Construction of Software Reliability Assessment Index System Based on ICMM

Shizhuang Yin[1],Yulu Shi[2],Zhifeng You[1] ,Quan Shi[1],Shihan Tan[1]

Shi Yulu :1329760604@qq.com ,Yin shizhuang: yinshi@aeu.edu.cn ,
Shi quan :y18525742489@126.com ,Tan shihan:13106566155@163.com
Communication author: You Zhifeng:18525742489@163.com

[1]Department of Equipment Command and Management, Army Engineering University, Shijiazhuang 050003, China
[2]People's Liberation Army (PLA) Factory No. 3302, Shijiazhuang 050003, China

Abstract: To address the problem of imperfect index system and strong uncertainty of expert scoring in the process of software reliability assessment, the influencing factor set of software node reliability is formed respectively by consulting the literature and experts; the influencing factor set determined by the experts is initially screened by the improved quality house and improved software maturity algorithm, and the obviously unreasonable or repeated factors are removed; the weights of different experts are determined to establish the index system of software node reliability assessment more precisely. By determining the weights of different experts, the software node reliability assessment index system is established more accurately. The construction of software reliability index system can provide a theoretical basis for the improvement of node destruction resistance, and the accurate classification of node state can provide data support for the overall network destruction resistance.

Keywords: ICMM, Software Reliability, IAHP, Expert Weights

1 Introduction

With the transformation of war mode from "information" to "intelligence" under the condition of high technology, artificial intelligence and information technology are widely used in weapon system and automated information system, and the core of artificial intelligence and information technology is the software in weapon equipment[1] . The proportion of software in the realization of the functions of weapons and equipment is getting bigger and bigger, and some software is no longer an accessory to the weapons, but is gradually replacing the functions originally realised by the hardware. In order to better describe this type of equipment, Professor Gan Mao zhi introduced the concept of software-intensive equipment from abroad. "Software-intensive equipment[2] " refers to a type of equipment in which software is in a dominant position in the fields of equipment development costs, development time or equipment functional characteristics. The scale and complexity of this type of equipment will increase rapidly with the wide application of artificial intelligence and information technology, and it plays an important role in modern military struggle, changing the form and process of war[3] .

MSIEID 2023, December 08-10, Guangzhou, People's Republic of China

DOI 10.4108/eai.8-12-2023.2344738

SIS system is composed of software subsystems and hardware subsystems, while hardware equipment nodes and software functional nodes constitute the basis of the whole equipment network, and when the state of nodes changes, the performance of equipment will also be affected. Existing complex network destructive research will nodes simply differentiated into damage and intact two states, which is inconsistent with the actual situation, reducing the accuracy of equipment state assessment. This paper focuses on the establishment of the SIS software node state assessment index system. And combined with the characteristics of software nodes to establish the indicator system and classification model. The construction of the node indicator system and the analysis of influencing factors can provide a theoretical basis for the improvement of node destruction resistance, and the accurate classification of node status can provide data support for the overall destruction resistance of the network.

2 Software Maturity Model CMM

Maturity is a common measure of productivity and quality of software organizations[4] and is often used to assess the degree of development of an entity's actions, thus providing a scientific basis for decision-making. Accurate assessment of the maturity of software nodes enhances the defense capability of the network and improves the ability to carry tasks and operations in cyberspace[5].

The CMM model builds a standard framework for describing the maturity of a software organization’s capabilities, covering five software organization maturity levels from immature to mature. As shown in Figure 1, the entire evaluation of the CMM operates in sequential top-down steps: each level, except Level 1, contains multiple Key Process Areas (KPAs) for achieving the objectives of that level. Figure 1 shows the CMM hierarchy.

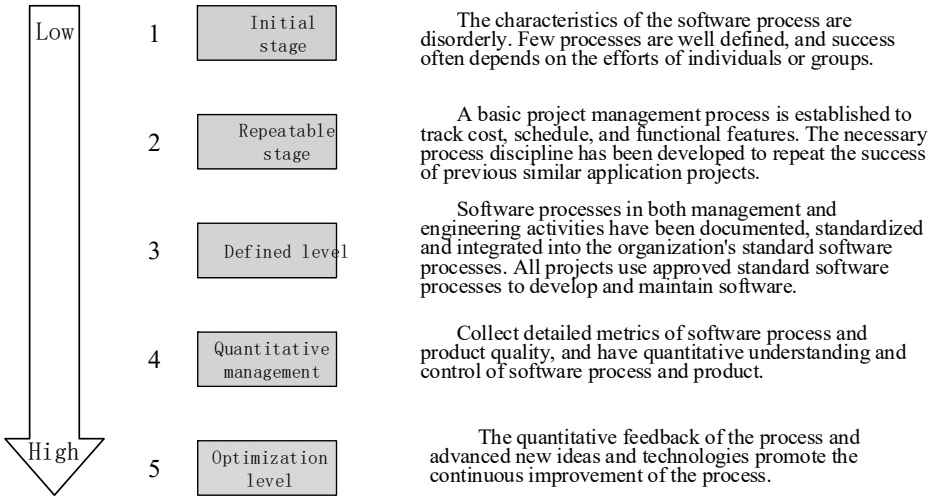

Figure 1 Schematic diagram of software maturity hierarchy based on CMM

Based on the characteristics of software-intensive equipment and the purpose of assessment, the following basic principles should be followed when constructing a system of software-intensive equipment destruction resistance indicators:

Purposefulness: The purpose of the assessment is clear and ensures that the work undertaken serves the purpose of the assessment. The purpose of the assessment could be to assess the level of resistance of software-intensive equipment or to assess differences in resistance between different software-intensive equipment.

Hierarchy: Software-intensive equipment destructiveness involves elements at multiple levels, including hardware, software and network. When establishing the indicator system, these elements should be divided into levels so that the structure of the indicator system is clear and it is easy to carry out a hierarchical assessment of anti-destructiveness to ensure that key information is not omitted.

Scientific: Through scientific analyses, ensure that the indicator system should be able to truly and accurately reflect the actual situation of software-intensive equipment resistance to destruction, and that the indicators should be as objective as possible.

Completeness: Considering all kinds of factors affecting the anti-destructiveness of software-intensive equipment, covering hardware, software, network and other aspects of the equipment, to ensure that the assessment indicator system can comprehensively and systematically reflect the construction level of anti-destructiveness of software-intensive equipment. The construction of a complete indicator system enables it to comprehensively assess the anti-destructiveness of software-intensive equipment from all sides.

Simplicity: Under the premise of ensuring the completeness of the indicator system, the main influencing factors are selected, important indicators are highlighted, and too many indicators are avoided in order to improve the efficiency of the assessment.

Sensitivity: The assessment indicators should be sensitive, i.e., able to respond sensitively to changes in software-intensive equipment resilience. The assessment indicators should be able to capture key factors and trends in equipment resilience so that problems can be identified in a timely manner and improvements can be made accordingly.

Operationalization: The assessment indicators should be able to provide concrete recommendations and measures to guide practical work. The assessment indicators should not only reflect the level of software-intensive equipment resilience, but also be able to provide practical operational guidance for improving and upgrading resilience.

3 Constructing the set of factors influencing software maturity

By analyzing the current status of research on software assessment methods such as CMM, it is found that most of the existing software assessment index systems are qualitative assessments, and in order to achieve quantitative evaluation, it is first necessary to identify the factors that affect software maturity. In this section, we refer to a survey report of 13 software development organizations in literature [6], and combine the concept of key domains of CMM to construct a set of factors influencing the maturity of software nodes, as shown in Table 1.

Table1 Set of factors affecting the maturity of software nodes

serial number	Level 1 indicators	Secondary indicators	factor
1	software maturity u_1	General Software Features	software scale
2			software category
3			Ratio of reused code
4			programming language
5		Management of the development process	development management
6			Documentation describes the frequency of change
7			Software Design Documentation
8			Phase Evaluation Criteria
9		Software Product Quality Assurance	Test Methods
10			Test Coverage
11			Test Tools
12			Test document
13		Software Engineering Practice	Design methodology
14			demand analysis
15			Detailed design
16			Proportion of highly qualified programmers
17		organizational capacity	Level of development effort
18			Level of development technology
19			work pressure
20			Size of the development workforce

The detailed description and definition of the factors influencing software maturity are as follows

(1) Software Size: The number of function points or amount of code in a software system. Quantitative methods can be used to measure the size of the software using the number of lines, function points or requirements-based function points.

(2) Software Category: Software Category refers to the classification of software according to its use or domain. For example, software can be divided into different categories such as sound processing software, graphic image processing software, and so on.

(3) Code Reuse Ratio: Code Reuse Ratio refers to the proportion of reused code used in the software development process. It can be calculated by the ratio of the number of lines of reused code to the total number of lines of code.

(4) Programming Language: A programming language is a formal language used to write computer programmers. The quantification method is based on the type and proportion of programming languages used.

(5) Development Management: Development management refers to the activities of organizing, planning, controlling and managing the software development process. The effect of development management can be quantified through project progress, resource allocation, risk management and other aspects.

(6) Specification Change Frequency: Specification change frequency refers to the frequency of modification of specification documents in the software development process. The frequency of change can be quantified by counting the number of versions of the specification file or the number of changes in each version.

(7) Software Design Document: The software design document is a detailed description of the design of the software system. The quality of the software design document can be quantified by assessing the completeness, clarity and compliance of the document.

(8) Phase Evaluation Criteria: Phase Evaluation Criteria are criteria used to assess the completion and quality of each phase of the software development process. Evaluation can be carried out by defining and quantifying the evaluation indicators and criteria for each phase.

(9) Testing Method: The methods and techniques used to verify that a software system meets the expected requirements. The effectiveness of the testing method can be quantified according to the testing strategy, test case design and execution process.

(10) Test Coverage: the degree of coverage of test cases in the software system. The ratio of executed test cases to the total number of test cases can be expressed.

(11) Testing Tool: A software tool that assists in testing activities. The degree of use and effectiveness of testing tools can be quantified according to the type and use of testing tools.

(12) Testing Documentation: Documents that record testing activities. Usually use the document is complete, accurate and timely and other characteristics to quantify the quality of test documents.

(13) Design Method: Methods and techniques used in the software development process for system design. The effectiveness of the method can be measured based on the type of design method and the degree of application.

(14) Requirement Analysis: The process of sorting out, analyzing and defining the requirements of a software system. It can be assessed by its consistency, completeness and accuracy.

(15) Detailed design: that is, on the basis of the system design of the software system module division, data structure design and other activities. The quality of detailed design can be quantified by assessing the completeness, clarity and compliance of the detailed design documents.

(16) Proportion of High-Quality Programmers: is the proportion of programmers with a high level of skills and experience on the team. The proportion of High-Quality Programmers can be quantified by counting the skill level and experience of team members.

(17) Development Effort: Development Effort is the amount of time, resources and effort invested in the software development process. Development Effort can be quantified by counting the working hours of the development team, project progress and resource consumption.

(18) Development Technical Level (DTL): DTL refers to the level of technology and methodology applied by the development team in the software development process. The

development technical level can be quantified by assessing the technical competence of the team members, the technical tools and frameworks used in the project, and so on.

(19) Work Pressure: Work Pressure refers to the work intensity and pressure faced by the development team in the software development process. Work pressure can be quantified through questionnaires, interviews, or by assessing the workload and pressure feelings of team members.

(20) Development Team Size: The size of the development team is the number of team members involved in software development. The size of the development team can be quantified by counting the number of people in the development team.

4 Initial screening of indicators

Initial screening of software node metrics was carried out using the AHP method.
The weights of R were calculated using the AHP method to determine the importance of objectivity R1, utility R2, measurability R3, and criticality R4.The scale of proportions is shown in Table 2.

Expertly assessed and obtained:

$$\boldsymbol{T}=\begin{bmatrix} 1 & a_{12} & \cdots & a_{1m} \\ a_{12} & 1 & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & 1 \end{bmatrix} \tag{1}$$

Table2 The scale 1 to 9 in AHP

Factor x is better than factor y	Quantized value
Equally important	1
Slightly more important	3
Obviously important	5
Strong importance	7
Extremely important	9
The median value of two adjacent judgments	2, 4,6,8

To test the consistency of the matrix T, the largest eigen root $\lambda_{\max}$ of the matrix T is calculated and then, the values of the consistency index CI and the test coefficient CR are calculated with the formulas respectively:

$$CI=\frac{\lambda_{\max}-n}{n-1},\quad CR=\frac{CI}{RI} \tag{2}$$

Where RI is the average random value of the consistency indicator, whose standard value is determined by the order n , and the value of RI is constant if n is constant. When $CR \le 0.1$ meets the consistency requirements. After normalization, the relative importance weights of each factor are calculated $\boldsymbol{W} = \left(\omega_1, \omega_2, \cdots, \omega_m\right)$.

Expert's matrix of relationships between the 4 screening requirements and the 20 influencing factors $\boldsymbol{R}_S$.

$$\boldsymbol{R}_S = \begin{bmatrix} 7 & 8 & 4 & 7 & 6 & 7 & 4 & 5 & 5 & 6 & 7 & 6 & 6 & 4 & 8 & 6 & 6 & 3 & 3 & 7 \\ 7 & 6 & 6 & 8 & 8 & 5 & 5 & 7 & 8 & 6 & 4 & 7 & 5 & 6 & 6 & 8 & 7 & 6 & 4 & 5 \\ 6 & 6 & 5 & 7 & 6 & 6 & 5 & 5 & 5 & 7 & 5 & 7 & 6 & 5 & 6 & 5 & 6 & 5 & 4 & 6 \\ 7 & 5 & 6 & 7 & 8 & 5 & 6 & 6 & 7 & 6 & 4 & 5 & 8 & 7 & 5 & 6 & 7 & 7 & 5 & 4 \end{bmatrix} \tag{3}$$

Eventually it can be calculated:

$$\begin{aligned} \boldsymbol{W}_3' = (&0.120, 0.107, 0.098, 0.132, 0.129, 0.097, 0.090, 0.108, 0.119, \\ &0.097, 0.070, 0.128, 0.091, 0.088, 0.093, 0.101, 0.102, 0.086, 0.063, 0.081) \end{aligned} \tag{4}$$

According to the index weight size sorting, and then select the corresponding proportion of indicators, software node rough selection of the indicator system shown in Figure 2.

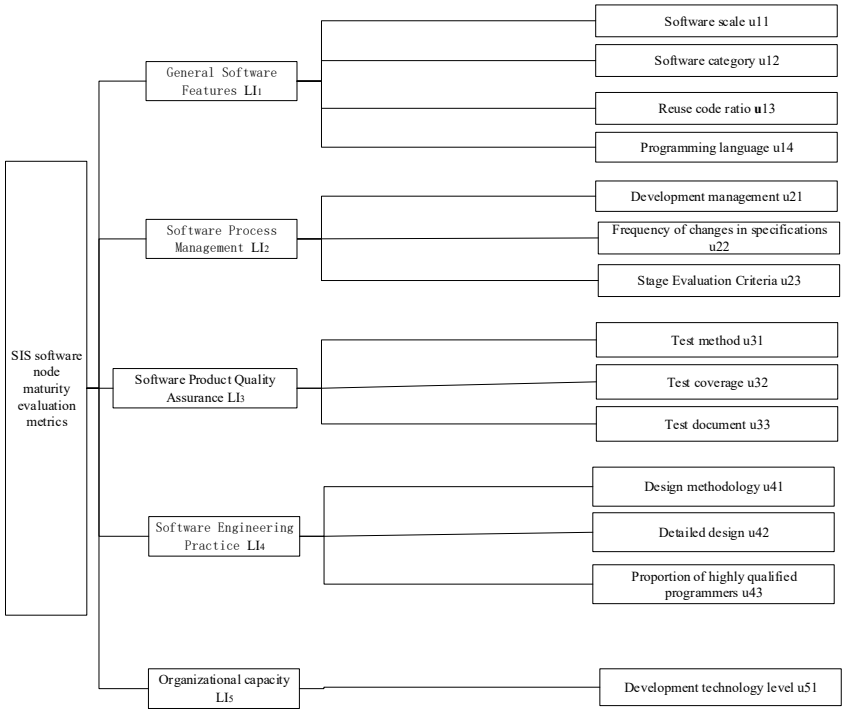

Figure2 Rough selection index system for software node state evaluation

5 Software nodes Fine screening of reliability metrics

Software node refers to the principle of relative independence and clarity of function, according to the maintenance requirements proposed by the user, the software system is divided into a number of subsystems that can be identified by the user, called software nodes. Software nodes can adopt different granularity according to the needs. In this paper, the nodes are subsystems composed of a number of components that carry certain functions. As the indexes of software node maturity assessment are more subjective compared to hardware nodes, they are prone to problems such as redundant noise irregularities. Therefore, the data of the software node is obtained by expert scoring, but the expert scoring process due to the different levels of experts, the direct use of the scoring table of experts is prone to cause some data distortion, in order to be able to effectively eliminate the error value that exists in the scoring of experts, the choice of the scoring data of the experts is sorted, to determine the weight of each expert in the scoring of the item, so as to make the scoring table obtained more accurate.

For the nth indicator, the scoring results of different experts on the indicator are counted separately, and the scoring results of the k expert on all indicators can be expressed as follows:

$$H_k = (h_{k1}, h_{k2}, \cdots h_{kn}) \tag{5}$$

The scoring results of all the experts were weighted to obtain an objective function of:

$$W = \sum_{k=1}^{m} \chi_k H_k \tag{6}$$

W is the objective function, i.e., the sum of scoring results. Substituting equation (5) into equation (6), there is:

$$W = (\sum_{k=1}^{m} \chi_k h_{k1}, \sum_{k=1}^{m} \chi_k h_{k2}, \cdots, \sum_{k=1}^{m} \chi_k h_{kn}) \tag{7}$$

h_{kn} Indicates the scoring result of the k expert on the n indicator, and the expert weight assessment model is shown in Figure 3, with the weights indicatingthe strength of reasonableness.

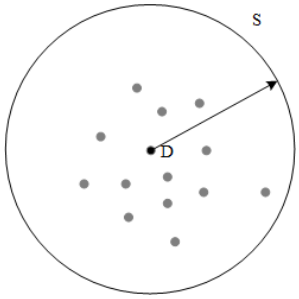

Figure3 Expert scoring weight evaluation model

Among them:

$$W<\sum_{k=1}^{m} h_k \tag{8}$$

$$\sum_{k=1}^{m} \chi_k = 1 \tag{9}$$

The absolute difference between the scoring results of the k expert and the i expert can be expressed as follows:

$$d_{ki} = \sqrt{\sum_{j=1}^{n} (\chi_k h_{kj} - \chi_i h_{ij})^2} \tag{10}$$

The smaller the absolute difference between the scoring results of the experts indicates a good consistency of opinion between the experts and a better rating, therefore, there is an optimization model as follows:

$$d = \min(\sum_{k=1}^{m} \sum_{i=1, i\neq k}^{m} d_{ki}^2) \tag{11}$$

After combining Eq. (8) and (9) as the constraint function and Eq. (11) as the objective function into the LaGrange function, there is the following functional relationship:

$$L(\chi, \lambda) = \sum_{k=1}^{m} \sum_{i=1, i\neq k}^{m} \sum_{j=1}^{n} ((\chi_k h_{kj} - \chi_i h_{ij})^2 - 2\lambda(\sum_{k=1}^{m} \chi_k - 1)) \tag{12}$$

Calculated by differentiating the above LaGrange function with respect to χ :

$$\frac{dL}{d\chi}=(m-1)\cdot(\sum_{j=1}^{n}h_{kj}^{2})\cdot\chi_k-\sum_{i=1,i\neq k}^{m}\sum_{j=1}^{n}(h_{kj}\cdot h_{ij})\cdot\chi_i-\lambda \tag{13}$$

Letting Eq. (13) be zero converts to a matrix P:

$$P=\begin{bmatrix}(m-1)\cdot(\sum_{i=1}^{n}h_{1i}^{2}) & -\sum_{i=1}^{n}(h_{1i}\cdot h_{2i}) & \cdots & -\sum_{i=1}^{n}(h_{1i}\cdot h_{mi})\\ -\sum_{i=1}^{n}(h_{2i}\cdot h_{1i}) & (m-1)\cdot(\sum_{i=1}^{n}h_{2i}^{2}) & \cdots & -\sum_{i=1}^{n}(h_{2i}\cdot h_{mi})\\ \vdots & \vdots & \ddots & \vdots\\ -\sum_{i=1}^{n}(h_{mi}\cdot h_{1i}) & -\sum_{i=1}^{n}(h_{mi}\cdot h_{2i}) & \cdots & (m-1)\cdot(\sum_{i=1}^{n}h_{mi}^{2})\end{bmatrix} \tag{14}$$

virtuous

$$\begin{cases}I=(1,1,\cdots,1)^T\\ \chi=(\chi_1,\chi_2,\cdots,\chi_m)^T\end{cases} \tag{15}$$

there are

$$P\chi-\lambda I=0 \tag{16}$$

Since $d_{ki}\geq 0$, P is a positive definite invertible matrix, and since the weight coefficients are not less than zero, it is obtained by combining Eq. (15) with the solution to Eq. (16):

$$\begin{cases}\lambda=\dfrac{1}{I^TP^{-1}I}\\ \chi=\dfrac{P^{-1}I}{I^TP^{-1}I}\end{cases} \tag{17}$$

Finally, the weights of the experts were compared, the scores of the experts with an error of more than 50% from the centroid data were removed, and the scores of the remaining experts were fused and summed according to the weights to obtain the final matrix.

Based on the experts' scoring values for each software subsystem to determine the specific values of the indicators for that software subsystem, and then use and expert weighting method to synthesize each expert's opinion, and finally obtain a scoring table containing the software subsystems, which includes the value of each indicator and the maturity level value of the software node.

After the above adjustments, the assessment indicators were collated and summarized, and the experts were repeatedly consulted, and the state assessment indicator system of software nodes and hardware nodes was finally determined, as shown in Figure 4.

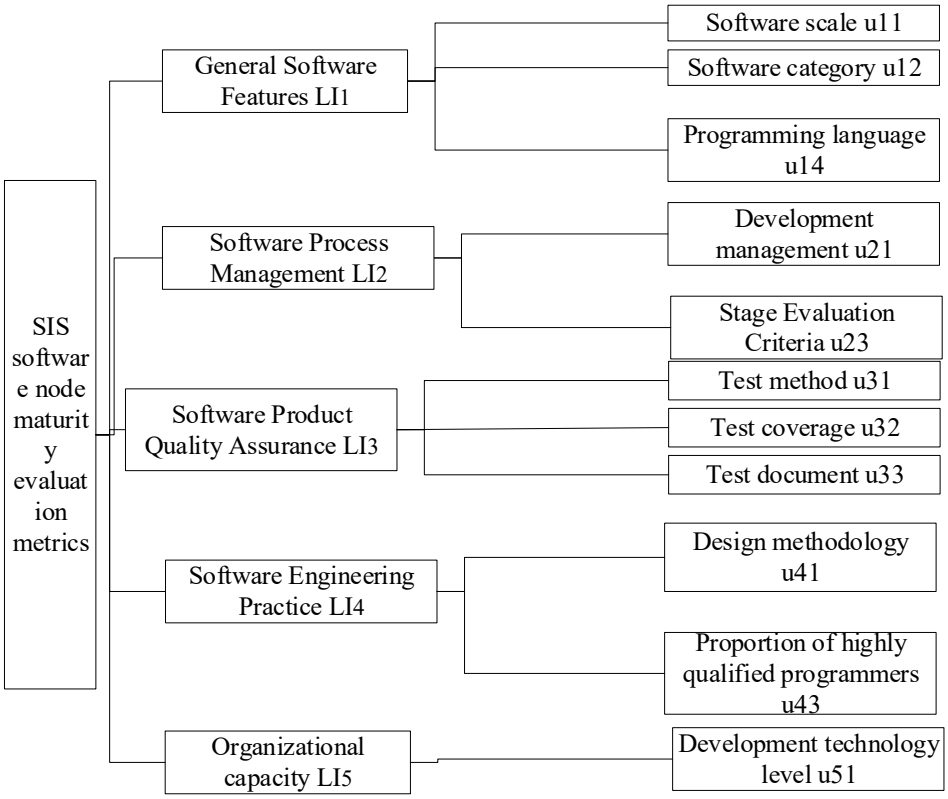

Figure 4 Final index system of software node

6 Conclusions

This paper establishes an index system for the state assessment of SIS software and hardware nodes. Analyzing the SIS software node influence factors, 20 software node influence factors are summarized by consulting the literature and expert consultation, etc. After that, the indicators are finely screened through the initial screening of the IQFD algorithm and the expert weighting method. It effectively solves the part of data distortion caused by subjective influence in the process of expert scoring, and effectively removes the error value existing in the expert scoring. Finally identified 14 software node indicators, for software nodes, the assessment indicator system can include maturity, reliability, security and other indicators. Maturity indicators can be used to assess the maturity of the development process, test coverage, defect repair and other aspects of the software node. By constructing the software and hardware assessment index system, the software nodes of software-intensive equipment can be comprehensively assessed and analyzed. This helps to understand the status and

performance of the nodes, identify potential problems and weaknesses, and take corresponding improvement and optimization measures to improve the overall performance and destruction resistance of the equipment.

References

[1] CHEN A Lei, LIU Zhen, ZHOU Chang et al. Research on the Development of Typical Intelligent Net Power Military Equipment in Foreign Countries[J]. Ship Electronic Countermeasures, 2023, 46(01): 14-21.

[2] Song Xuewen, Geng Huafang. Software-intensive equipment comprehensive security [M]. Beijing: National Defence Industry Press, 2011.

[3] LIU Mengyue , LI Ji'an,HUANG Maosheng. Research on software and hardware failure modes of software-intensive equipment[J]. Electronic Product Reliability and Environmental Testing,2017,35(04):42-48.

[4] CHEN Huiping,CHEN Jingyue. Static analysis of embedded software maturity based on minimum confidence[J]. Journal of Jilin University (Information Science Edition),2021,39(05):596-601.

[5] Zhang Xuan.The theory and practice of CMM software configuration management[J]. Communication World,2017(08):67-68.

[6] Xuemei Zhang, Hoang Pham. An Analysis of Factors Affecting Software Reliability. the Journal of System and Software, 2000.50: 43-56