the 31st annual international symposium on Computer architecture. 32, 2(2004), pp. 224-235.

[5] Harrold, M. J.; Rothermel, G.; Sayre, K. et al. An empirical investigation of the relationship between spectra differences and regression faults. // Software Testing Verification and Reliability. 10, 3(2000), pp. 171 -194. DOI: 10.1002/1099-1689(200009)10:3<171::AID-STVR209>3.0.CO;2-J

[6] Renieris M.; Reiss, S. P. Fault localization with nearest neighbor queries. // International Conference on Automated Software Engineering, 2003, pp. 30 -39. DOI: 10.1109/ase.2003.1240292

[7] Liang, Guo; Roychoudhury, A.; Tao, Wang. Accurately choosing execution runs for software fault localization. // International Conference on Theory and Practice of Software, 2006, pp. 80-95.

[8] Wong, W. E.; Debroy, V.; Gao, R. Z.; Li, Y. H. The DStar Method for Effective Software Fault Localization. // IEEE Transactions on Reliability. 36, 1(2014), pp.290-380. DOI: 10.1109/TR.2013.2285319

[9] Kim, D.; Tao, Y.; Kim, S.; Zeller, A. Where Should We Fix This Bug? A Two-Phase Recommendation Model. // IEEE Transactions on Software Engineering. 39, 11(2013), pp. 1597-1610. DOI: 10.1109/TSE.2013.24

[10] Sahoo, S. K.; Criswell, J.; Geigle, C.; Adve, V. Using Likely Invariants for Automated Software Fault Localization. // ACMSIGPLAN Notices, 48, 4(2013), pp. 139-151. DOI: 10.1145/2499368.2451131

[11] Natella, R.; Cotroneo, D.; Duraes, J. A.; Madeira, H. S. On Fault Representativeness of Software Fault Injection. // IEEE Transactions on Software Engineering. 39, 1(2013), pp. 80-96. DOI: 10.1109/TSE.2011.124

[12] Wong, W. E.; Debroy, V.; Golden, R.; Xiaofeng, Xu; Thuraisingham, B. Effective Software Fault Localization Using an RBF Neural Network. // IEEE Transactions on Reliability. 61, 1(2012), pp. 149-169. DOI: 10.1109/TR.2011.2172031

[13] Mehrkanoon, S.; Suykens, J. A. K. LS-SVM approximate solution to linear time varying descriptor systems. // Automatica. 48, 10(2012), pp. 2502-2511. DOI: 10.1016/j.automatica.2012.06.095

[14] Xiuhua, Yuan; Yiqiang, Wang; Yan, Gu. Software fault location of CNC system based on similar path set and artificial neural network. // Advances in Mechanical Engineering. 5, (2013), pp. 357308-1–357308-9.

Authors' addresses

Yan Gu
School of Electro-mechanical Engineering,
Changchun University of Technology,
Changchun, 130012, Jilin, China
pcik001@qq.com

Yiqiang Wang
Corresponding author
Ningbo Institute of Technology,
Zhejiang University,
Ningbo, Zhejiang 315000, China
jluwang@gmail.com

Jieqiong Lin
School of Electro-mechanical Engineering,
Changchun University of Technology,
Changchun, 130012, Jilin, China
linjieqiong@mail.ccut.edu.cn

XiuhuaYuan
School of Mechanical and Automotive Engineering,
Liaocheng University,
Liaocheng, Shandong, 252000, China
yxhjlu@163.com