Redundancy Mechanism of Software System and Reliability Analysis

Chunlin Yin1,a, Zhengyun Fang2,b, Na Zhao3,c

1Electric Power Research Institute Yunnan Power Grid co. ltd. Kunming 650217 China
2Yunnan Power Grid co. ltd. Kunming 650011 China
3College of Software, Yunnan University, Kunming 650091, China

a10878968225@qq.com, b413769194991@139.com, c43027029@qq.com

Abstract. In the design and development of software systems, redundancy mechanism is a technique that improves the reliability of a system. Therefore, the mechanism of redundant processing of core modules in a module-based software system was proposed in this study on the basis of the analysis of the structural redundancy of a software system and its impact on reliability. This mechanism includes a dual-module redundancy structure of a single module and a dual-module redundant structure of combined modules; fault detection and judgment method was also presented. Simultaneously, the reliability of the system is analyzed on the basis of various redundant structures.

1. Introduction

Under the promotion of related applications and backgrounds, the complexity and scale of software systems are constantly expanding. The reliability of software has become an important index for measuring a software system. However, as a core of computer technology, the production status and quality of software systems are occasionally unsatisfactory, and the failure of software systems in key areas may frequently cause serious and even disastrous consequences for humans and the environment.

In the field of software dependability research, Laprie proposed the concept of software dependability from studying security-critical systems [1] as early as 1991. In 1997, the National Science and Technology Council clearly proposed the concept of high confidence [2]. Literature [3] contended that high confidence software must be able to satisfy a series of key properties, such as security, real-time performance, reliability, fault tolerance, and confidentiality, when providing services in the system. Software fault tolerance is a technology that improves the software system’s reliability, and research on it can be divided into two main categories, namely, software redundancy and time redundancy [4-6]. Software redundancy refers to adding some modules when the system is designed in which the entire system still completes the required tasks even if a component fails. Time redundancy is based on the idea of retry-on-failure, that is, to roll back to the appropriate checkpoint for re-execution when a failure occurs [7]. At a compiler level, Reis et al. made a version of redundancy on software by copying and merging instructions to be able to check the consistency of instructions at the set synchronization point with improved fault tolerance and execution efficiency [6]. Through dynamic monitoring and rollback technology, an appropriate reduction point was established in Literature [8] to monitor the system; thus,
the system can be restored to the previously restored point in time. In Literature [9], a mixed fault-tolerant model of the redundant software of an evolutionary module based on evolutionary computing was proposed after analysing the effect of modularity on reliability. In Literature [10], a programmable controller redundancy system based on network control was designed. Literature [11] developed a secure computer system based on two out of three redundant structures. Reliable software, as a challenging and valuable research topic in the field of computer science, has attracted considerable attention from scholars at home and abroad.

In modern software engineering technology, modularization has become a general development trend of software technology [12], and the module-based software development technology has been extensively developed [13]. To improve the reliability of the software system, software structure redundancy was proposed, and its reliability was analysed on the basis of the analysis of the relationship between system structure redundancy and reliability.

2. Software structure redundancy and its impact on reliability

Reliability of a software system refers to the key qualities that the system requires to satisfy; these key qualities are called highly reliable qualities of a system. The violations of these key qualities will cause intolerable losses of the software system [3]. To reduce the incidence of software failure, those key components that are prone to failure need to be identified and then adjusted.

In the module-based software system, the module is a unit with a certain scale (relative independence and replaceability) and has a stable composition mode to complete a definite and distinguishable function. The module is the component of the software system and the carrier of the software in its high-reliability structure design. Module redundancy is the key module that slotted off when several modules in the software system fail. To ensure the reliability of the software system, other redundant modules can replace the failure module to continue to provide services. In the system, if the modules M1 and M2 have the same function, then they are called M1 and M2 are mutually redundant modules. When a module fails, the module that continues to provide services must be detected and reconfigured for the data environment to ensure the consistency of the system’s operation. Therefore, a system with redundant modules will not only not affect the implementation of the system but also improve the fault tolerance of the system to satisfy the requirements of reliability.

3. Dual modular redundancy structure and performance of a single module

A system with a dual modular redundancy structure refers to the existence of two fully redundant modules in the system, in which two identical modules run in parallel, and the results will be detected.

The two modules in the dual modular redundancy system must produce the same result when they operate under the same input and environment, as illustrated in Figure 1.

![Figure 1. Dual-redundant structure of the module](image-url)
assuming that $M_1$ and $M_2$ are completely independent and that the probability of failure is the same as $p$, the probability of detecting the failure in the system operation is $1 - (1 - p)^2 = p(2 - p)$. The probability that one has a failure and the other does not while the whole is detected as a failure is $2p(1 - p)$.

4. Dual modular redundancy structure and performance of a combined module

The modules in software systems may have different relationships that require redundant processing. These relationships may be parallel, series, and multiple modules. The modules can be removed to yield improved results. To simplify the modules, the parallel and series relationships between the two modules are analysed in the present study, and the relationship among multiple modules are similarly processed.

4.1. Dual modular redundancy of two parallel modules

Redundant modules with different parallel structures are represented by various subscripts. For example, $M_1$ and $B_1$ and $M_2$ and $B_2$ are two sets of the same redundant modules. Two main ways can be adopted to achieve dual modular redundancy for two parallel modules, as demonstrated in Figures 2 and 3.

**Figure 2.** Parallel module dual modular redundancy mode 1

For parallel module dual modular redundancy mode 1, $M_1$ and $B_1$ with two parallel relationships and $M_2$ and $B_2$ with a series relationship initially run in parallel. Then, they are detected to determine whether a failure occurs. At this time, if a module failure occurs, then the system will be considered to fail. If the probability of failure of $M_1$ and $M_2$ is $p_1$, then the probability of failure of $B_1$ and $B_2$ is $p_2$. If each module is independent, then the probability of failure when $M_1$ and $B_1$ are paralleled is $1 - (1 - p_1)(1 - p_2)$. The results of the two sets of parallel modules are inconsistent, and the probability of failure is considered $1 - (1 - (p_1 + p_2 - p_1 p_2))^2$ after detecting the detector.

**Figure 3.** Parallel module dual modular redundancy mode 2

For parallel module dual-modular redundancy mode 2, given the first modules’ ($M_1$ and $M_2$) and second modules’ ($B_1$ and $B_2$) redundant treatment, their fault detection does not affect each other. The probability of failure caused by the parallel mode of $M_1$ and $M_2$ is $1 - (1 - p_1)^2$. The probability of failure
caused by the parallel mode of \( B_1 \) and \( B_2 \) is \( 1 - (1 - p_1)^2 \). In this case, if a detector detects a fault, then the detector is considered to have failed, and the probability is:

\[
(2p_1 - p_1^2) + (2p_2 - p_2^2) - (2p_1 - p_1^2)(2p_2 - p_2^2).
\]

4.2. Dual modular redundancy of two series modules

Two main ways may be used to achieve redundancy for two series modules, as exhibited in Figures 4 and 5.

For the series module dual modular redundancy mode 1, after operating the two modules in series, they are checked for the same result to determine if a fault has occurred. If the probability of failure of \( M_1 \) and \( M_2 \) is \( p_1 \), then the probability of failure of \( B_1 \) and \( B_2 \) is \( p_2 \). If they are independently operated, then the probability of failure when \( M_1 \) and \( B_1 \) are in series is \( p_1 + p_2 - p_1p_2 \). At this time, if one of the four modules fails, then the detector considers a failure in the system, and its probability is \( 1 - (1 - (p_1 + p_2 - p_1p_2))^2 \).

For the series module dual modular redundancy mode 2, two series modules are operated separately to detect whether the same operating results will determine a failure. The two modules are detected separately, and the number of detection will increase to determine whether \( M_1 \) or \( M_2 \) or \( B_1 \) or \( B_2 \) caused the system failure in accordance with the test results. If \( M_1 \) and \( M_2 \) and \( B_1 \) and \( B_2 \) are detected, the probabilities of failure are \( 1 - (1 - p_1)^2 \) and \( 1 - (1 - p_2)^2 \), correspondingly. Therefore, for the series module dual modular redundancy of the corresponding mode 2, the probability of failure detected in the operation is \( 2p_1(1 - p_1) + (1 - 2p_1)(1 - p_1)(1 - (1 - p_1)^2) \).

5. Conclusion

Software systems are becoming increasingly important in the socioeconomic progress, and life is driven by related applications and backgrounds; failure of software systems can frequently have serious and even disastrous consequences for humans and the environment. Software reliability, as a valuable and challenging research direction in the computer science field, has been highly valued by the scientific community and industry.

The root cause of unreliable software systems being built is the existence of a fault, which indicates defective parts in the software system. The system’s fault-tolerant method can be used to reduce the incidence of software failures, to detect whether the system is faulty in a system where the failure already exists, and to determine the components that have failed. Thus, to improve the reliability of the software
system, redundancy mechanisms are added in this study to design a module-based software system structure. The method of the redundancy mechanism in a software system is to perform a redundant treatment on the key and core modules that may fail, including the dual modular redundancy structure of a single module and the dual modular redundancy mode of the two series/parallel modules. Thus, other redundant components can continue to provide services even when several key modules in the software system fail; this mechanism ensures the reliability of the software system operation.

References

[1] Laprie J C. Dependability: Basic concepts and terminology. Vienna: Springer-Verlag, 1991.
[2] NSTC. Research Challenges in High Confidence Systems. Proceedings of the Committee on Computing Information and Communications Workshop, 1997.
[3] Chen Huowang, Wang Ji, Dong Wei. High Confidence Software Engineering Technologies. ACTA ELECTRONICA SINICA, 2003, 31 (S1): 1933 - 1938.
[4] Saha G K. Software based fault tolerance: a survey. Ubiquity, 2006, 7 (25): 1 - 15.
[5] Reis G A, Chang J, Vachharajani N, Rangan R, August D I, Mukherjee S S. Software-controlled fault tolerance. ACM Transactions on Architecture and Code Optimization, 2005, 2(4): 366-396.
[6] Reis G A, Chang J, Vachharajani N, Rangan R, August D I. SWIFF: Software implemented fault tolerance. Proceedings of the International Symposium on Code Generation and Optimization. Washington DC, USA: IEEE Computer Society. 2005: 243 - 254.
[7] Xie Zaipeng, Sun Hongyu, Saluja K. A survey of software fault tolerance techniques [EB/OL]. http://www.pld.ttu.ee/IAF0030/Paper_4.pdf, 2010-05-22.
[8] Sathre J, Zambreno J. Automated software attack recovery using rollback and huddle. Springer Journal of Design Automation for Embedded Systems, 2008, 12 (3): 243 - 260.
[9] He Jialang, Zhang Kun, Meng Jin, Zhang Hong. Hybrid Software Fault-tolerant Model Based on Evolvable-modules Redundancy Journal of Nanjing University of Science and Technology, 2012, 36 (2): 272 - 277.
[10] Zhang Lizhong. Design of a dual bus dual controllers PLC software redundancy system. Journal of Shaanxi Shaanxi University of Technology (Natural Science Edution), 2014, 30 (3): 41 - 46.
[11] Huang Tao, Chen Xiangxian, Huanghai. Safety Computer System Based on 2 out of 3 Redundant Structure Computer Engineering, 2011, 37 (18): 254 - 257.
[12] Yang Fuqing. Thinking on the Development of Software Engineering Technology. JOURNAL OF SOFTWARE, 2005, 16 (1): 1 - 7.
[13] Atkinson C, Buns e C, Gross H-G, Peper C. Component-based Software Development for Embedded Systems. Berlin Heidelberg, Germany: Springer-Verlag, 2005.