Hindawi
Science and Technology of Nuclear Installations
Volume 2017, Article ID 2981943, 9 pages
https://doi.org/10.1155/2017/2981943

Research Article

Design and Development Framework of Safety-Critical Software in HTR-PM

Chao Guo, Huasheng Xiong, Xiaojin Huang, and Duo Li

Institute of Nuclear and New Energy Technology, Collaborative Innovation Center of Advanced Nuclear Energy Technology, Key Laboratory of Advanced Reactor Engineering and Safety of Ministry of Education, Tsinghua University, Beijing 100084, China

Correspondence should be addressed to Duo Li; liduo@tsinghua.edu.cn

Received 29 December 2016; Revised 19 May 2017; Accepted 28 May 2017; Published 28 June 2017

Academic Editor: Luca Podofillini

With the development of information technology, the instrumentation and control system of nuclear power plant nowadays rely heavily on the massive and complex software to ensure the safe and efficient operation of the power plant. The improvement of the software design and development for the safety systems has been a research focus for its decisive impact on the nuclear safety. The framework of the software design and development for reactor protection system in High Temperature Gas-Cooled Reactor-Pebble bed Module was introduced in this paper. Firstly, during the design period, in addition to multichannel redundancy, grouping of protection variables and diverse 2-out-of-4 logics were adopted by different subsystems of each channel in case of common cause failure. Then a series of development characteristics together with strict software verification and validation were performed. Thirdly, during the software test period, an improved software reliability growth model based on the Goel-Okumoto model according to the analysis of fault severity was proposed to help in estimating the reliability of the software product and identifying the software release time.

1. Introduction

The analog instrumentation and control (I&C) systems are being replaced by the digital ones in the nuclear power plants (NPPs) during the last decade as the rapid development of computer science. Although providing a series of advantages such as better accuracy of monitor and control, more friendly human-machine interface, and higher automation level, digital I&C systems are still facing challenges connected with the modern information technologies [1]. The digital system also brings challenges to the probability safety assessment (PSA) of the NPP as there is no mature and widely used method for software reliability evaluation.

The improvement and evaluation of software reliability for the digital I&C system have been a research focus and the Nuclear Regulatory Commission of America also referred to this research topic in its five-year research plan for digital I&C system proposed in 2010 [2]. This issue is even critical for the safety-related systems like the reactor protection system (RPS) and the software failure could be unacceptable.

The design and development of digital I&C systems are different from the analog ones. For example, the redundancy arrangement of multiple channels is widely used to reduce the failure rate effectively for the analog RPS, while the digital component brings common cause failure (CCF) possibility which cannot be overcome by redundancy as all redundant channels are of the identical software and hardware. The regulatory authority emphasizes quality, diversity, and defense-in-depth as procedures against CCF of digital RPS [3].

Besides quality control (QC) process, the safety-critical software of an NPP has to undertake independent verification and validation (V&V) process following the requirement of the IEEE 1012 [4], which guarantees the software quality by enforcing rigorous V&V activities. Documentation audit is the major activity taken during the early period of the V&V process while the software unit test and the integrated test play a crucial role in the software coding and system integration.

Software reliability growth models (SRGMs) are commonly used to assess the software reliability. The SRGMs

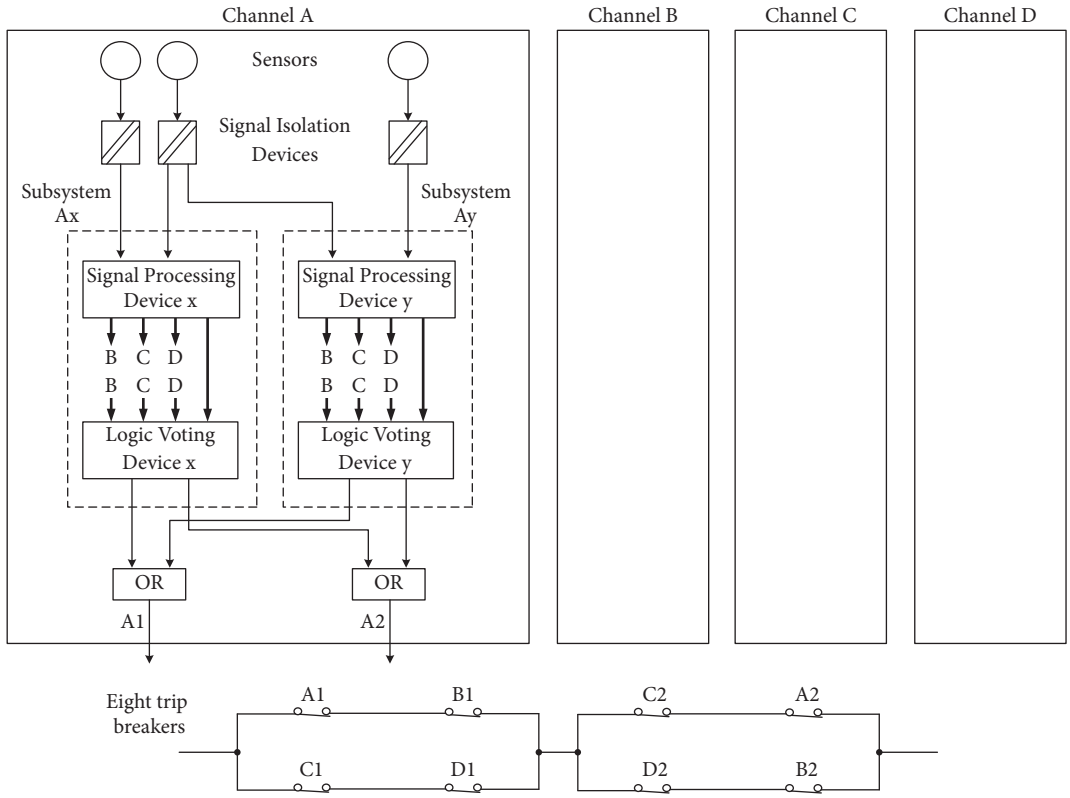

Figure 1: Architecture of the reactor protection system (RPS) in HTR-PM.

are able to track the reliability growth curve during the test process which helps the development group to evaluate the number of residual faults and the development cost and to make decisions for the software release time and the test resource allocation [5–8]. During the last four decades, hundreds of software reliability models have been developed for different kinds of applications, and the available models are usually based on certain assumptions and limitations [9, 10]. With the particularity of safety-critical software of the RPS, the severity of the software fault should be studied and the critical faults should be paid more attention.

High Temperature Gas-Cooled Reactor-Pebble bed Module (HTR-PM) is the first HTGR to be commercially operated in the world, and the RPS of HTR-PM is the first digital RPS domestically designed, developed, and commercially operated in China. There are several characteristics for the software design and development process of RPS in HTR-PM. For example, it employs two groups of protection variables and diverse 2-out-of-4 voting logics for two subsystems in each channel in case of CCF. In addition, it is a custom-made system which only serves the HTR-PM, and the software is independently developed and is not based on the configuration of function modules on a platform; thus the software has been simplified effectively [11].

In this paper, the architecture of the RPS in HTR-PM is first introduced in Section 2. Section 3 shows the software design and development features, including the voting logic diversity, development optimization, and the software V&V. A novel software reliability growth model based on the analysis of fault severity is derived in Section 4. At last, the conclusion of this paper is given in Section 5.

2. Architecture of Reactor Protection System

The RPS of HTR-PM can generate the reactor trip signal and the engineered safety featured actuation signal (ESFAS) in case of design basis accidents. The RPS has four redundant channels, A, B, C, and D, together with eight trip breakers as shown in Figure 1. Each channel includes several sensors, several Signal Isolation Devices, two Signal Processing Devices, and two Logic Voting Devices. They can fulfil the functions including signal measurement and isolation, unit transformation, setpoint comparison, and the first-step 2-out-of-4 (2/4) logic. The second-step 2/4 voting logic is performed by eight trip breakers.

Only Signal Processing Devices and Logic Voting Devices are the digital ones among these devices, and others are all analog ones. The Signal Processing Device x and the Logic Voting Device x in one channel are defined as subsystem x. Similarly, the Signal Processing Device y and the Logic Voting Device y in one channel are defined as subsystem y. Both subsystems in one channel can achieve the unit transformation and 2/4 voting functions. The hardware and software of subsystems Ax, Bx, Cx, and Dx in four channels are identical, and the hardware and software of subsystems

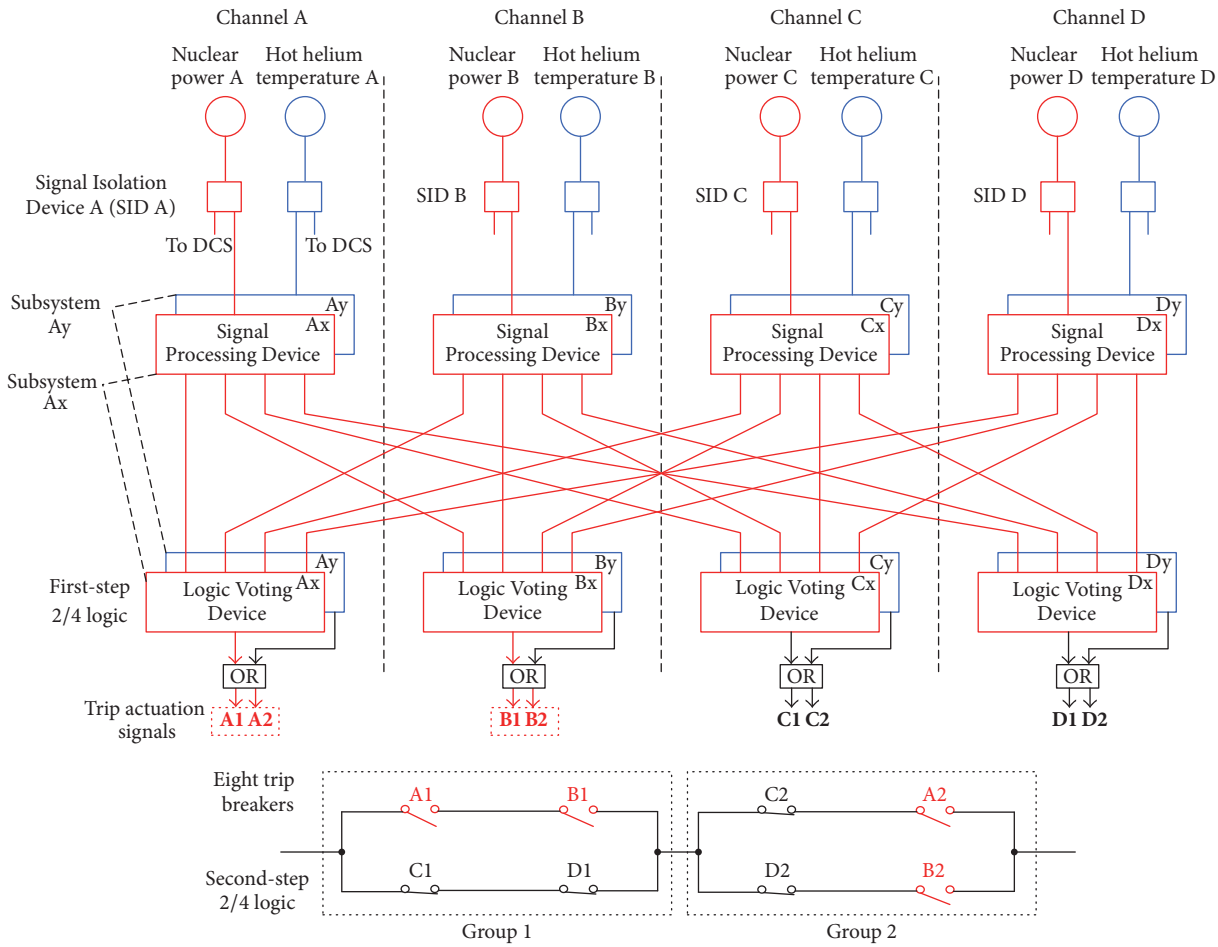

FIGURE 2: An example for the grouping of protection variables in two subsystems of one channel.

Ay, By, Cy, and Dy in four channels are identical, too. But the software of subsystems x and y is different in case of CCF.

Each channel receives dozens of hardwired analog signals from the sensors including the nuclear power, hot/cold helium temperature, helium pressure, and steam pressure. Some other switch signals like the status of the nuclear instrumentation system are also collected by the RPS for certain logic decision. It should be noted that there are a set of independent sensors for each channel which is subject to the principles of the redundancy requirement and the entity separation requirement.

The Signal Isolation Devices in each channel allocate the input signals to subsystems x and y of the same channel, the distributed control system (DCS), and the postaccident monitoring system if necessary. These signals are isolated electrically to ensure that the single fault in any equipment will not affect others.

Each design basis accident of HTR-PM corresponds at least two protection variables which are then divided into two groups. Subsystems x and y are used to deal with these two groups of protection variables, respectively, in order to decrease the risk of CCF. Figure 2 shows an example for the grouping diversity. When the design basis accident "false removal of control rods in power operation condition of HTR-PM" happens, there are two consequences: (1) abnormal increase of nuclear power; (2) abnormal increase of hot helium temperature. Two protection variables "nuclear power" and "hot helium temperature" are then assigned to subsystems x and y, respectively. As the protection variables that two subsystems deal with are different, the corresponding software for unit transformation and protection logics is also different. Each subsystem outputs reactor trip signals and ESFAS signals independently, so subsystems x and y act as the grouping diversity for safety functions. In addition, the voting logic diversity of two subsystems will be introduced in Section 3.1.

Figure 2 also shows the topological relations between four channels: every Signal Processing Device x sends information to four Logic Voting Devices x simultaneously, and the first-step 2/4 logic is realized by the Logic Voting Devices x. Similar processes are taken for four subsystems y. Taking Channel A as an example, the Logic Voting Devices Ax and Ay each generate a reactor trip signal according to the 2/4 voting results and these two signals then perform "OR" logic to generate two trip actuation signals A1 and A2 to actuate trip breakers A1 and A2, respectively. Thus even if one subsystem

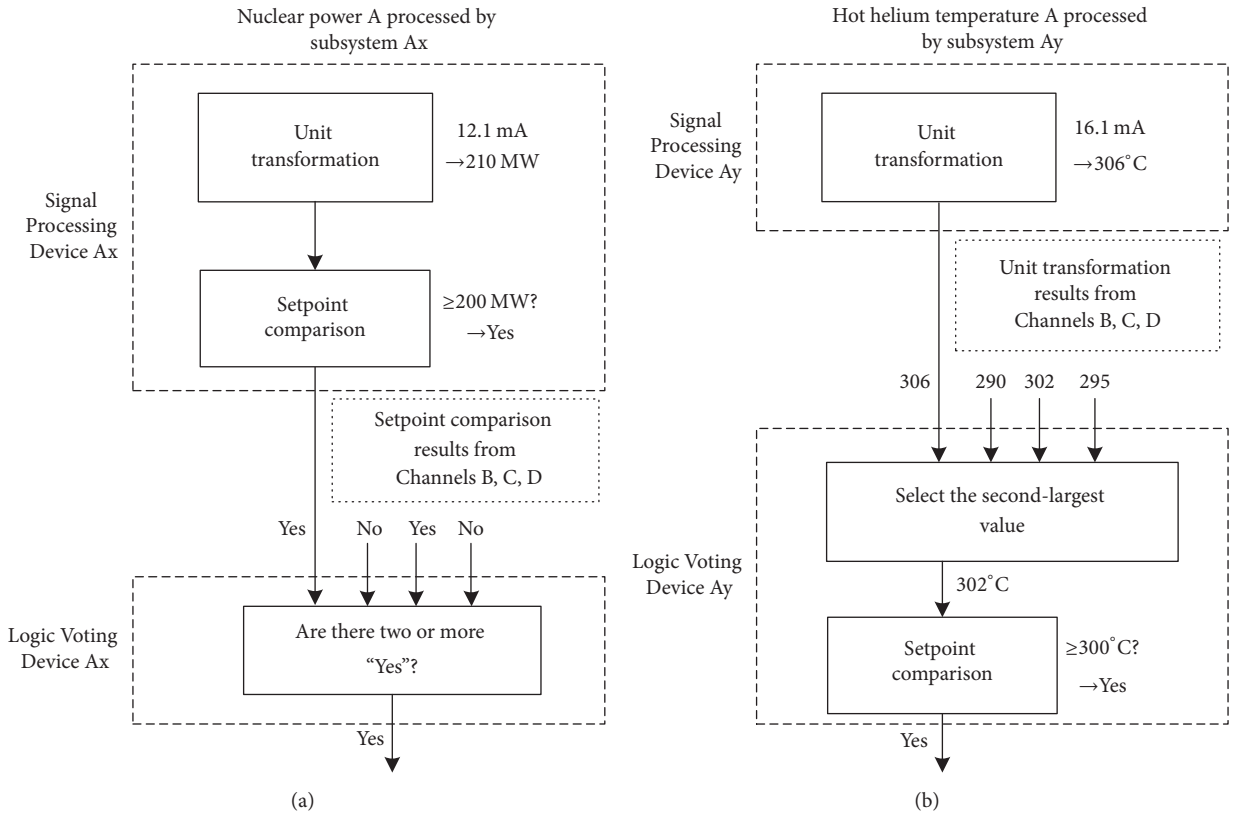

Figure 3: An example of diversity of 2/4 voting logic for (a) subsystem Ax and (b) subsystem Ay in Channel A.

fails to generate reactor trip actuation signal in the case of a design basis accident, the diverse subsystem can also ensure the reactor trip.

Eight trip breakers with series-parallel structure are used to cut off the power supply of control rod system. They are assigned to two groups as shown in Figure 2. Each channel of RPS controls two trip breakers belonging to different groups. If any 2-out-of-4 channels output the trip actuation signals, the reactor trip is realized. In Figure 2, Channels C and D are supposed to fail to work and only trip breakers A1, A2, B1, and B2 controlled by Channels A and B are successfully opened. It can be found that the power supply of the control rod can still be cut off under such circumstance, so the second-step 2/4 logic is achieved.

3. Software Design and Development

3.1. Voting Logic Diversity. In addition to the grouping diversity of protection variables proposed in Section 2, the voting logic diversity is also employed against CCF in the RPS of HTR-PM. Specifically, the 2/4 logic in subsystem x and subsystem y of each channel is achieved using different ways. The design basis accident "false removal of control rods in power operation condition" and two related protection variables are also taken as an example to illustrate this diversity, which is shown in Figure 3.

Either subsystem in each channel consists of two devices, that is, a Signal Processing Device and a Logic Voting Device. Every subsystem is able to realize the unit transformation and the logic voting functions separately. For example, the unit transformation processes of "nuclear power" and "hot helium temperature" are performed by the Signal Processing Devices Ax and Ay in two subsystems, respectively, as shown in Figure 3. The corresponding software codes for unit transformation of these two variables in two Signal Processing Devices are naturally different.

The 2/4 logic of two subsystems is also designed to be different. In subsystem Ax, the Signal Processing Device Ax identifies whether the value of protection variable is larger than the setpoint. If so, a "Yes" signal is generated and sent to Logic Voting Devices Ax, Bx, Cx, and Dx of four channels simultaneously. On the other hand, each Logic Voting Device x receives four "Yes" or "No" comparison results from Signal Processing Devices Ax, Bx, Cx, and Dx and then performs 2/4 logic. For the example in Figure 3(a), the Logic Voting Device Ax receives two "Yes" signals so a reactor trip signal is generated according to the 2/4 logic. The same processing operations are implemented on other channels at the same time.

For subsystem y, however, after unit transformation in Signal Processing Device Ay, the value of protection variable is directly sent to four Logic Voting Devices y of four channels without setpoint comparison. Then every Logic Voting Device y chooses the second largest value from four protection variables. And the setpoint comparison is performed only for this second largest value. Take the case in

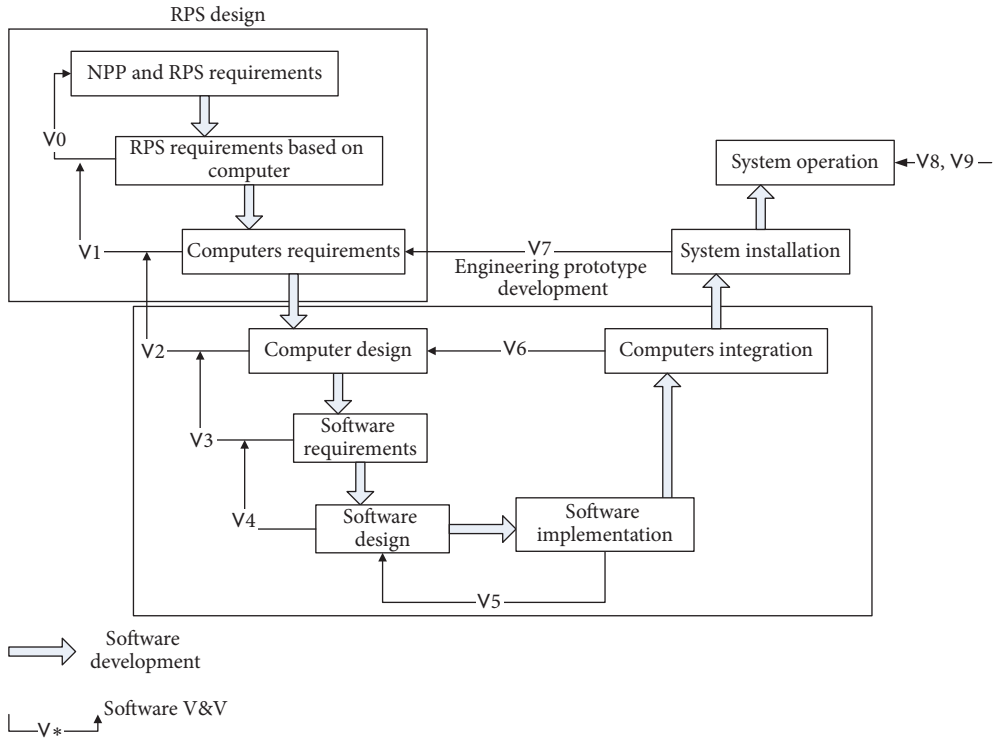

Figure 4: Software verification and validation model of RPS in HTR-PM.

Figure 3(b) as an example; as the second largest value of the hot helium temperature is greater than the setpoint, it can be derived that the largest value should also be greater than the setpoint, so the 2/4 logic is fulfilled and a reactor trip signal is generated by the Logic Voting Device Ay.

Grouping of protection variables and diverse voting logic achieve software difference between two subsystems which can effectively reduce the probability of CCF.

3.2. Software Development Characteristics. The introduction in Section 3.1 refers to the software design optimization based on voting logic diversity. In this section, the approaches taken during the software development process of the RPS in HTR-PM will be discussed.

3.2.1. No Configuration Platform. Software development based on a configuration platform is a common method for the coding of RPS, and most well-known RPSs in the world are developed in this way. The famous platforms include the Common Q platform of Westinghouse and the TELEPERM XS platform of AREVA. The platform modules are selected and configured for a specific project. One of the problems for this kind of configuration is that the reliability of the whole platform and the complex configuration tool is difficult to validate. Furthermore, the configuration process may introduce undesired functions because of functional relevance of the platform modules.

The software for the RPS of HTR-PM is not developed based on a platform. Every line of the software code is written in demand rather than being selected from the platform library. The advantage for this development method is that there is no need to prove the reliability of the configuration platform and configuration tool to the regulatory agency as no platform or configuration tool is used. This simplification reduces the difficulty for the design audit of the RPS in HTR-PM. On the other hand, all software is newly developed and only used for this project. There is no application precedent for the software code. So the software verification and validation process should be paid special attention to ensure the software reliability.

3.2.2. Customized Software. Without a configuration platform and a configuration tool, the RPS in HTR-PM is developed totally by customized software, which means that the drivers of the hardware, the database, the application functions, and the displays were all developed from the beginning. In this way, the software of the RPS in HTR-PM could be simplified effectively. The decrease in the amount of software code is an important guarantee for software reliability.

3.3. Software Verification and Validation. Software V&V for the RPS is a mandatory activity required by the related standards like the IEEE 1012 and this process should be carried out during every development period, which can ensure that the objectives "do things right" and "do the right things" [4].

Figure 4 illustrates the software V&V model [12] that the RPS in HTR-PM employed during the development process. This V-shaped model shows both the development

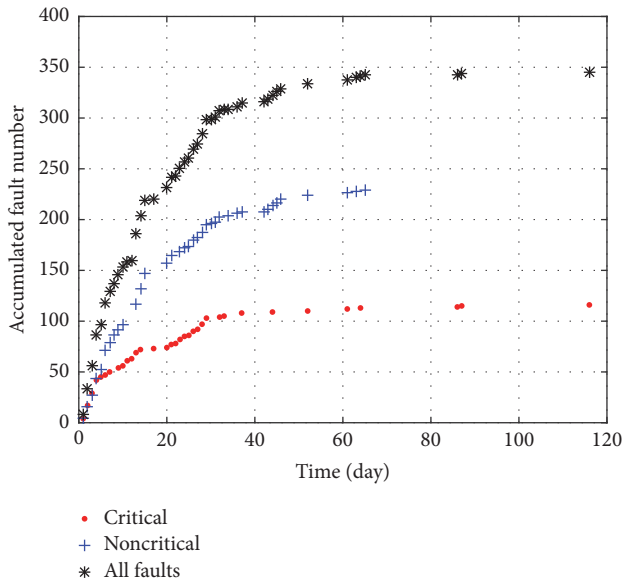

Figure 5: Accumulated fault numbers during software test.

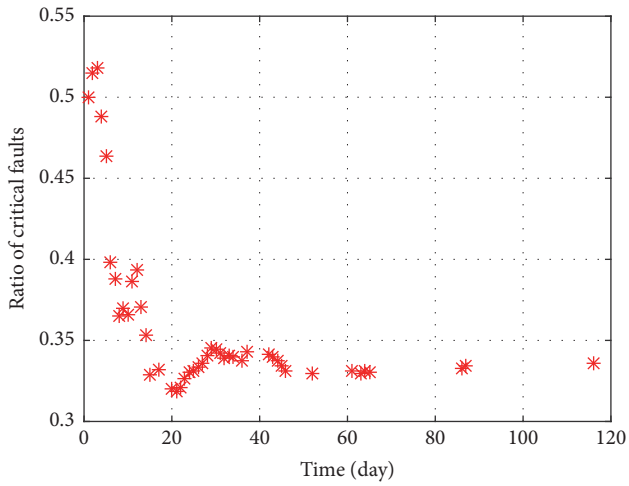

Figure 6: Ratio of critical faults to total accumulated faults.

and the V&V processes, and the output of the software development process is the input of the V&V process. For example, the output of the "software design" is the software design specification, and the output of the "software implementation" process is the software code, so the duty of the V&V process "V5" is software testing based on the software design specification. In addition to software testing, there are two main activities for the software V&V: in the early stages of development, the major V&V activity is document audit; in the later part of software development, the major V&V activity is integrated testing and system testing. It should be noted that every V&V process should be taken under strict rules, and a serious of reports are expected to prove the effectiveness of the V&V activities.

According to the requirement of the related standard, an independent software V&V should be taken. The RPS of HTR-PM was developed by China Techenergy Co., Ltd., and the software V&V is taken charge of by the Institute of Nuclear and New Energy Technology of Tsinghua University, so the V&V group is organizationally independent of the development group in technical, management, personnel, and financial aspects which ensures the effectiveness of the V&V process.

4. Software Reliability Modelling

4.1. Software Test Data Analysis. A novel software reliability model based on the fault severity analysis was proposed. Figure 5 shows the accumulated fault numbers during the software test process of a processing module of the RPS in HTR-PM. The x-axis means the date to detect the fault and the y-axis means the accumulated fault number. The faults are classified to critical and noncritical faults according to the severity recorded in the fault report. It can be found that the accumulated faults increase quickly at the beginning of the test and around 90% of the faults were detected during the first 1/3 of test period. As the critical faults have more severe effects on the safety functions of the RPS and have a greater impact on software reliability, the critical fault was specially studied in this paper.

The ratio of the critical faults to total faults was analyzed and the ratio curve is shown in Figure 6. It can be found that, during the first 20% of the test period, the proportion of critical fault declines rapidly; during the remaining 80% of the test period, the proportion of the critical fault basically remains unchanged. This trend is related to many factors, such as the testers' experience and the test strategies. By including more parameters related to the testing process in the SRGM, the model is closer to reality and the precision of the model can be improved. Specifically, the ratio trend in Figure 6 can be fitted with an exponential function.

4.2. Proposed Software Reliability Model. Suppose that $\{N(t), t \geq 0\}$ is a counting process, where N is the total detected faults in time t. A mean value function (MVF) $m(t)$ is used to express the expected number of accumulated faults by time t, and the general form of the SRGM based on a nonhomogeneous Poisson process (NHPP) that has Poisson distribution with mean $m(t)$ is described as

$$P\{N(t) = n\} = \frac{m(t)^n}{n!} e^{-m(t)}, \quad n = 0, 1, 2, \ldots \tag{1}$$

and $m(t)$ can be expressed in terms of the failure intensity function $\lambda(t)$:

$$m(t) = \int_0^t \lambda(x)\, dx. \tag{2}$$

Goel-Okumoto (G-O) model is a common and effective SRGM [13] and $m(t)$ for the G-O model is

$$m(t) = a\left(1 - e^{-bt}\right). \tag{3}$$

Here a is the expected total number of faults in the software and b is the detection rate per fault. The corresponding failure intensity function $\lambda(t)$ is expressed as

$$\lambda(t) = abe^{-bt}. \tag{4}$$

In this paper, the faults are classified to critical and noncritical faults as illustrated in Figure 5. The ratios of accumulated critical and noncritical faults to total fault are supposed to be r_c and r_n, respectively. It is easy to get the expression of the MVFs of the critical and noncritical faults:

$$\begin{aligned} m_c(t) &= a \cdot r_c(t) \cdot \left(1 - e^{-bt}\right) \quad (0 \le r_c \le 1), \\ m_n(t) &= a \cdot r_n(t) \cdot \left(1 - e^{-bt}\right) \quad (0 \le r_n \le 1). \end{aligned} \tag{5}$$

And, according to the definition, the relationship of r_c and r_n is given by

$$r_c + r_n = 1. \tag{6}$$

Based on the analysis in Section 4.1, the ratio of critical fault can be regarded as an exponential function. So r_c is supposed to have the form as follows:

$$r_c(t) = xe^{-yt} + z, \quad (x > 0, y > 0, z > 0), \tag{7}$$

where x and y are defined as the proportional and exponential decrease rate of the critical-fault ratio, respectively, and z is defined as the stable value of critical-fault ratio. $r_c(t)$ approaches z when the time t is large enough. So the MVF of critical and noncritical faults can be derived by (5), (6), and (7) as follows:

$$\begin{aligned} m_c(t) &= a \cdot \left(xe^{-yt} + z\right) \cdot \left(1 - e^{-bt}\right), \\ m_n(t) &= a \cdot \left(1 - xe^{-yt} - z\right) \cdot \left(1 - e^{-bt}\right). \end{aligned} \tag{8}$$

The modelling process is shown as follows: firstly, the software fault data is classified to critical and noncritical faults according to the severity information of the test report; secondly, the ratio of critical faults to total accumulated faults is calculated; then the ratio is fitted with (7), and the parameters x, y, and z are estimated during this process; thirdly, the accumulated fault numbers are applied to fit (8), and the parameters a and b are estimated.

4.3. Experimental Results and Discussion. The software reliability model proposed above was validated with the data of software fault obtained during the development of RPS in HTR-PM illustrated in Section 4.1.

The ratio in Figure 6 was first fitted with (7) and the parameters were estimated with the Least Square Estimation (LSE) method. The data with $t \le 40$ d was used for fitting and other data were used to validate the prediction ability of the model. The fitted result is shown in Figure 7, and the specific fitting values of parameters in (7) are listed in Table 1.

This fitting result was then used to estimate the parameters in (8) and LSE was applied for parameter estimation. Let N_i^c and N_i^n, $i = 1, 2, \ldots, k$, denote the observed cumulative numbers of critical faults and noncritical faults by time t_i, respectively, where $N_i^c + N_i^n = N_i$. The parameters a and b in (8) can be estimated by solving

$$\min_{a,b} S^2 = \sum_{i=1}^{k} \left[\left(N_i^c - m_c(t_i)\right)^2 + \left(N_i^n - m_n(t_i)\right)^2\right]. \tag{9}$$

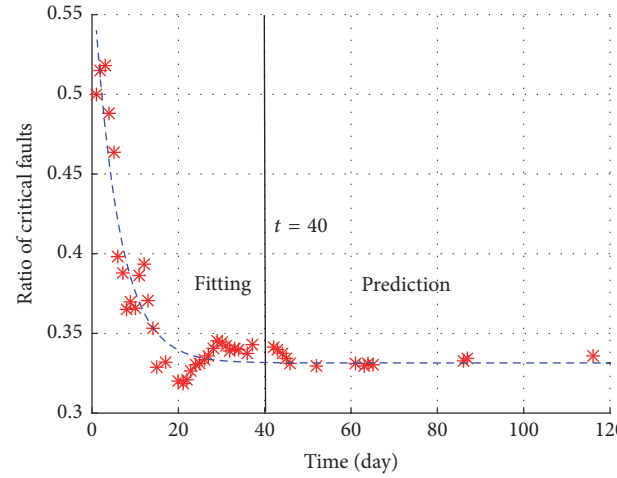

Figure 7: Fitted curve of ratio of critical faults to total accumulated faults.

Table 1: Estimated parameters of the proposed model, G-O model, and inflection S-shaped model.

Model	a	b	x/β	y	z
Our model	345.4856	0.0613	0.2477	0.1712	0.3315
G-O model	356.6320	0.0569	—	—	—
ISS model	486.5345	0.0004	−0.9913	—	—

Table 2: Fitting and prediction results of three models.

Model	RMSE	
	Fitting	*Prediction*
Our model	8.8251	2.2758
G-O model	8.5034	7.3072
ISS model	8.0388	32.6790

The G-O model and the inflection S-shaped (ISS) model [14] were also adopted for reliability modelling as a comparison to the proposed model. The fitting results and the prediction effects are shown in Table 1 and Table 2, respectively. Figures 8 and 9 illustrate the fitting and prediction results of these three models.

From Table 2 it can be easily found that the fitting effects of three models are similar while the model proposed in this paper gives the best prediction effect compared with the other two models. This conclusion can also be arrived at from Figure 9 where the proposed model gives the most accurate fitting results, which means that this proposed model is more effective in evaluating the residual faults, assessing the release time of the software, allocating the test resource, and so on.

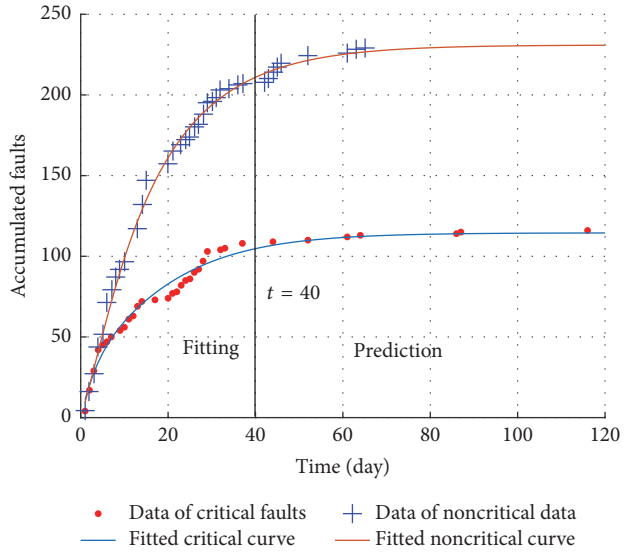

Figure 8: Fitted curves of the critical and noncritical faults.

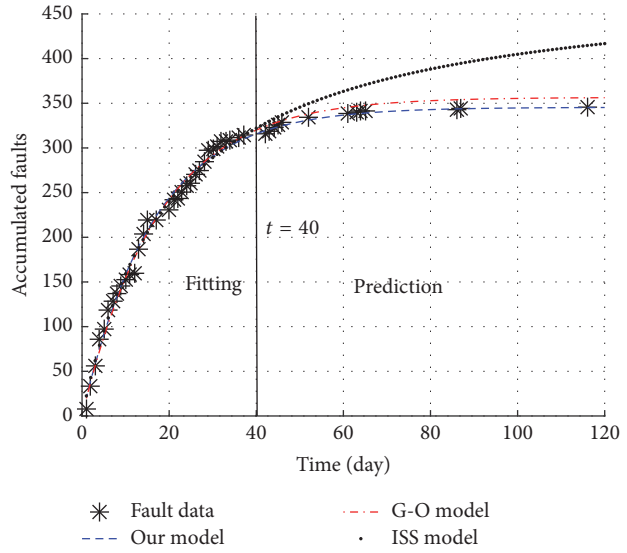

Figure 9: Fitted curves of three models for total accumulated faults.

It should be noted that the proposed model is closely related to the shape of the ratio curve. The ratio curve is affected by multiple factors, such as test case design, test strategies, and test experience of the testers. One of the premises when using the proposed model may be that the testers already have sufficient testing experience to locate the critical faults more quickly than the noncritical ones. If the accumulated fault data is collected at the beginning of the whole test project when the testers are not yet familiar with the testing tool, the tested software, and the test techniques, the shape of the ratio curve may be different. For example, the ratio of the critical faults to total faults may be small at the beginning; with the continuous progress of the testing process, the ratio may gradually increase. Then a new function should be chosen for ratio curve fitting, while the modelling approaches are similar to that proposed in this paper.

5. Conclusion

Digital instrumentation and control systems play an important role in the safe and efficient operation of the nuclear power plants. Due to the severe consequence of software failure, the design and development of safety-critical software should be paid special attention. In this paper, the software design and development framework of reactor protection system in High Temperature Gas-Cooled Reactor-Pebble bed Module were studied.

Firstly, the protection variables corresponding to every design basis accident were grouped and dealt with in two subsystems, respectively, in each channel. The 2/4 logic performed by two subsystems of each channel was also designed to be different. The grouping of protection variables and the 2/4 voting logic diversity can be effective approaches against CCF of four-channel redundancy.

A series of characteristics during the software development process were also introduced, including no configuration platform and the customized software function. Together with the software development, an independent software verification and validation activity was carried out to ensure the software reliability. At last, a novel software reliability growth model based on the G-O model was developed by considering the fault severity to estimate the reliability of the software product. This model has better prediction effect than the G-O and the ISS models. The influencing factors of the ratio curve and the processing method for a different shape of ratio curve were also discussed.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Science and Technology Major Project (Grant no. ZX06901) and Tsinghua University Initiative Scientific Research Program (Grant no. 20151080380).

References

[1] M. Yastrebenetsky and V. Kharchenko, "Reliability and safety of nuclear power plant instrumentation and control systems: New challenges and solutions," in *Proceedings of the 2nd International Symposium on Stochastic Models in Reliability Engineering, Life Science, and Operations Management (SMRLO '16)*, pp. 47–55, February 2016.

[2] U.S. Nuclear Regulatory Commission, in *NRC Digital System Research Plan FY 2010-FY 2014*, 2010.

[3] NUREG-0800 Chapter 7 BTP 7-14, *Guidance on Software Reviews for Digital Computer-Based Instrumentation and Control Systems*, Chapter 7, 2016.

[4] IEEE 1012-2004, "IEEE Standard for Software Verification and Validation," 2016.

[5] X. Li, M. Xie, and S. H. Ng, "Sensitivity analysis of release time of software reliability models incorporating testing effort with multiple change-points," *Applied Mathematical Modelling. Simulation and Computation for Engineering and Environmental Systems*, vol. 34, no. 11, pp. 3560–3570, 2010.

[6] C.-Y. Huang and M. R. Lyu, "Optimal release time for software systems considering cost, testing-effort, and test efficiency," *IEEE Transactions on Reliability*, vol. 54, no. 4, pp. 583–591, 2005.

[7] K.-C. Chiu, J.-W. Ho, and Y.-S. Huang, "Bayesian updating of optimal release time for software systems," *Software Quality Journal*, vol. 17, no. 1, pp. 99–120, 2009.

[8] M. Xie and B. Yang, "A study of the effect of imperfect debugging on software development cost," *IEEE Transactions on Software Engineering*, vol. 29, no. 5, pp. 471–473, 2003.

[9] Y. Yang and R. Sydnor, "Multi-threads software reliability estimation based on test results and software structure," in *Proceedings of the 10th International Conference on Probabilistic Safety Assessment and Management 2010 (PSAM '10)*, pp. 2326–2336, June 2010.

[10] NUREG/CR-6942, *Dynamic Reliability Modeling of Digital Instrumentation and Control Systems for Nuclear Reactor Probabilistic Risk Assessments The Ohio State University U . S . Nuclear Regulatory Commission Office of Nuclear Regulatory Research Dynamic Reliability Modeli*, 2007.

[11] D. Li, H. Xiong, and C. Guo, "Design and Development of HTR-PM Reactor Protection System," in *Proceedings of the 21st International Conference on Nuclear Engineering*, Chengdu, China, 2013.

[12] National Nuelear Safety Administration of China, *HAD 102/16 Software for computer based systems important to safety in nuclear power plants*, 2004.

[13] A. L. Goel and K. Okumoto, "Time-dependent error-detection rate model for software reliability and other performance measures," *IEEE Transactions on Reliability*, vol. 28, no. 3, pp. 206–211, 1979.

[14] M. Ohba, "Inflection S-shaped software reliability growth model," in *Stochastic models in reliability theory*, vol. 235, pp. 144–162, Springer, 1984.