Received 30 June 2022, accepted 28 July 2022, date of publication 5 August 2022, date of current version 10 August 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3196934

RESEARCH ARTICLE

Standardization Workflow Technology of Software Testing Processes and its Application to SRGM on RSA Timing Attack Tasks

NAN LI[1,2], QIANG HAN[1,2], (Member, IEEE), YANGYANG ZHANG[3], CONG LI[1,2], YU HE[1,2], HAIDE LIU[1,2], AND ZIJIAN MAO[1,2]

[1]School of Computer Science and Engineering, North Minzu University, Yinchuan 750021, China
[2]The Key Laboratory of Images & Graphics Intelligent Processing of State Ethnic Affairs Commission, North Minzu University, Yinchuan 750021, China
[3]China Electronics Standardization Institute, Beijing 100007, China

Corresponding author: Qiang Han (hanqiang@nmu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61862001, and in part by the High Level Talents (Top Academic Talents) of North Minzu University under Grant 2019BGBZ07.

ABSTRACT Due to changing software requirements and increasing system complexity, which put forward new challenges to the related testing processes to ensure software trustworthiness. Software testing is an important means to improve software quality, and standardized software testing process has become an effective measure to ensure software testing quality. In order to solve the problem of software quality decline caused by non-standard behaviors in software testing processes, this paper proposes a standardization workflow technology to construct the management system of software testing processes, and applies the workflow management system to the software testing of RSA timing attack tasks. For the numerical experimental setup, two software fault detection time-domain datasets are considered, i.e., datasets generated from the RSA timing attack program in functional testing processes and workflow verification system used to manage the RSA timing attack processes in non-functional testing processes. Then, software reliability is evaluated by employing multiple heterogeneous software reliability growth models (SRGMs) with corresponding assumed conditions, parameter information and reliability prediction methods. Furthermore, we compare the fitting power and predictive performance of multiple SRGMs in terms of the model evaluation criteria, which helps testers select a better SRGM from the model evaluation results for reliability analysis. The experimental results show that the proposed standardization workflow technology can reasonably regulate the software testing processes, making the fault data input into SRGMs more reliable and reliability analysis results more credible.

INDEX TERMS Software testing, standardization workflow technology, workflow management system, RSA timing attack tasks, software reliability growth model.

I. INTRODUCTION

With the increasing of software scale and complexity, the demand for standardized software testing process is becoming more and more urgent. It is necessary to improve the software testing processes to adapt to the changes of external environment and ensure the information security. The main purpose of software testing is to prevent the losses to users caused by internal faults of software, so it is particularly important to study an effective method to regulate the software testing processes. Software testing is a key part of the software development process, which can be used to identify whether the software meets the correctness and completeness, such as providing correct outputs for different inputs and completing a certain task within a specified time [1]. The testers need to perform corresponding functional operations under the specified test environment to find potential faults in the software [2]. The

The associate editor coordinating the review of this manuscript and approving it for publication was Porfirio Tramontana.

testers tend to consider the abnormal situation incomplete in the requirements analysis stage of software testing, resulting in an imperfect design for testing standards and test cases, thus, affecting the quality of software testing [3]. Moreover, software reliability is closely related to the scale, type and development environment of software, the number of potential faults and remaining faults in a software, the development methods adopted and the technical level of developers, etc. For these reasons, software needs to be fully tested in accordance with credible testing processes before being delivered to users [4]. The relevant standards in the field of software testing define the general model for software testing process, and the standardization as a technical means to regulate the testing processes can effectively guide the testing work. Therefore, this paper follows relevant international standard documents to formulate the corresponding testing processes for RSA timing attack tasks.

In 1977, RSA encryption algorithm was proposed by Rivest, Shamir and Adleman. RSA is a public key cryptosystem used to ensure the security of communications or transactions. It can continuously perform a large number of encryption and decryption operations on plaintext at a fast speed [5]. Kocher [6] first proposed analyzing and attacking the covert timing channel based on the RSA modular exponentiation algorithm and effectively evaluating the information leakage problem of the RSA covert timing channel. In our existing work [7], we capture the random variable characteristics, such as the execution time of different ciphertexts processed by the RSA decryption algorithm, and adopt the multivariate statistical analysis method to crack the private key to improve the accuracy of RSA attacks on private keys. However, potential faults in related software with RSA timing attack tasks may affect the attack effect and thus reduce the reliability level.

To solve the reliability problem of related software with RSA timing attack tasks, this paper proposes a management method of software testing standardization processes based on workflow technology. In the debugging process, multiple heterogeneous software reliability models are used to perform reliability growth analysis to evaluate software quality. Among them, two software fault detection time-domain datasets are considered, i.e., datasets generated from the RSA timing attack program in functional testing processes and its workflow verification system in non-functional testing processes. Based on the jBPM workflow engine, we design and implement a standardization workflow management system of software testing processes for RSA timing attack tasks. By summarizing the commonalities of different testing processes and constantly optimizing them, the process compliance is improved, and the analysis of software fault data is ensured. Furthermore, we can evaluate the software reliability by predicting reliability measurement metrics, such as the total number of faults, the number of remaining faults, and the interval time to the next fault. The main work and contributions of this research are as follows:

1) Referring to the BPMN model of the software testing process, the developed standardization workflow management system is used to realize the standardized management of the test platform, which facilitates collaboration between the testers. Thus, the testing efficiency and testing quality are improved.

2) In the reliability testing, the reliability of the RSA timing attack program and workflow verification system are analyzed and compared by using the hypergeometric distribution model, the Jelinski-Moranda (J-M) model, a software failure stochastic model based on a non-homogeneous Poisson process (NHPP) and the Burr-type XII NHPP-based model.

3) During the process of fault detection and fault repair, we can evaluate the software reliability in terms of different reliability measurement methods to provide guidance for software release decisions. In addition, model evaluation criteria are used to compare the SRGMs mentioned in this paper, and a more suitable SRGM will be applied to a future reliability analysis.

The remainder of this paper is organized as follows: Section II outlines the existing related work. In Section III, we propose a standardized testing process for RSA timing attack tasks in combination with the international standards and Chinese national standards in the field of software testing, and perform the testing work on the designed system. In Section IV, we input the fault sample data into multiple selected SRGMs, and then estimate the relevant parameters of the probability distribution model corresponding to SRGM by employing statistical analysis method, so as to evaluate the software reliability. In Section V, we compare the fitting power and predictive performance of different SRGMs in terms of the model evaluation criteria to help the testers select a more suitable SRGM for a future reliability analysis. The paper ends with an overview of some conclusions and future research directions.

II. RELATED WORK

A. SOFTWARE TESTING STANDARDIZATION

Software testing runs through the entire software life cycle and is a reliable means to ensure software quality. Software testing standardization regulates the technical methods, tools, and the organizational management of software testing through formulation, organization, implementation and supervision, which can improve the software reliability and ensure the software quality. Ding *et al.* [8] analyzed the development status and trend of software testing standardization, summarized the existing problems of the current software testing standardization and gave suggestions. Masuda *et al.* [9] proposed a method for a complex software testing analysis by using the definitions of software architecture in international standards, and applied the method to specific case studies. Shen *et al.* [10] explored ensuring software quality through software standards and technical management according to the process of the software engineering international standard ISO/IEC

25000 series in China. Shen also claimed that it was necessary to excavate software testing methods and create automatic software testing tools to continuously improve the testing process management mechanism. In the field of system and software engineering, Zhang *et al.* [11] studied the system and framework of the system life cycle process standard and analyzed the development process of system and software engineering standardization combined with other standards, which promoted the progress of standardization work in the future.

At present, most studies on software testing standardization focus on the discussion of test standards and the guarantee of standards to software quality. However, the studies on the application of standardization in business process management (BPM) of software testing is relatively few, and the management and control for software testing processes are lacking. Aiming at the shortcomings of the standardization of software testing processes mentioned above, this paper proposes a management method of software testing standardization processes based on workflow technology to realize the information management of testing workflow from the perspective of BPM.

B. jBPM WORKFLOW ENGINE

jBPM, which is short for Java Business Process Management, is an open-source workflow management framework based on the Java language, with functions, such as process definition, process deployment, process execution and process management [12]. It provides a graphical tool based on JBoss jBPM Process Definition Language (jPDL), a process designer. The jPDL uses an intuitive process definition language to describe operations, such as tasks in a business process, wait states and timers, and then correlates these operations through a process control mechanism. Software developers can take the jBPM framework as the foundation for BPM to develop business process management modules and functions. The system architecture of the jBPM workflow engine is shown in Fig. 1 [13].

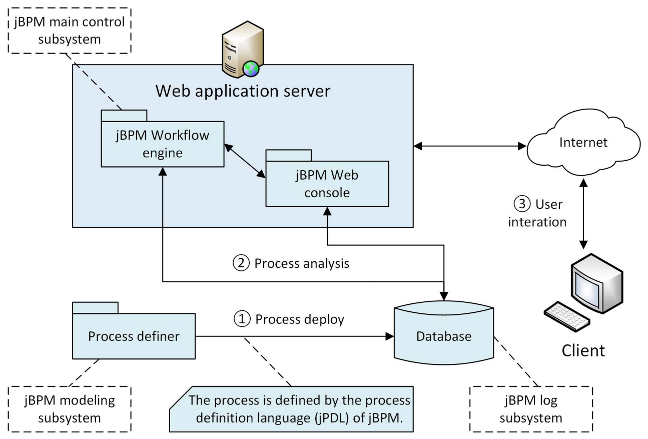


FIGURE 1. The system architecture of the jBPM workflow engine.

The process management of the jBPM is mainly divided into three stages, including process definition and deployment, process analysis and user interaction.

1) Process definition and deployment: The jPDL file of the process is defined through the process designer and deploys the process definition to the database.

2) Process analysis: The jBPM workflow engine reads and parses the process definition file that needs to be deployed, converts the jpdl.xml file into a process definition object, and persists it to the database via the jBPM web console.

3) User interaction: The saved process definition is instantiated by the jBPM workflow engine, and the client user can manipulate the process instance by invoking the interaction interface.

In this paper, we develop a standardization workflow management system of software testing processes based on jBPM engine, and the workflow verification system for ensuring the rationality of RSA timing attack processes is also developed on jBPM engine. The software testing for the workflow verification system includes functional testing and non-functional testing. Among them, the jBPM subsystem supporting the workflow verification system belongs to the non-functional testing of RSA timing attack tasks, which is composed of modeling subsystem, main control subsystem and log subsystem. The modeling subsystem is responsible for completing the process definition, the main control subsystem is responsible for deploying and analyzing the process definition objects, and the log subsystem is responsible for storing users' operation information on process instances in the database.

Aiming at the complex characteristics of software types, test roles, standard information and test implementation processes involved in software testing, as well as the research requirements of standardization workflow technology for software testing, this paper explores a design and implementation method of testing processes based on jBPM technology.

C. RSA TIMING ATTACK

The RSA algorithm obtains the public key and the private key in terms of the value of the prime number. The basic principle is that both the encryption and decryption operations for data revolve around modular exponentiation operations. During the process of RSA encryption and decryption operations, when the key bit value is 1, the modular multiplication operation needs to be implemented; however, when the key bit value is 0, the assignment operation needs to be completed [14]. There is an obvious time difference between the modular multiplication operation and assignment operation, resulting in different decryption times for different keys. The attackers can use statistical analysis and other methods to crack the key according to the characteristics of the time difference [15], [16].

For the current key bit, we assume that the attacker guesses that it is 0 or 1. When the guess result is correct, the variance is reduced; when the guess result is wrong, the variance is increased. Therefore, when the key guess is correct, the variance value is small. The attacker can use this method to repeatedly execute to obtain the key value until the complete key is obtained [17]. The variance-based RSA timing attack tasks are shown in Algorithm 1 [18].

Algorithm 1 Variance-Based Attack Method

```
Description: $T_j$ is the total running time of the $j^{th}$ ciphertext under the correct key, $T^c_{j,k}$ is the processing time of the first $(k+1)$ guess key bits of the $j^{th}$ ciphertext, $c$ is the guessed value of the $k^{th}$ key bit and $c \in \{0,1\}$ .
1. $k = 0$, $guess\_key[0] = 1$
2. FOR $k = 1$ TO $w - 1$ DO
3.   FOR $j = 1$ TO $N$ DO
4.     $\Delta T^0_{j,k} = T_j - T^0_{j,k}$
5.     $\Delta T^1_{j,k} = T_j - T^1_{j,k}$
6.   END FOR
7.   $\Delta T^0_{j,k}\ (j = 1, 2, \ldots, N) \in \Delta k^0$
8.   $\Delta T^1_{j,k}\ (j = 1, 2, \ldots, N) \in \Delta k^1$
9.   IF $Var\left(\Delta k^0\right) > Var\left(\Delta k^1\right)$ THEN
10.      $guess\_key[k] = 1$
11.  ELSE
12.      $guess\_key[k] = 0$
13.  ENDIF
14. ENDFOR
```

So far, our software testing work on RSA private key attack tasks is the variance-based side channel timing attack. The authors of literature [19] proposed a new effective data encryption algorithm, which is responsible for the encryption and decryption processes. Literature [20], [21] applied machine learning and deep learning to the field of side channel attack. These will expand new ideas for our future software testing direction.

D. SOFTWARE RELIABILITY ASSESSMENT METHOD

Software failures are caused by potential faults in the software. For the purpose of avoiding losses to users due to software faults, it is necessary to conduct software testing before releasing to find more faults and predict its reliability by using software reliability models [22]. Software reliability refers to the probability that the software runs for a specific period of time in a given environment without failures, which is one of the key factors that must be considered in the process of system design, research and operation [23], [24]. At present, there are many kinds of software reliability models, and the predictive power of different models is also different. Reliability modeling is the basis for reliability analysis and an effective means to ensure software quality. Furthermore, SRGM, as an important mathematical tool of reliability modeling, plays significant role in evaluating reliability, predicting reliability growth trend and determining optimal software release decision.

The hypergeometric distribution model can be used to predict the number of remaining faults in a software, and this model has strong scalability. Tohma *et al.* [25] proposed a new model to predict the number of remaining faults in a software based on hypergeometric distribution, and displayed the application of the model in actual data by fitting the growth trend of the cumulative number of faults. Literature [26] studied six methods for estimating parameters of the hypergeometric distribution model and compared their accuracies. Among them, the least-squares sum method can be well applied to estimate the number of residual faults in a software and the estimation time is greatly shortened. The J-M model [27] satisfies a continuous time independent distribution and is the research basis of many other reliability models, which have been widely applied. The earliest SRGM assumed that software failures satisfy the Poisson process. When the number of faults detected and successfully repaired was greater, the failure rate is lower. Liu *et al.* [28] proposed a new parameter estimation method for the J-M model, which provided guiding significance for software reliability theory and model research. NHPP belongs to an exponential failure time model, which regards the number of failures within unit interval time as an independent Poisson random variable. Goel and Okumoto [29] proposed a software fault stochastic model based on NHPP, which aimed to provide quantitative indicators for software performance evaluation, and compared with the predictive effect of the J-M model. Okamura and Dohi [30] focused on the parameter estimation problem for NHPP-based software reliability models, proposed 12 EM (Expectation-Maximization) algorithms for NHPP-based software reliability models applied to fault count data, and demonstrated that the developed algorithms are suitable for software reliability assessment through numerical experiments. Li [31] adopts Burr-type distribution to describe the distribution of software fault detection time, which proves the usefulness of the Burr-type NHPP-based software reliability models. However, these above studies only focus on whether the SRGM has good predictive power and lack of standardized constraints on software testing process, which will lead to inaccurate parameter estimation of SRGM and affect the prediction results of software reliability.

Furthermore, most software testing assessment tools focus on fault data analysis, model parameter estimation and reliability prediction. Among them, EM algorithm has been widely used in parameter estimation of software reliability models [32]. In Table 1, the analysis methods adopted by several typical tools are presented, and the standardization workflow management system proposed in this paper is compared with these tools.

III. SOFTWARE TESTING STANDARDIZATION PROCESSES AND SYSTEM IMPLEMENTATION

A. MULTI-LAYER TEST PROCESS MODEL

Statistics show that the workload of software testing usually accounts for more than 40% of the total software development, and the cost for software testing accounts for 30% to 50% of the total development cost [38]. Therefore, software testing is a key approach for risk treatment in software

TABLE 1. Comparative analysis of software testing assessment tools.

Proposed Tool	Adopted Approach
Software reliability assessment tool on spreadsheet (SRATS) [33]	It provides 11 types of NHPP-based SRGMs for fault detection data, which can be used for software reliability assessment, and the maximum likelihood estimation is realized based on EM algorithm.
Software reliability tool for PHSRM [34]	It enhances the parameter estimation algorithms for phase-type software reliability model based on EM algorithm and can handle grouped data, which is conducive to the reliability assessment of actual software development projects.
Software failure and reliability assessment tool (SFRAT) [35]	It supports the analysis of failure data generated during testing and predicts the reliability of software systems.
Covariate software failure and reliability assessment tool (C-SFRAT) [36]	It implements the test activity allocation strategy for covariate software reliability and security models, and supports the calculations of covariate models.
Automated testing tool [37]	It adopts automated testing methods with different optimization techniques to improve test efficiency, as well as reduce testing cost and time consumption.
Standardization workflow management system in this paper	This workflow management system pays more attention to the standardization of software testing processes and adopts the maximum likelihood estimation method to realize parameter estimation, which makes the fault data input to SRGMs more reliable and software evaluation results more credible.

development. With the aim of ensuring the accuracy of the RSA timing attack, the developers optimize the attack process through a workflow verification system to achieve a better attack effect. We take the related software with RSA timing attack tasks as test objects to perform the corresponding testing work. The international standard ISO/IEC/IEEE 29119-2 [39] and the Chinese national standard GB/T 38634.2-2020 [40] are taken as the implementation criteria of our testing work, providing the personnel responsible for software testing with testing guidance. Fig. 2 divides the testing activities that may be performed in the software life cycle into an organizational test process, test management processes, and dynamic test processes, known as a multi-layer test process model. Among them, the dynamic test processes are used to conduct the dynamic test in the test stage and coordinate the relationship between the testers of different roles in test management.

The responsibilities of each layer in the multi-layer dynamic test processes are as follows:

1) Organizational test process: It is used to establish and manage organizational test specifications. This process formulates detailed organizational test policy and organizational test practices for a test organization, and receives feedback from project management within the organization for update and improvement. The implementation of software testing standardization processes depends on the applicable policy and procedures of organizational test process.

2) Test management processes: They cover the monitoring and management of the entire test project or different test phases or various test type, and needs to be consistent with the organizational test process. According to the actual test situation, the problems encountered in the test management processes will generate feedback to the organizational test process, thus updating the organizational test specifications to improve the organizational test process. These processes ensure the normal execution of a project test plan.

3) Dynamic test processes: They are responsible for carrying out the generic processes defined by dynamic test within a particular test phase or test type, and coordinating the relationship between different roles of testers in the execution of test activities. The dynamic test processes include test design and implementation, test environment construction and maintenance, test execution and test incident reporting. These processes provide detailed criteria for a series of test activities in the software testing standardization processes.

B. DESIGN OF SOFTWARE TESTING PROCESSES

The requirements analysis is a necessary process of project development. The requirements analysts should accurately understand users' specific requirements on the function, performance and reliability of the project, as well as convert the users' informal requirements description into the complete requirements definition in terms of the international requirements engineering standard [41], and formulate the software requirements specification. Then, the developers need to develop the program or system on the basis of fully completing the users' requirements analysis. In an attempt to find the indigenous faults in the software more efficiently, the fault insertion personnel reasonably insert faults in terms of specified functions in the software requirements specification. Next, the testers perform the testing work in strict accordance with the software

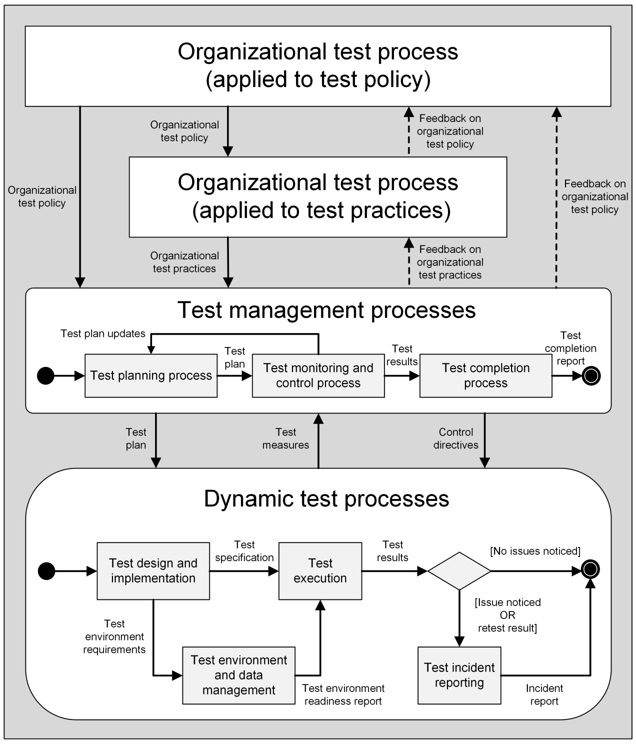

Organizational test process
(applied to test policy)
Organizational test policy
Feedback on organizational test policy
Organizational test policy
Organizational test process
(applied to test practices)
Feedback on organizational test policy
Organizational test practices
Feedback on organizational test practices
Test management processes
Test plan updates
Test plan
Test results
Test completion report
Test planning process
Test monitoring and control process
Test completion process
Test plan
Test measures
Control directives
Dynamic test processes
Test specification
Test results
[No issues noticed]
Test design and implementation
Test execution
[Issue noticed OR retest result]
Test environment requirements
Test environment and data management
Test environment readiness report
Test incident reporting
Incident report

FIGURE 2. The multi-layer model showing all test processes.

requirements specification. In the testing process, the relevant testers in different roles should follow the standardized testing process and immediately repair the detected faults, thereby continuously improving the software reliability. The reliability analysts adopt different SRGMs to evaluate the current software quality based on the test results of each test stage, providing guidance for establishing the optimal software release decision.

The business process modeling notation (BPMN) is derived from the field of process description that can be executed by the process engine of the workflow and is used to create business-level models and executable models. BPMN is adopted to conduct the modeling of the software testing process to ensure that the testing process is executed in accordance with the specified process [42]. After completing the collection and arrangement of relevant software testing technologies and standards, we also analyze the situation of software testing standardization. In the actual testing of related software with RSA timing attack tasks, we focus on using advanced information technology to combine the standardized system in the field of software testing with the actual testing processes, and make the actual testing processes conform to the corresponding specifications. At the same time, the software testing process framework of RSA timing attack tasks is constructed according to the theoretical guidance of standard documents, as shown in Fig. 3.

In Fig. 3, testers perform test activities according to the test plan, which reflects the dynamic test processes and provides guidelines for the implementation of software testing. The standardization workflow management system of software testing processes designed in this paper is based on the jBPM workflow engine, and the workflow model is established in terms of BPMN specification. The standardization workflow technology is adopted to manage the software testing, so that the testing efficiency is improved while the testing processes are standardized. In addition, it also can prevent developers and testers from deviating from requirements during the development and testing process, respectively, which reduces the software reliability. The method of standardization workflow technology is mainly composed of six key components, i.e., requirements proposal, project development, test execution, SRGM selection, data statistics and results analysis. By standardizing each link and activity content of software testing, we can construct the software testing standard system of workflow management system.

C. PROCESS OF DEPLOYMENT AND EXECUTION

For the purpose of testing the various functional modules of the RSA timing attack program and workflow verification system, we design the flow path for nodes in terms of the execution sequence and logical relationship between all test task nodes, forming a complete test execution process. The process designer is used to instantiate the test execution process conforming to the BPMN model into the workflow engine. The process definition is accomplished by creating XML files recognized by the jBPM workflow engine, and the process information is stored in the database. Table 2 shows the log information recorded in the database with respect to the testing process execution, which corresponds to the testing process specified in the BPMN diagram. It lists the test activity name, start time, end time and transition status between activity nodes.

The test execution process in the jBPM workflow engine mainly consists of a master process and a subprocess. When the process flows to the fork node, the master process begins to convert into multiple branch paths that are executed in parallel, or subprocesses. After these subprocesses converge at the join node, the process continues to flow in the path of the master process. The fork node and the join node always appear in pairs, which together constitute the subprocess set of the test execution process. The function is that multiple tasks must be completed before entering the next task of the join node. However, if the master process or subprocess fails to meet the requirements after completing a task node, the master process or subprocess rejects it to the previous task node. The test execution process of related software with RSA timing attack tasks is shown in Fig. 4, where a master process contains two subprocesses in the execution process. After the deployment of the process definition is completed, the workflow engine creates a process instance, and the deployed log information can be viewed from the database.

The test execution process mentioned above includes two types of rejection operations, as follows:

1) If the test data recorded by the subprocess are not standardized (not in conformity with the contents

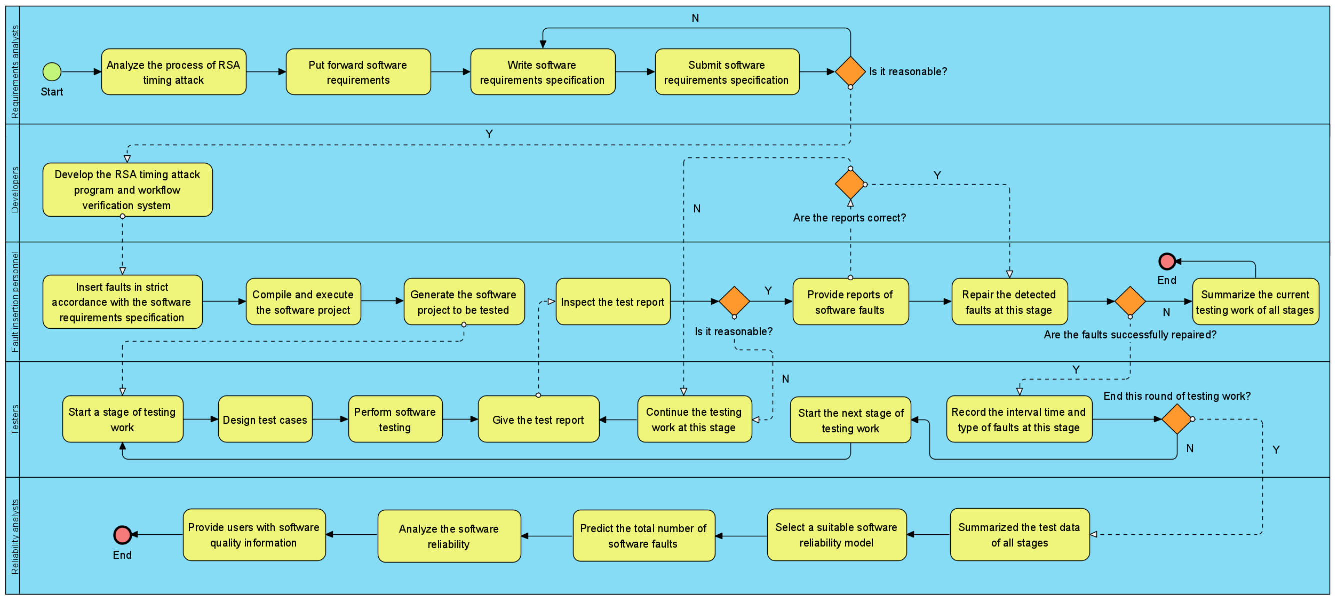

Requirements analysts
Start
Analyze the process of RSA timing attack
Put forward software requirements
Write software requirements specification
Submit software requirements specification
Is it reasonable?
N
Y
Developers
Develop the RSA timing attack program and workflow verification system
Are the reports correct?
N
Y
Fault insertion personnel
Insert faults in strict accordance with the software requirements specification
Compile and execute the software project
Generate the software project to be tested
Inspect the test report
Is it reasonable?
Y
N
Provide reports of software faults
Repair the detected faults at this stage
Are the faults successfully repaired?
N
Y
End
Summarize the current testing work of all stages
Testers
Start a stage of testing work
Design test cases
Perform software testing
Give the test report
Continue the testing work at this stage
Start the next stage of testing work
Record the interval time and type of faults at this stage
End this round of testing work?
N
Y
Reliability analysts
End
Provide users with software quality information
Analyze the software reliability
Predict the total number of software faults
Select a suitable software reliability model
Summarized the test data of all stages

FIGURE 3. The BPMN diagram of the software testing process for RSA timing attack tasks.

TABLE 2. Log information of test process execution.

ACTIVITY_NAME	START	END	TRANSITION
Analyze the process of RSA timing attack	2022-05-25 08:55:21	2022-05-25 08:55:21	to Put forward software requirements
Put forward software requirements	2022-05-25 08:55:21	2022-05-25 11:44:31	to Write software requirements specification
Write software requirements specification	2022-05-25 11:44:31	2022-05-25 13:55:50	to Are the requirements reasonable?
Are the requirements reasonable?	2022-05-25 13:55:50	2022-05-25 13:55:51	no
Modify requirements	2022-05-25 13:55:51	2022-05-25 14:43:54	to Develop RSA timing attack program and workflow verification system
Develop RSA timing attack program and workflow verification system	2022-05-25 14:43:54	2022-05-25 18:25:10	to Generate the software project to be tested
Generate the software project to be tested	2022-05-25 18:25:10	2022-05-25 18:32:43	to Start a stage of testing work
Start a stage of testing work	2022-05-25 18:32:43	2022-05-25 20:35:47	to Give the test report
Give the test report	2022-05-25 20:35:47	2022-05-25 20:36:53	to Inspect the test report
Inspect the test report	2022-05-25 20:36:53	2022-05-25 20:39:22	to Is the report correct?
...	...	...	...
Repair the detected faults at this stage	2022-05-25 22:00:46	2022-05-25 22:33:19	to Execute the next stage of testing work
Execute the next stage of testing work	2022-05-25 22:33:19	2022-05-26 09:05:56	to Record the test data of all stages
Record the test data of all stages	2022-05-26 09:05:56	2022-05-26 13:39:51	to Select a suitable software reliability model
Select a suitable software reliability model	2022-05-26 13:39:51	2022-05-26 13:47:32	to Analyze the software reliability
Analyze the software reliability	2022-05-26 13:47:32	2022-05-26 14:55:37	to Provide users with software quality information
Provide users with software quality information	2022-05-26 14:55:37	2022-05-26 15:15:59	to End

of the software requirements specification), then the subprocess is rejected to the "Execute the functional/non-functional tests" node. Until the recorded test data meets the requirements, the process flows to the next step.

2) When the master process counts test data, it finds that the test data provided by the subprocess is incomplete (the test data recorded by a subprocess is missing), and then the master process is rejected to the "Execute the software testing in compliance with the BPMN model" node, and thus, each subprocess executes the functional/non-functional tests again. Until all subprocesses provide complete test data, the process flows to the next step.

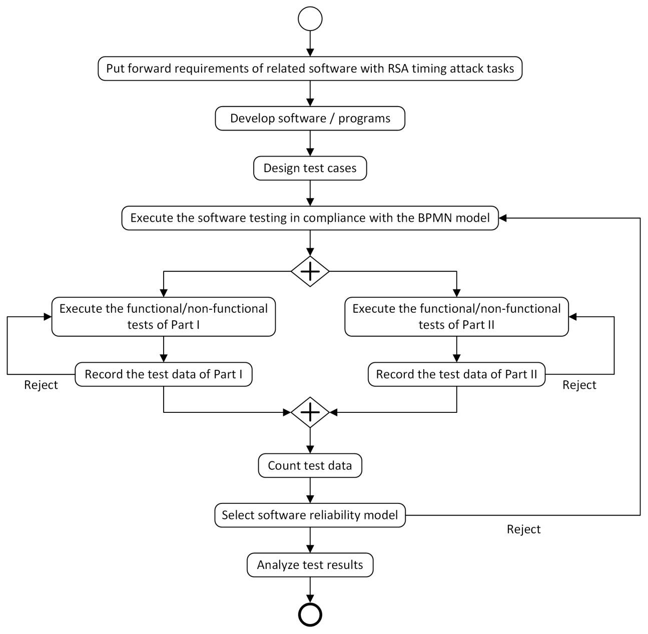

Put forward requirements of related software with RSA timing attack tasks
Develop software / programs
Design test cases
Execute the software testing in compliance with the BPMN model
Execute the functional/non-functional tests of Part I
Execute the functional/non-functional tests of Part II
Record the test data of Part I
Record the test data of Part II
Reject
Reject
Count test data
Select software reliability model
Reject
Analyze test results

FIGURE 4. The standardized test execution process for RSA timing attack tasks.

D. SYSTEM DESIGN AND IMPLEMENTATION

The standardization workflow management system of software testing processes adopts the integrated framework of 'Spring + Struts + Hibernate''. The SSH framework was developed based on Java and it is an open-source framework for web applications. Web service technology is coordinated by service-oriented workflow applications to provide different functions [43]. The application of web technology to support remote access functions facilitates dynamic collaboration between the testers [44]. Under the support of this system, testers with different roles can pass or reject the relevant test task node. The functional modules of the system mainly include requirements management, developer management, tester management, project functions management and reliability management. Among them, the requirements management module is responsible for controlling the master process of test execution, while the project functions management module and the reliability management module are responsible for controlling the subprocesses of test execution, and the master process can aggregate the test data generated by all subprocesses.

The system is mainly composed of nine objects, namely, organizational test process, requirements, project, test management processes, test design technique, test, data, SRGM and analysis. The detailed explanations for these objects are as follows:

1) Organizational test process: The organizational test specifications established and managed by this process is responsible for standardizing the overall test process and supporting the entry of test policy, test practices and test strategy, thus providing guidance for relevant testing work. During actual testing, this process can monitor the usage of organizational test specifications for timely updates and maintenance.

2) Requirements: The software requirements specification is the basic criterion that developers, testers and test management personnel should follow together.

3) Project: Developers realize software project according to software requirements. In this paper, the software projects under the test are RSA timing attack program and workflow verification system.

4) Test management processes: The test management personnel control and monitor the testing processes according to the project test plan, and employ Petri nets modeling technology to analyze the test execution logs to determine the rationality and correctness of the testing processes. Furthermore, the test management processes feed back the test progress to the organizational test process, which together maintain the orderly implementation of the dynamic test processes.

5) Test design technique: During the execution of dynamic testing, appropriate test design technique should be selected in terms of the test project, such as specification-based testing (black box testing), structure-based testing (white box testing) and experience-based testing.

6) Test: Software testing is performed under a standardized testing process to find software faults and measure software quality, which is a process for providing testing information.

7) Data: Test data are observed in actual software testing processes, and two software fault detection time-domain datasets are considered in this paper. In addition, the master process can aggregate the test data generated by all the subprocesses and persist the data into the database.

8) SRGM: Test data is input into multiple SRGMs to solve the parameter values of the probability distribution model corresponding to each SRGM. The selected SRGM includes four models, which are hypergeometric distribution model, J-M model, G-O NHPP model and Burr-type XII NHPP-based model.

9) Analysis: Reliability analysts employ SRGM to evaluate the software reliability and feed back the visual analysis results to users, developers and testers, which is conducive to establishing the optimal software release decision.

Fig. 5 is the class model of the system, reflecting various relationships between class objects. Each class object has corresponding attributes and methods, and the system can invoke them during runtime. The relationships between class objects in Fig. 5 mainly include dependency, association and generalization. Specifically, requirements are monitored by the organizational test process and need the participation of the test management processes. Therefore, the requirements class has dependency and association relationship with the organizational test process class and the test management processes class, respectively. The requirements class and the project class are associated to each other. The updates of requirements will promote the iterative development of project, and the defects reflected in project development will further improve the requirements. The project class and the test class are also associated to each other, and the project reliability can be improved by repairing the

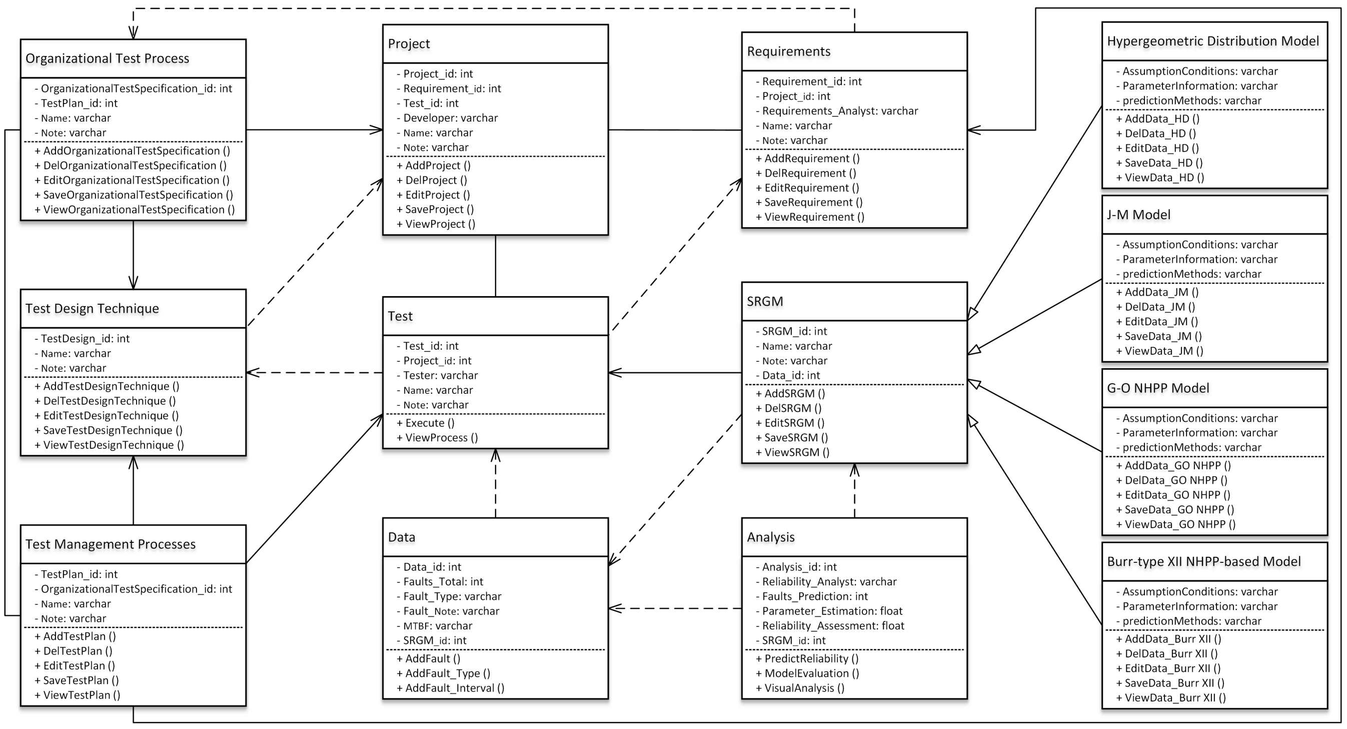
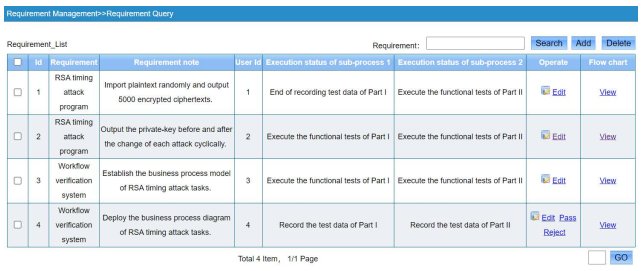


FIGURE 5. Class diagram for the management system of the software testing standardization processes.

faults detected in the testing process. The test class depends on the requirements class and the test design technique class, because the test implementation needs to follow the software requirements specification and test design content. Also, the test class is monitored and controlled by the test management processes class. The SRGM class has generalization relationship with the four model classes, and these model classes are indirectly related with the other class objects through the SRGM class. The SRGM class is associated with the test class and depends on the data class. Moreover, the analysis class depends on the SRGM class. For one thing, SRGM needs to be driven by data, and the data depends on actual test execution; for another, the analysis of test data depends on the applied SRGM. In Fig. 5, the project class, test design technique class, test class and data class constitute the dynamic test processes. Together with the organizational test process class and test management processes class, they construct a multi-layer test process model, making the design and implementation of the workflow management system conform to the standard specifications of the software testing process.

By combining international standards and Chinese national standards in the field of software testing, we execute the tests according to a standardized testing process on the system, which facilitates collaboration among the testers and improves the testing quality. Compared with traditional software testing, the system in this paper abstracts the standardized testing process into independent test nodes, constructs the testing workflow based on the logical relationship between the testing nodes, and realizes the standardized management and status monitoring for the testing process.


Requirement Management>>Requirement Query

Requirement_List

Requirement:

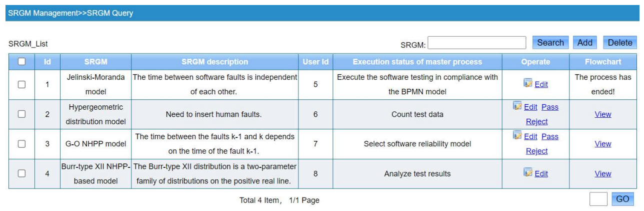
	Id	Requirement	Requirement note	User Id	Execution status of sub-process 1	Execution status of sub-process 2	Operate	Flow chart
☐	1	RSA timing attack program	Import plaintext randomly and output 5000 encrypted ciphertexts.	1	End of recording test data of Part I	Execute the functional tests of Part II	Edit	View
☐	2	RSA timing attack program	Output the private-key before and after the change of each attack cyclically.	2	Execute the functional tests of Part I	Execute the functional tests of Part II	Edit	View
☐	3	Workflow verification system	Establish the business process model of RSA timing attack tasks.	3	Execute the functional tests of Part I	Execute the functional tests of Part II	Edit	View
☐	4	Workflow verification system	Deploy the business process diagram of RSA timing attack tasks.	4	Record the test data of Part I	Record the test data of Part II	Edit Pass Reject	View

Total 4 Item， 1/1 Page

FIGURE 6. Interface of the master process execution.

The interface of the master process execution is shown in Fig. 6. The master process can view the execution status of all the current subprocesses and is also constrained by the execution status of the subprocesses at the same time. The interface of the two subprocess executions is shown in Fig. 7 and Fig. 8. Whether the subprocess execution can pass or not is also constrained by the compliance checking of the master process. Thus, the mutual restriction relationship between different processes is formed to achieve the effect of standardized management for the testing process.

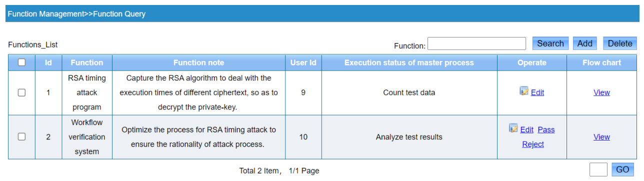
At present, we have conducted six rounds of software testing work of RSA timing attack tasks on the system, effectively improving the quality of the tested software. We can evaluate the system according to the international standard ISO/IEC 30130:2016 [45] and Chinese national standard "Software and Systems engineering – Capabilities



☐	Id	SRGM	SRGM description	User Id	Execution status of master process	Operate	Flowchart
☐	1	Jelinski-Moranda model	The time between software faults is independent of each other.	5	Execute the software testing in compliance with the BPMN model	Edit	The process has ended!
☐	2	Hypergeometric distribution model	Need to insert human faults.	6	Count test data	Edit Pass Reject	View
☐	3	G-O NHPP model	The time between the faults k-1 and k depends on the time of the fault k-1.	7	Select software reliability model	Edit Pass Reject	View
☐	4	Burr-type XII NHPP-based model	The Burr-type XII distribution is a two-parameter family of distributions on the positive real line.	8	Analyze test results	Edit	View

Total 4 Item， 1/1 Page

FIGURE 7. Interface I of the subprocess execution.



☐	Id	Function	Function note	User Id	Execution status of master process	Operate	Flow chart
☐	1	RSA timing attack program	Capture the RSA algorithm to deal with the execution times of different ciphertext, so as to decrypt the private-key.	9	Count test data	Edit	View
☐	2	Workflow verification system	Optimize the process for RSA timing attack to ensure the rationality of attack process.	10	Analyze test results	Edit Pass Reject	View

Total 2 Item， 1/1 Page

FIGURE 8. Interface II of the subprocess execution.

of software testing tools'' (Plan No. 20190837-T-469) that China plans to issue and implement. During the execution of dynamic testing, the capabilities supported by the system include test design, test execution control, monitoring, data validation and verification, and runtime environment management, etc. Furthermore, the system also has some capabilities of test management, such as test management, test asset configuration management, incident management, defect management and defect tracking, and verification and validation report. The above capabilities jointly cover the eight quality characteristics of project, including functional suitability, reliability, usability, performance efficiency, maintainability, portability, compatibility and security, and are applicable to the three levels of granularity – smallest unit, intermediate units and largest unit. In summary, the system basically implements the multi-layer test process model defined in standards [39] and [40], and focuses on the implementation of the dynamic test processes.

IV. RELIABILITY ANALYSIS

A. TEST DATA

The software fault data adopted in this section is derived from the test data of the RSA timing attack program and workflow verification system. For the implementation of BPM, the business process management system (BPMS) based on a workflow engine can provide support for BPM and promote trustworthy process automation management [46]. Therefore, it is necessary to introduce the process management mechanism of the workflow verification system (a kind of BPMS) to make the attack processes execute normally when performing RSA timing attacks, which is conducive to ensuring the success rate of RSA attacks on private keys. The RSA timing attack program is encoded by the C language, while the workflow verification system is encoded by the Java language based on the jBPM. Among them, the test link of the RSA timing attack program belongs to functional tests. The testers only need to consider each function to

TABLE 3. Fault datasets.

Data	No. of faults	Testing time/h	Source
DS1	4	123.1	RSA timing attack program V 1.0
DS2	8	44.1	RSA timing attack program V 2.0
DS3	13	170.5	RSA timing attack program V 3.0
DS4	5	25.9	Workflow verification system V 1.0
DS5	14	175.2	Workflow verification system V 2.0
DS6	10	131.7	Workflow verification system V 3.0

be tested and do not need to consider the internal structure and code of the whole software. However, the test link of the jBPM subsystem supporting the workflow verification system belongs to non-functional tests. The testers need to test the system from the aspects of the modeling subsystem, main control subsystem, log subsystem and web front-end pages. To detect the potential indigenous faults N_0 in the software more accurately, the faults introduced randomly into the software are called inserted faults N_1, and the test ends after the testers detect all N_1 as much as possible. Its purpose is to make it easier to find N_0 during the process of excluding N_1 and to repair the discovered N_0 and N_1, thereby improving the quality of the software product. We conducted three rounds of tests on the RSA timing attack program and workflow verification system respectively, and the generated fault datasets are shown in Table 3. In the numerical experiments, we record the fault datasets DS3 and DS5, and the details are given in Table 4.

The experimental data in this paper adopts two sets of fault datasets in Table 4. By following the instructions of the reliability testing process in the Chinese national standard GB/T 29832.3-2013 ''Reliability of system and software – Part 3: Testing method'' [47], this section estimates the model parameters of the SRGMs used based on the fault data, and then predicts the software reliability and achieves the reliability growth analysis.

B. AN ANALYSIS OF THE HYPERGEOMETRIC DISTRIBUTION MODEL AND THE J-M MODEL

The hypergeometric distribution model finds potential indigenous faults in the software by introducing faults artificially, which reduces the difficulty of software testing. This model adopts the maximum likelihood estimation method to predict the estimated value $\hat{N}_0$ of indigenous faults in the software, which is expressed as follows:

$$\hat{N}_0 = \left[\frac{N_1(n-k)}{k}\right]_{integer} \tag{1}$$

where k represents the number of inserted faults detected, n is the total number of faults detected, and $(n-k)$ is the number of indigenous faults detected. According to the test data in Table 4, the number of indigenous faults in the testing work

TABLE 4. Fault data information.

Tested software	No. of faults	Fault type	Interval time/h	Cumulative time/h
RSA timing attack program (DS3)	1	N_1	9.0	9.0
	2	N_0	2.2	11.2
	3	N_0	12.5	23.7
	4	N_1	5.4	29.1
	5	N_0	5.0	34.1
	6	N_0	17.9	52.0
	7	N_1	18.7	70.7
	8	N_1	12.6	83.3
	9	N_1	10.5	93.8
	10	N_0	29.4	123.2
	11	N_0	3.2	126.4
	12	N_0	15.3	141.7
	13	N_1	28.8	170.5
Workflow verification system (DS5)	1	N_1	1.0	1.0
	2	N_0	13.1	14.1
	3	N_0	0.6	14.7
	4	N_1	4.3	19.0
	5	N_1	1.2	20.2
	6	N_1	5.1	25.3
	7	N_1	36.3	61.6
	8	N_1	5.1	66.7
	9	N_1	12.7	79.4
	10	N_1	11.3	90.7
	11	N_1	17.6	108.3
	12	N_1	2.0	110.3
	13	N_0	16.0	126.3
	14	N_0	48.9	175.2

of the 3rd round and the 5th round are predicted to be 14 and 4, respectively.

With the iteration and update of software version, the number of indigenous faults in the software is increasing. Meanwhile, the fault detection time plays an important role in the analysis of most software reliability models. The reliability assessment method of J-M model is to predict the software quality by fitting the failure law based on historical fault data. The J-M model regards all types of software faults as the same type, that is, the total number of software faults in this paper is equal to the aggregation of indigenous faults and inserted faults. The failure rate expression of the model for random variables is defined as follows:

$$Z(i)=\Phi\,[N-(i-1)] \quad (i=1,2,\cdots,N;\ N>0;\ \Phi>0) \tag{2}$$

where Φ represents the unreliability rate when there is one remaining fault, N represents the estimated total number of potential faults in the software before the start of the testing work, and i represents the number of faults.

The J-M model assumes that the interval time from $(i-1)^{th}$ to i^{th} is a random variable and obeys the exponential distribution with $\Phi[N-(i-1)]$ as the parameter. The reliability function is as follows:

$$R(x_i)=\exp[-\Phi(N-i+1)x_i] \tag{3}$$

where x_{i} represents the observed interval time between faults. The maximum likelihood estimation method is adopted to predict the total number of potential faults N in the software, the estimated value of N can be solved by Formula (4) (represented by $\hat{N}$) as follows:

$$\sum_{i=1}^{n}\frac{1}{(N-i+1)}=\frac{n\sum_{i=1}^{n}x_i}{N\sum_{i=1}^{n}x_i-\sum_{i=1}^{n}(i-1)x_i} \tag{4}$$

The estimated value of Φ can be solved by Formula (5) as follows (represented by $\hat{\Phi}$):

$$\Phi=\frac{n}{N\sum_{i=1}^{n}x_i-\sum_{i=1}^{n}(i-1)x_i} \tag{5}$$

Formulas (4) and (5) are solved in terms of the test data of DS3, and the estimated values of N, Φ are $\hat{N}=19$, $\hat{\Phi}=0.00653$ respectively. When the software life cycle obeys an exponential distribution, the reciprocal of failure rate is called the mean time between failures (MTBF), MTBF $=1/Z(i)$. Hence, it can be predicted that the fault detection time in the 14th stage is approximately 21.9 hours. Similarly, the estimated values of N, Φ for DS5 are $\hat{N}=17$, $\hat{\Phi}=0.00973$, which can predict that the fault detection time in the 15th stage is approximately 25.7 hours. The relationship between the estimated number of initial faults N and MTBF for DS3 and DS5 is shown in Fig. 9 (a) and (b), respectively.

The mean value function $m(t)$ of the J-M model can be expressed as follows:

$$m(t)=\hat{N}(1-e^{-\hat{\Phi}t}) \tag{6}$$

where t is the cumulative fault detection time.

As seen from Table 4, 13 faults are eliminated for the RSA timing attack program in DS3. Both ends of Equation (4) are approximately equal when $N=19$; the total number of faults is predicted to be 19. Hence, it can be estimated that the number of remaining faults in the software is 6. It is known that 12 inserted faults are introduced into the program, and thus, the number of indigenous faults is estimated to be 7.

A total of 14 faults are eliminated for the workflow verification system in DS5, and the J-M model predicts that there are 17 faults in the software; hence, the number of remaining total of faults is 3. It is known that 9 inserted faults are introduced into the system, so the number of indigenous faults is estimated to be 8. However, due to many uncertain factors in the debugging process, the test is usually imperfect, as an imperfect debugging process [48]. The overall

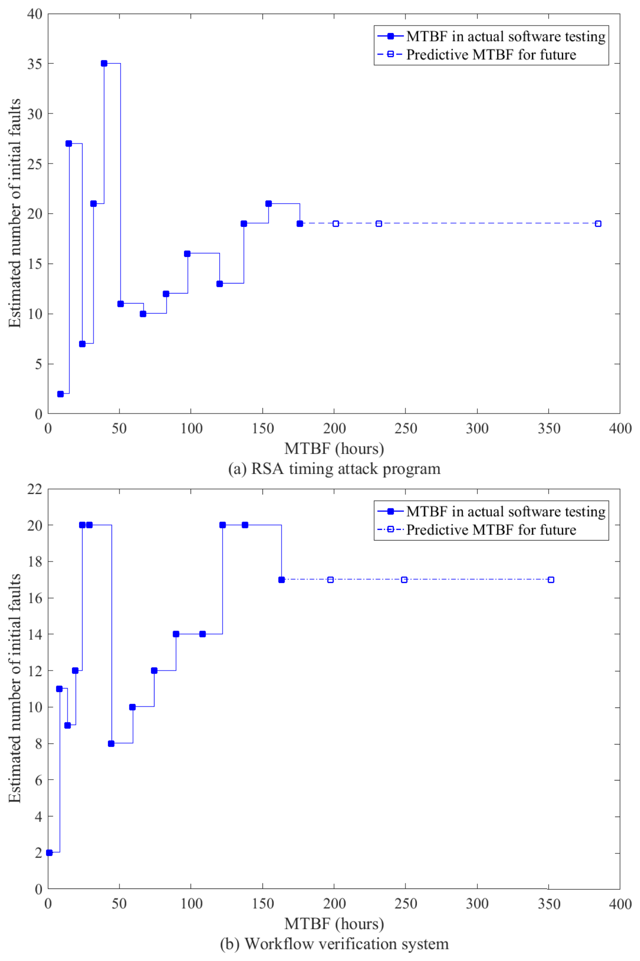


FIGURE 9. Prediction for the total number of software faults (DS3 and DS5).

performance of the imperfect debugging model is better than that of the perfect debugging model, which is a complex random process, considering that the actual debugging will be affected by multiple subjective factors, including debugging ability, skill and psychological change [49]. When repairing the detected fault, we additionally introduce a new fault. Therefore, a total of 10 inserted faults are introduced in DS5, and the number of indigenous faults is estimated to be 7.

C. AN ANALYSIS OF THE G-O NHPP MODEL

The more complex the software, the harder it becomes to test it. In the process of fault detection, the occurrence of faults is random. The J-M model regards the initial number of faults in a software as an unknown fixed constant and does not consider the relationship between the two adjacent interval time, which may reduce the accuracy of reliability assessment. Therefore, considering the actual software testing situation, we adopt the NHPP-based model as the stochastic model of software failure behaviors, and this model has certain flexibility and goodness-of-fit performance [50]. In this paper, the NHPP models also regard all types of software faults as the same type. The G-O NHPP model assumes that the interval time between faults $(i-1)$ and i depends on the time to fault $(i-1)$.

The cumulative number of faults detected at time is an independent incremental process, which obeys the NHPP with mean function $m(t)$, and $m(t)$ is proportional to the cumulative distribution function (CDF) with fault detection time [29,51]. Then, the basic relationship between the probability of n faults detected at time t and $m(t)$ is as follows:

$$\Pr\{N(t)=n\}=\frac{(m(t))^n}{n!}\exp(-m(t)),\quad n=0,1,2,\ldots \tag{7}$$

where $m(t)$ represents the expected value of the total number of software faults at time t, and the faults in a software are limited. Thus, the expression of $m(t)$ is as follows:

$$m(t)=\begin{cases}0, & \mathrm{t}=0\\ a, & \mathrm{t}\rightarrow\infty\end{cases} \tag{8}$$

where a is the expected value of the total number of faults that are finally detected. Assuming that the expected value of the total number of faults in $(t,\ t+\Delta t)$ is proportional to the number of faults not detected at time t, then the expression is as follows:

$$m(t+\Delta t)-m(t)=b(a-m(t))\Delta t+o(\Delta t) \tag{9}$$

where b is the proportionality coefficient. Let $\Delta t\rightarrow 0$ in Formula (9), we can obtain the differential equation as follows:

$$m'(t)=ab-bm(t) \tag{10}$$

Solving Formula (10) in terms of the boundary conditions in Formula (8), we can obtain the equation as follows:

$$m(t)=a(1-e^{-bt}) \tag{11}$$

The fault detection rate function can be obtained from Formula (10) or (11) as follows:

$$\lambda(t)\ =m'(t)=abe^{-bt} \tag{12}$$

We let X_k represent the time interval between the $(k-1)^{th}$ fault and the k^{th} fault, and we let S_k represent the cumulative time taken to detect k faults. Then, we let $S_{k-1}=s$, and the reliability function of X_k is as follows:

$$R_{X_k|S_{k-1}}(x|s)=\exp[-a(e^{-bs}-e^{-b(s+x)})] \tag{13}$$

Suppose that n software faults are detected, and the time sequence is given by $s=\{s_1,s_2,\ldots,s_n\}$. Then, the likelihood function is expressed as follows:

$$f(s_1,s_2,\ldots,s_n)=\exp(-m(s_n))\prod_{k=1}^{n}ab\exp(-bs_k) \tag{14}$$

Therefore, the log likelihood function is written as follows:

$$L(a,b|s)=n\ln a+n\ln b-a(1-e^{-bs_n})-b\sum_{k=1}^{n}s_k \tag{15}$$

By maximizing $L(a, b|s)$ with respect to a and b, we can seek the parameter estimates $\hat{a}$ and $\hat{b}$ of the maximum likelihood function. Taking the partial derivative of Formula (15) with respect to a and b and equating it to zero, we obtain the likelihood equation as follows:

$$\begin{aligned} n/a &= 1 - \exp(-bs_n) \\ n/b &= \sum_{k=1}^{n} s_k + as_n \exp(-bs_n) \end{aligned} \tag{16}$$

We can obtain $\hat{a} = 24.38$ and $\hat{b} = 0.00447$ by solving Formula (16) in terms of DS3. Therefore, the expected value of the total number of faults in $[0, t]$ is $m(t) = 24.38 \cdot (1 - e^{-0.00447 \cdot t})$. Similarly, $\hat{a} = 17.55$ and $\hat{b} = 0.00912$ can be solved in terms of DS5, and the mean value function is $m(t) = 17.55 \cdot (1 - e^{-0.00912 \cdot t})$. The fitted curves of the estimated mean value function and actual fault data of these two rounds of tests are shown in Fig. 10 (a) and (b), respectively.

After obtaining the estimates of the unknown parameters, we can then obtain the maximum likelihood estimate of the mean value function $m(t)$. The confidence interval for $N(t)$ based on the Poisson distribution can be approximately obtained, where the 90% upper and lower confidence bounds for the $N(t)$ process are shown in Formula (17) as follows:

$$\begin{aligned} \hat{m}(t)_{upper} &= m(t) + z_{\alpha/2}\sqrt{m(t)} \\ \hat{m}(t)_{lower} &= m(t) - z_{\alpha/2}\sqrt{m(t)} \end{aligned} \tag{17}$$

where $\hat{m}(t)_{upper}$ and $\hat{m}(t)_{lower}$ are the upper and lower confidence bounds functions, respectively, and $z_{\alpha/2}$ is the $100(1 - \alpha)$ percentile that follows the standard normal distribution [52]. When $\alpha = 0.1$, it represents the 90% confidence interval, that is, the value of z is 1.64. Fig. 10 also shows the 90% confidence interval for the $N(t)$ process.

Substituting the estimated values of a and b into Formula (13), we can obtain the following software reliability function as follows:

$$\begin{aligned} &R_{X_{14}|S_{13}}(x|170.5) \\ &\quad = \exp[-24.38 \cdot (e^{-0.00447 \cdot 170.5} - e^{-0.00447 \cdot (170.5+x)})] \end{aligned}$$

In the 3rd round of the test, the G-O NHPP model predicts that 24 faults exist in the software. At present, 13 faults are detected, and the remaining 11 faults remain to be discovered, which achieves the purpose of evaluating software quality. The reliability functions for the J-M model and the G-O NHPP model, based on $n = 8, n = 13$ and $n = 8, n = 14$, are shown in Fig. 11 (a) and (b), respectively.

In Fig. 11, the greater the value of n in the early test stage, the higher the software reliability predicted by the J-M model; however, in the whole test stage, the greater the value of n, the higher the software reliability predicted by G-O NHPP model. This is because with the increase of the value of n, the number of remaining faults is decreasing, and the reliability will be improved accordingly. Fig. 11 also indicates that the G-O NHPP model produces a more conservative predictive effect than the J-M model.

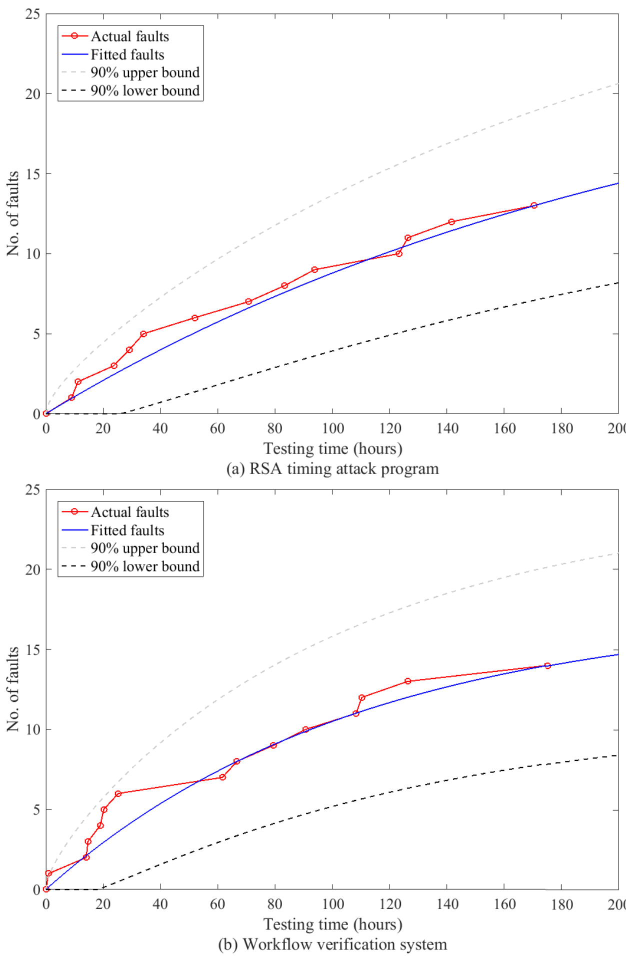


FIGURE 10. The actual and fitted values of the total number of faults and the 90% confidence intervals for the counting process (DS3 and DS5).

D. AN ANALYSIS OF THE BURR-TYPE XII NHPP-BASED MODEL

The Burr distribution [53] contains 12 kinds of distribution functions, which is a family of continuous probability distributions proposed by Burr in the study of differential equations. This distribution has been widely used in many fields, such as applied statistics, quality control and reliability. The Burr distribution presented some properties and theories of continuous functions and considered the fitting problems of functions. Burr [53] introduced a family of CDFs and satisfied the following differential equations as follows:

$$\frac{dF(x)}{dx} = F(x)(1 - F(x))g(x, F(x)) \tag{18}$$

where $g(x, F(x))$ is positive for $0 \leq F(x) \leq 1$. It can be seen in Formula (18) that $F(x)$ has the nondecreasing property, and $dF(x)/dx$ is zero at $F(x) = 0$ or $F(x) = 1$. In the case of $g(x, F(x)) = g(x)$, solving Formula (18) can be obtained as follows:

$$F(x) = \frac{1}{[e^{-\int g(x)dx} + 1]} \tag{19}$$

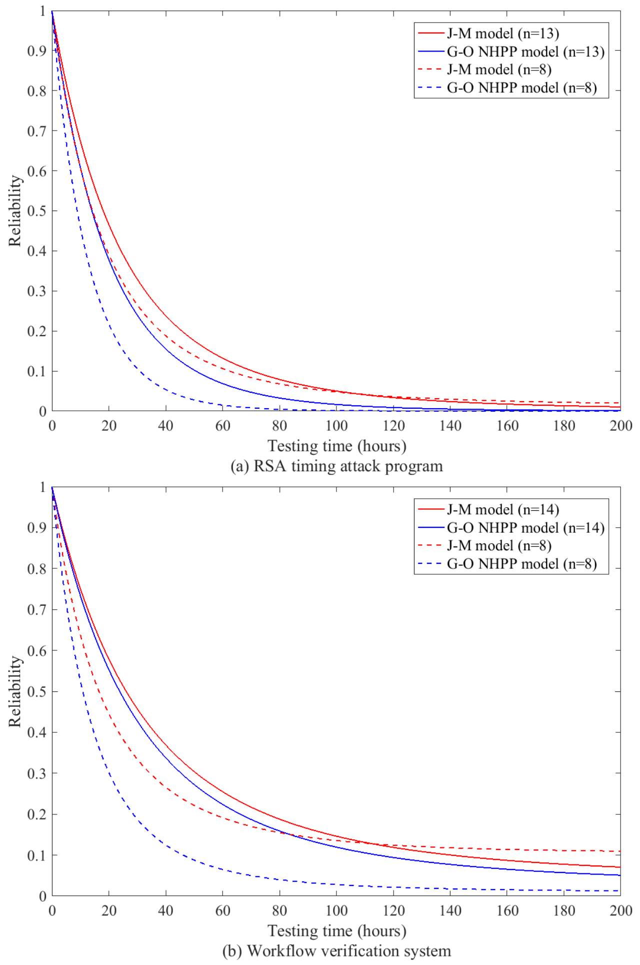


FIGURE 11. Predictive software reliability assessment with the J-M and G-O NHPP models (DS3 and DS5).

It should be noted that the selection of $g(x)$ should make $F(x)$ monotonically increase from 0 to 1 within the specified time x. The Burr-type XII distribution is one of 12 Burr-type distributions derived by Burr and is widely used to describe the distribution of software fault detection time [51]. The Burr-type XII distribution is a two-parameter family of distributions on the positive real line, and its CDF is as follows:

$$F(x) = 1 - (1 + x^a)^{-b} \; (x > 0, a > 0, b > 0) \tag{20}$$

where a and b are the shape parameters. In this paper, we apply the Burr-type XII distribution by introducing an additional scale parameter d to represent the mean value function $m(t)$ of an NHPP-based SRGM, as follows:

$$m(t) = kF(t) = k[1 - (1 + (t/d)^a)^{-b}] \; (k > 0) \tag{21}$$

From the property of the Poisson distribution, we obtain the equation as follows:

$$m(t) = \int_0^t \lambda(x)dx \tag{22}$$

where $\lambda(x)$ is a continuous function called the intensity function. We suppose that n faults are detected in the test stage, and the time sequence of fault detection is $T = \{t_1, t_2, \ldots, t_n\}$. Therefore, the likelihood function and log likelihood function with respect to T are expressed as follows:

$$L(k, d, a, b|T) = \exp(-m(t_n)) \prod_{i=1}^{n} \lambda(t_i) \tag{23}$$

$$\ln L(k, d, a, b|T) = \sum_{i=1}^{n} \ln \lambda(t_i) - m(t_n) \tag{24}$$

By substituting $m(t)$ and $\lambda(t)$ in line with Burr-type XII distribution into Formula (24), it can be obtained:

$$\begin{aligned} &\ln L(k, d, a, b|T) \\ &= k[1 + (t_n/d)^a]^{-b} - k \\ &\quad + \sum_{i=1}^{n} \left\{ \ln(kab/d) + \ln (t_i/d)^{(a-1)} - \ln [1 + (t_i/d)^a]^{(b+1)} \right\} \end{aligned} \tag{25}$$

Taking the partial derivatives of Formula (25) with respect to k, d, a and b, and equating them to zero, respectively, and we can obtain:

$$\begin{cases} \dfrac{\partial L}{\partial k} = \dfrac{n}{k} + \left[1 + (t_n/d)^a\right]^{-b} - 1 = 0 \\ \dfrac{\partial L}{\partial d} = -abk\left[1 + (t_n/d)^a\right]^{-b-1} (t_n/d)^{a-1} (-t_n/d^2) \\ + \displaystyle\sum_{i=1}^{n} \left[-\frac{1}{d} - \frac{a(b+1)(t_i/d)^{a-1}(-t_i/d^2)}{1 + (t_i/d)^a} - \frac{a-1}{d} \right] = 0 \\ \dfrac{\partial L}{\partial a} = -bk\left[1 + (t_n/d)^a\right]^{-b-1} (t_n/d)^a \ln(t_n/d) \\ + \displaystyle\sum_{i=1}^{n} \left[\frac{1}{a} - \frac{(b+1)(t_i/d)^a \ln(t_i/d)}{1 + (t_i/d)^a} + \ln(t_i/d) \right] = 0 \\ \dfrac{\partial L}{\partial b} = -k\left[1 + (t_n/d)^a\right]^{-b} \ln\left(1 + (t_n/d)^a\right) \\ + \displaystyle\sum_{i=1}^{n} \left[\frac{1}{b} - \ln\left(1 + (t_i/d)^a\right) \right] = 0 \end{cases} \tag{26}$$

When solving the problem for maximum likelihood function, we can use the gradient rise method to solve it step by step, so as to obtain the maximum likelihood function value and model parameter values. The calculation process of the gradient rise method is to iteratively solve the maximum value along the gradient rise direction. The main solution process is as follows:

$$\begin{cases} k_{m+1} = k_m + \eta \dfrac{\partial L\,(k_m, d_m, a_m, b_m)}{\partial k_m} \\ d_{m+1} = d_m + \eta \dfrac{\partial L\,(k_m, d_m, a_m, b_m)}{\partial d_m} \\ a_{m+1} = a_m + \eta \dfrac{\partial L\,(k_m, d_m, a_m, b_m)}{\partial a_m} \\ b_{m+1} = b_m + \eta \dfrac{\partial L\,(k_m, d_m, a_m, b_m)}{\partial b_m} \end{cases} \tag{27}$$

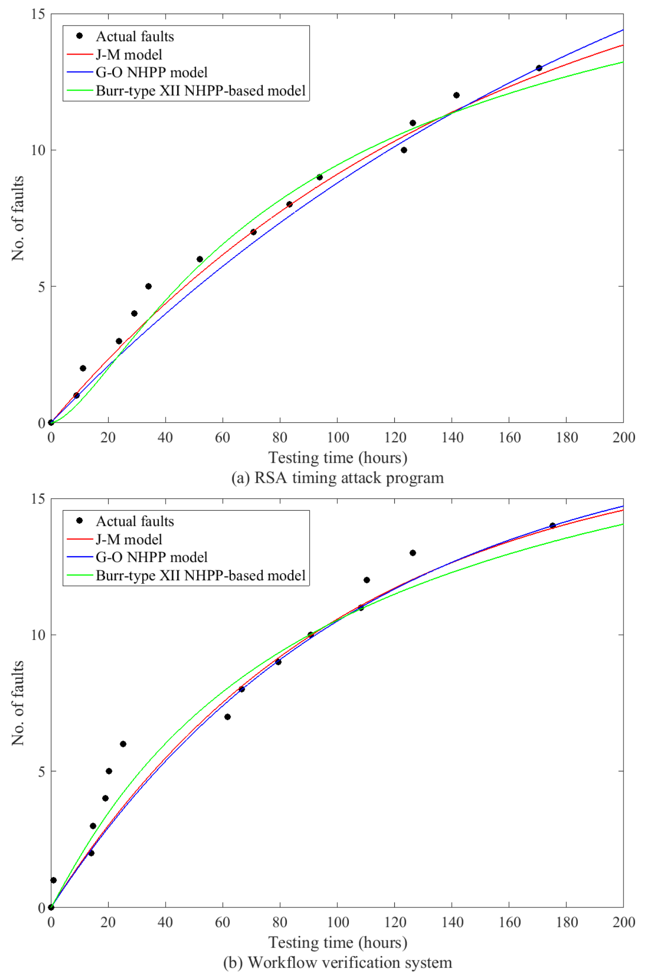


FIGURE 12. The observed and estimated cumulative number of faults (DS3 and DS5).

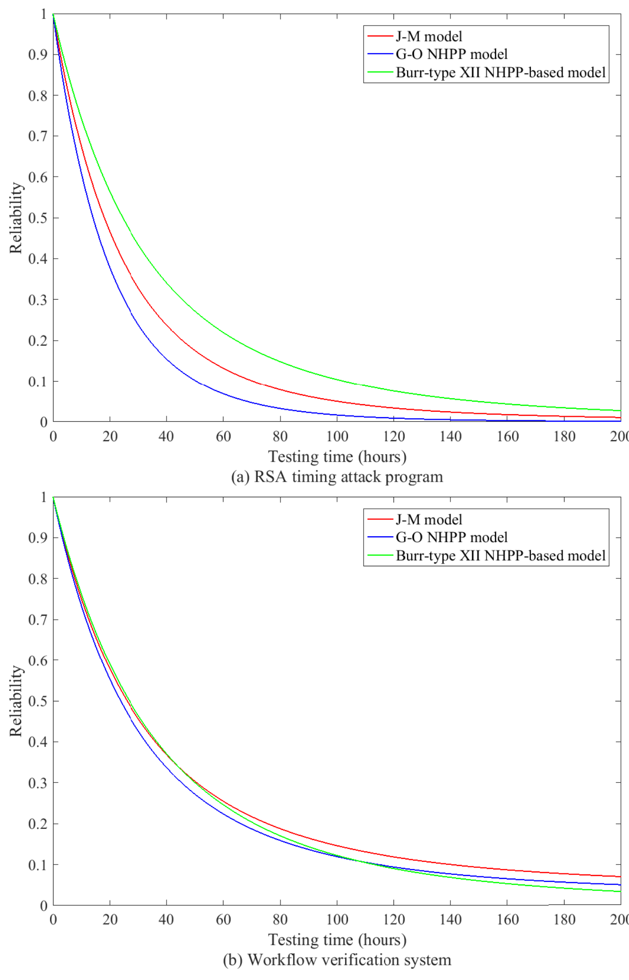


FIGURE 13. Predictive software reliability assessment with multiple SRGMs (DS3 and DS5).

where m is the number of gradient iterations; k_m, d_m, a_m and b_m are the updated values of the parameters at the m^{th} iteration; $\partial L\left(k_m, d_m, a_m, b_m\right)/\partial k_m$ is the gradient of L with respect to k at the $(m+1)^{th}$ iteration; $\eta(\eta > 0)$ is the search step in the gradient direction, and the selection of the step determines the convergence property and convergence speed of the gradient method. In the standardization workflow management system of software testing processes, the optimization technique of gradient method is adopted to complete the numerical searches by controlling the number of iterations and setting the reasonable start point and step. The purpose of combining the maximum likelihood function with the gradient method is to find the local maximum value of the likelihood function, and optimize the search result as much as possible to make it closer to the maximum value of the likelihood function. In addition, when the difference value between the likelihood function and the last iterative likelihood function tends to converge, the search stops, so the model parameter values $\hat{k}$, $\hat{d}$, $\hat{a}$ and $\hat{b}$ corresponding to the maximum likelihood function can be obtained. From the test data of DS3 and DS5, $\hat{k} = 24.79$, $\hat{d} = 42.10257$, $\hat{a} = 1.54886$, $\hat{b} = 0.30515$ and $\hat{k} = 23.76$, $\hat{d} = 61.94323$, $\hat{a} = 1.06100$, $\hat{b} = 0.59823$ can be solved, respectively. Fig. 12 plots the fitted curves of the mean value functions for the J-M model, the G-O NHPP model and the Burr-type XII NHPP-based model.

Finally, we quantitatively evaluate the software reliability with the Burr-type XII NHPP-based model and compare it with other SRGMs. The conditional probability that the i^{th} fault is not detected between $(t, t{+}x]$ $(x \geq 0)$ on the condition that the $(i-1)^{th}$ has detected at testing time t is given as follows:

$$\begin{aligned} R(x|t) &= \Pr\{N(t+x) - N(t) = 0 | N(t) = n\} \\ &= \exp\{-[m(t+x) - m(t)]\} \end{aligned} \tag{28}$$

where $N(t)$ is a counting process and n is the cumulative number of faults detected at time t. $m(t)$ given in Formula (21) is substituted by Formula (28) to obtain the software reliability for the Burr-type XII NHPP-based model, as follows:

$$R(x|t) = \exp\{k[(1 + ((t+x)/d)^a)^{-b} - (1 + (t/d)^a)^{-b}]\} \tag{29}$$

Fig. 13 (a) and (b) show the software reliability assessment with the J-M model, G-O NHPP model and Burr-type XII

TABLE 5. Assessment results of different SRGMs.

Criterion/Model	Dataset	Parameter setting	MSE	SAE	MEOP	Variation	R^2
J-M model	DS3	$\hat{N} = 19, \hat{\Phi} = 0.00653$	**0.27305**	**5.41747**	**0.27837**	**0.42776**	**0.98050**
G-O NHPP model	DS3	$\hat{a} = 24.38, \hat{b} = 0.00447$	0.51658	7.74876	0.40404	0.48669	0.96090
Burr-type XII NHPP-based model	DS3	$\hat{k} = 24.79, \hat{d} = 42.10257,$ $\hat{a} = 1.54886, \hat{b} = 0.30515$	0.41194	7.21598	0.42428	0.58296	0.97058
J-M model	DS5	$\hat{N} = 17, \hat{\Phi} = 0.00973$	0.98303	10.05115	0.37502	0.85674	0.93951
G-O NHPP model	DS5	$\hat{a} = 17.55, \hat{b} = 0.00912$	1.05353	**9.98699**	**0.34141**	0.85598	0.93458
Burr-type XII NHPP-based model	DS5	$\hat{k} = 23.76, \hat{d} = 61.94323,$ $\hat{a} = 1.06100, \hat{b} = 0.59823$	**0.79929**	10.35710	0.60383	**0.82868**	**0.95081**

NHPP-based model on DS3 and DS5, respectively. It can be observed that the predictive reliability values of Burr-type XII NHPP-based model is higher than that of the other two models, but three models show similar reliability values on DS5.

V. MODEL EVALUATION CRITERIA

A. MODEL COMPARISON CRITERIA

Different SRGMs have different fitting effect for the same fault dataset. Since DS3 and DS5 are obtained as fault counts, thus we employ the following goodness-of-fit criteria to examine and compare the predictive power of all selected models. These criteria are described as follows:

1) Mean Square Error (MSE)

The MSE [54] is used to describe the average distance between the cumulative number of faults actually observed and the predictive value of the faults, which is defined as follows:

$$\text{MSE} = \frac{\sum_{i=1}^{n} (\hat{m}(t_i) - y_i)^2}{n} \tag{30}$$

2) Sum of Absolute Error (SAE)

The SAE [54] is used to describe the distance between the predicted number of faults and the observed data, which is defined as follows:

$$\text{SAE} = \sum_{i=1}^{n} \left|\hat{m}(t_i) - y_i\right| \tag{31}$$

3) Mean Error of Prediction (MEOP)

The MEOP [55] is usually used to compare the predictive power, which is defined as follows:

$$\text{MEOP} = \frac{\sum_{i=k}^{n} \left|\hat{m}(t_i) - y_i\right|}{n-k+1} \tag{32}$$

4) Variation

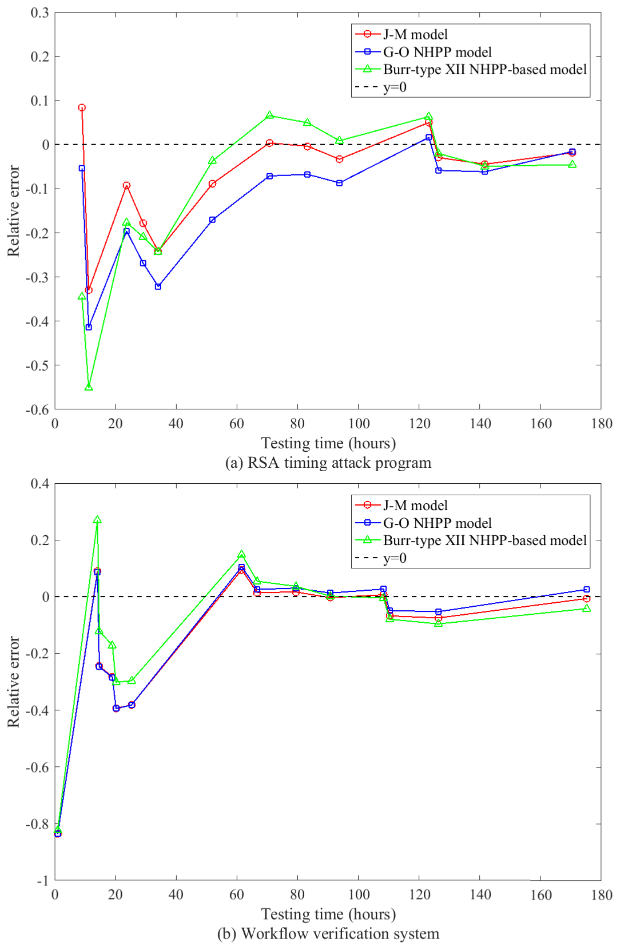
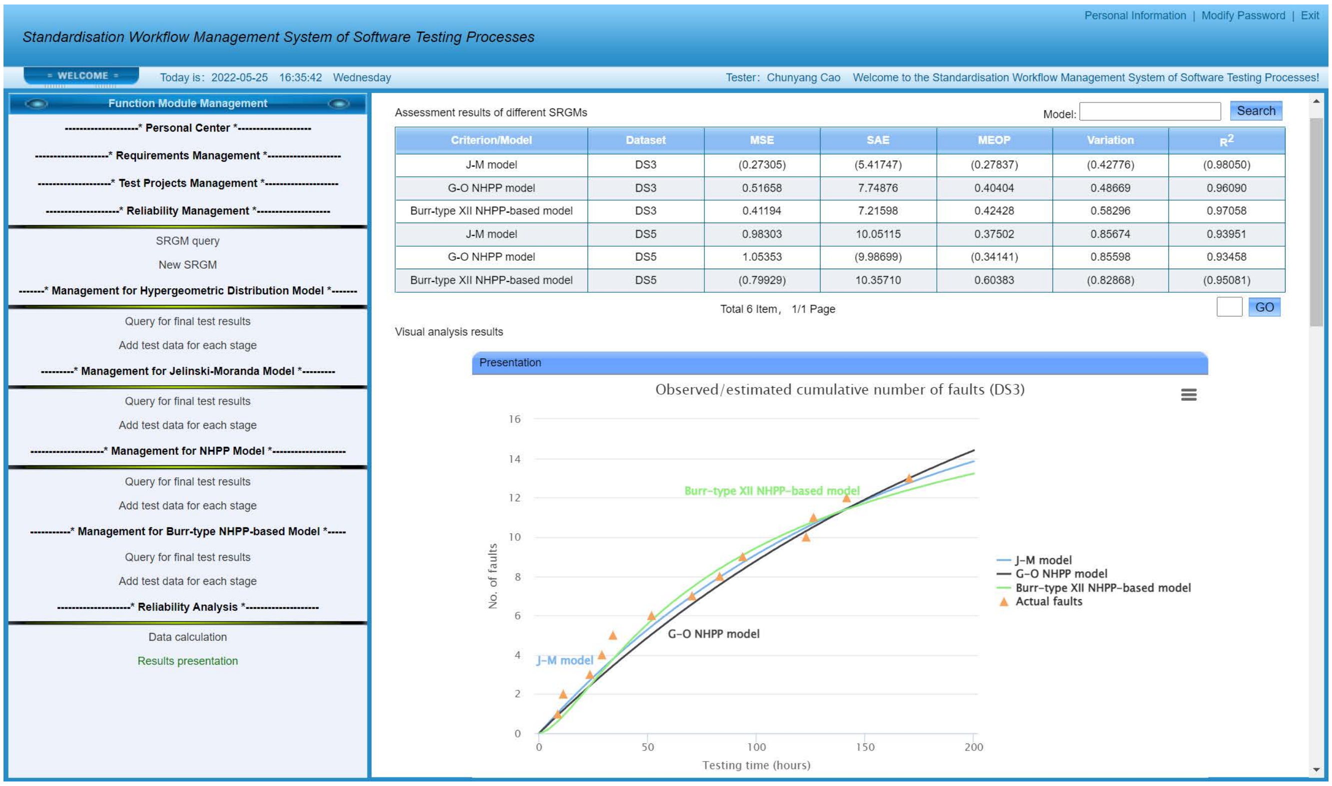


FIGURE 14. Relative error of fault detection based on different models (DS3 and DS5).

The Variation [56] is the standard deviation of prediction bias, which is defined as follows:

$$\text{Variation} = \sqrt{\frac{\sum_{i=1}^{n} \left((\hat{m}(t_i) - y_i) - \text{Bias}\right)^2}{n-1}} \tag{33}$$


Standardisation Workflow Management System of Software Testing Processes

Personal Information | Modify Password | Exit

Today is: 2022-05-25 16:35:42 Wednesday

Tester: Chunyang Cao Welcome to the Standardisation Workflow Management System of Software Testing Processes!

Assessment results of different SRGMs

Criterion/Model	Dataset	MSE	SAE	MEOP	Variation	R^2
J-M model	DS3	(0.27305)	(5.41747)	(0.27837)	(0.42776)	(0.98050)
G-O NHPP model	DS3	0.51658	7.74876	0.40404	0.48669	0.96090
Burr-type XII NHPP-based model	DS3	0.41194	7.21598	0.42428	0.58296	0.97058
J-M model	DS5	0.98303	10.05115	0.37502	0.85674	0.93951
G-O NHPP model	DS5	1.05353	(9.98699)	(0.34141)	0.85598	0.93458
Burr-type XII NHPP-based model	DS5	(0.79929)	10.35710	0.60383	(0.82868)	(0.95081)

Total 6 Item, 1/1 Page

Visual analysis results

WELCOME
Function Module Management
--------------------* Personal Center *--------------------
--------------------* Requirements Management *--------------------
--------------------* Test Projects Management *--------------------
--------------------* Reliability Management *--------------------
SRGM query
New SRGM
-------* Management for Hypergeometric Distribution Model *-------
Query for final test results
Add test data for each stage
---------* Management for Jelinski-Moranda Model *---------
Query for final test results
Add test data for each stage
--------------------* Management for NHPP Model *--------------------
Query for final test results
Add test data for each stage
----------* Management for Burr-type NHPP-based Model *-----
Query for final test results
Add test data for each stage
--------------------* Reliability Analysis *--------------------
Data calculation
Results presentation
Model:
Search
GO
Presentation
Observed/estimated cumulative number of faults (DS3)
No. of faults
Testing time (hours)
Burr-type XII NHPP-based model
G-O NHPP model
J-M model
— J-M model
— G-O NHPP model
— Burr-type XII NHPP-based model
▲ Actual faults

FIGURE 15. A jBPM-based standardized test system for model assessment.

The expression of bias is as follows:

$$\text{Bias} = \sum_{i=1}^{n} \left[\frac{\hat{m}(t_i) - y_i}{n} \right] \tag{34}$$

5) Correlation Index of Regression Curve (R^2)

The R^2 [57] can measure the percentage of variations explained by the model and examine the fitting power of SRGMs, which is expressed as follows:

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{m}(t_i))^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2} \tag{35}$$

where $\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i$.

In the above formulas, n is the number of fault sample data, k is the selected fault number, y_i is the observed cumulative number of faults at time t_i, and $\hat{m}(t_i)$ is the estimated cumulative number of faults by SRGM at time t_i. Furthermore, the assessment method of the hypergeometric distribution model discussed in Section IV is different from the other three models. The hypergeometric distribution model employs cumulative number of faults and fault type to predict the total number of software faults, while other models mentioned in this paper employ the interval time and cumulative number of faults to predict the total number of software faults at a certain time in the future. In order to intuitively present the predictive power of different SRGMs over time, Table 5 lists the assessment results of the J-M model, G-O NHPP model and Burr-type XII NHPP-based model on DS3 and DS5.

For the MSE, SAE, MEOP and Variation mentioned above, a smaller indicator value indicates a stronger fitting power. However, a larger value of R^2 indicates that the fitting power for the data is better. For a more accurate model comparison, the fitting results of the two datasets by different SRGMs are presented in Table 5, where the bold font marks the optimal indicator value in each dataset. It can be observed that the fitting effect of the J-M model on the test data of DS3 is better than that of the G-O NHPP model and Burr-type XII NHPP-based model. However, the Burr-type XII NHPP-based model has a better fitting effect on the test data of DS5. By contrasting the model comparison criteria, the testers can select a more suitable SRGM for the test data analysis in terms of different datasets.

B. MODEL PREDICTIVE PERFORMANCE CRITERIA

The model predictive performance criteria is used to verify the predictive performance of multiple SRGMs on different fault datasets, which provides guidance for evaluating the software reliability. Relative error (RE) is used to evaluate the predictive performance of these models during fault detection [58], as follows:

$$\text{RE} = \frac{\hat{m}(t_{i+1}) - m_{t_{i+1}}}{m_{t_{i+1}}} \tag{36}$$

where $m_{t_{i+1}}$ represents the cumulative number of faults detected at time t_{i+1}, and $\hat{m}(t_{i+1})$ represents the fault data detected up to time t_i to estimate the parameter values of the model and calculate the number of faults detected at time t_{i+1}. The RE is a one-step prediction, which is used to measure the predictive performance of SRGMs. If RE is closer to 0, the predictive performance of the model is better. If RE is greater than 0, the number of fault detections in the software is overestimated. However, if RE is less than 0, the number of fault detections in the software is underestimated. Fig. 14 plots the RE prediction curves of different models on the 2 fault datasets.

It can be seen from Fig. 14 that the three models have good fit and prediction performance in fault detection, which demonstrates that the modeling and management of testing processes are more accurate. On the whole, the models begin to converge rapidly and gradually approach the curve $y = 0$ in the second half of the fault datasets, indicating that their predictive performance is improving.

C. MODEL ASSESSMENT OF STANDARDIZATION WORKFLOW MANAGEMENT SYSTEM

The reliability management module of the standardization workflow management system of software testing processes is developed to evaluate the fitting effect and predictive performance of SRGMs. The models implemented in the system are the hypergeometric distribution model, the J-M model, the G-O NHPP model and the Burr-type XII NHPP-based model, and the detailed analysis of these models is given in sections IV and V. The input of the reliability management module is fault datasets DS3 and DS5, and the output is the reliability analysis results of SRGMs. The standardization workflow management system can not only regulate the software testing processes, but also have great significance in improving the fault detection rate and software quality. Fig. 15 presents a screen shot of the system. After the analysis of test data is completed, the system gives the model assessment results of different SRGMs. In addition, it also can draw visual graphs of the estimated mean value functions, the software reliability functions and the RE prediction curves for all the SRGMs on different fault datasets.

VI. CONCLUSION

This paper proposes a standardization workflow technology to regulate the software testing processes, and constructs a workflow management system to realize the efficient management for software testing processes. Taking the RSA timing attack program and workflow verification system as the test objects, we focus on combining the standardized system in the field of software testing with the actual testing processes, as well as extracting and summarizing the standards of the testing work to instruct the testing processes. Throughout numerical experiments with two fault datasets observed in actual software testing, we have employed multiple SRGMs to evaluate the software reliability and compared these SRGMs in fitting power and predictive performance. Specifically, the J-M model provides lower MSE, SAE, MEOP, Variation and higher R^2 on DS3. However, the Burr-type XII NHPP-based model provides a better fitting effect on DS5. Therefore, testers can select a more appropriate SRGM from the model evaluation results for reliability analysis. In software testing, the idea of standardized process and the construction of workflow management system are interrelated and mutually reinforcing. The construction of the system should consider the current testing technology and testing standards comprehensively, and improve the testing process combined with the actual situation of software development. For one thing, the formulation of standardized testing process can accurately and efficiently regulate the design and implementation of workflow management system's functional modules; for another, the iterative upgrade of workflow management system also promotes a higher degree for process standardization. These two aspects make the implementation of software testing projects more scientific and standardized, and ensure the reasonable execution of the testing processes. The proposed standardization workflow technology can provide instructive significance for future related test projects, increase the probability of finding software defects, and thus improving the software quality.

In future research works, we plan to combine the standardization workflow technology with the EM algorithm in software reliability assessment to obtain more accurate parameter estimation, thus making reliability analysis results more reliable and software testing processes more standardized. Furthermore, the Activiti engine covers workflow, service collaboration and other areas with better flexibility, scalability and execution efficiency, and is widely used in the field of BPM. We need to fully understand the software architecture of jBPM, and hope to employ the Activiti engine to implement the process management of software testing in the future. In the process of test management, we will pre-process the logs of software testing processes generated by the standardization workflow management system, and then use Petri nets to carry out process acquisition, process decomposition, process combination and process verification on the logs. Therefore, this will make the modeling of standardization patterns for software testing processes more trustworthy.

ACKNOWLEDGMENT

The authors would like to thank the support and help in scientific research offered by the Key Laboratory of Images & Graphics Intelligent Processing of State Ethnic Affairs Commission, North Minzu University (IGIPLaboratory) and also would like to thank the anonymous reviewers and editors for their suggestions.

REFERENCES

[1] K. Sneha and G. M. Malle, "Research on software testing techniques and software automation testing tools," in *Proc. Int. Conf. Energy, Commun., Data Anal. Soft Comput. (ICECDS)*, Chennai, India, Aug. 2017, pp. 77–81.

[2] F. Okezie, I. Odun-Ayo, and S. Bogle, "A critical analysis of software testing tools," *J. Phys., Conf. Ser.*, vol. 1378, no. 4, Dec. 2019, Art. no. 042030.

[3] Z. Sun, C. Hu, C. Li, and L. Wu, "Domain ontology construction and evaluation for the entire process of software testing," *IEEE Access*, vol. 8, pp. 205374–205385, 2020.

[4] K.-Y. Cai, C.-A. Sun, and C. Nie, "Software reliability assessment: A software cybernetics perspective," *Scientia Sinica Inf.*, vol. 49, no. 11, pp. 1528–1531, Nov. 2019.

[5] A. M. Mamathashree, K. Remya, and B. J. S. Kumar, "Fault analysis detection in public key cryptosystems (RSA)," in *Proc. Int. Conf. Commun. Signal Process. (ICCSP)*, Chennai, India, Apr. 2017, pp. 505–508.

[6] P. C. Kocher, "Timing attacks on implementations of Diffie–Hellman, RSA, DSS, and other systems," in *Proc. 16th Annu. Int. Cryptol. Conf. Adv. Cryptol.*, Santa Barbara, CA, USA, Aug. 1996, pp. 104–113.

[7] C. Li, Q. Han, B. Lei, H. Liu, C. Liu, and Y. He, "Timing attacks in single-chip microcomputer through workflow verification," in *Proc. 8th Int. Conf. Dependable Syst. Their Appl. (DSA)*, Yinchuan, China, Aug. 2021, pp. 716–721.

[8] S. Ding, T. Li, and P. Yu, "Research on software testing standardization at home and abroad," *China Standardization*, vol. 11, pp. 109–113, Jun. 2019.

[9] S. Masuda, Y. Nishi, and K. Suzuki, "Complex software testing analysis using international standards," in *Proc. IEEE Int. Conf. Softw. Test., Verification Validation Workshops (ICSTW)*, Porto, Portugal, Oct. 2020, pp. 241–246.

[10] P. Shen, X. Ding, W. Ren, and C. Yang, "Research on software quality assurance based on software quality standards and technology management," in *Proc. 19th IEEE/ACIS Int. Conf. Softw. Eng., Artif. Intell., Netw. Parallel/Distrib. Comput. (SNPD)*, Busan, South Korea, Jun. 2018, pp. 385–390.

[11] Y. Zhang, Z. Liu, Q. Han, and W. Zhang, "Development of system life cycle processes standardization and future evolution analysis," in *Proc. 8th Int. Conf. Dependable Syst. Their Appl. (DSA)*, Yinchuan, China, Aug. 2021, pp. 240–246.

[12] H. Jiang, "Research and implementation of financial approval system based on jBPM engine," in *Proc. IEEE 3rd Adv. Inf. Technol., Electron. Autom. Control Conf. (IAEAC)*, Chongqing, China, Oct. 2018, pp. 394–399.

[13] X. Junying, L. Yan, and L. Ziyu, "Research and implementation on communication organization workflow management system based on service-oriented architecture," in *Proc. IEEE Int. Conf. Artif. Intell. Inf. Syst. (ICAIIS)*, Dalian, China, Mar. 2020, pp. 548–551.

[14] B. Mao, W. Hu, A. Althoff, J. Matai, Y. Tai, D. Mu, T. Sherwood, and R. Kastner, "Quantitative analysis of timing channel security in cryptographic hardware design," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 9, pp. 1719–1732, Sep. 2018.

[15] B. Mao, W. Hu, D. Mu, H. Zhang, Y. Tai, and L. Hong, "Quantitative analysis of information leakage through hardware RSA timing channel based on entropy theory," *Chin. J. Comput.*, vol. 41, no. 2, pp. 426–438, Feb. 2018.

[16] W. Schindler, "A timing attack against RSA with the Chinese remainder theorem," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst.*, Lecture Notes in Computer Science, 2000, pp. 109–124.

[17] S. Guo, T. Wang, and X. Zhao, *Side-Channel Analysis in Cryptography: Principles, Methodologies and Modern Applications*. Beijing, China: Science Press, 2014, pp. 60–61.

[18] B. Mao, D. Mu, W. Hu, H. Zhang, and M. Qin, "Detection and quantitative analysis of timing channel in RSA cryptographic core," *Comput. Eng. Appl.*, vol. 53, no. 10, pp. 8–12, May 2017.

[19] A. D. Dwivedi, R. Singh, U. Ghosh, R. R. Mukkamala, A. Tolba, and O. Said, "Privacy preserving authentication system based on non-interactive zero knowledge proof suitable for Internet of Things," *J. Ambient Intell. Hum. Comput.*, pp.1–11, Sep. 2021.

[20] B. Hettwer, S. Gehrer, and T. Güneysu, "Applications of machine learning techniques in side-channel attacks: A survey," *J. Cryptograph. Eng.*, vol. 10, no. 2, pp. 135–162, Jun. 2020.

[21] S. Jin, S. Kim, H. Kim, and S. Hong, "Recent advances in deep learning-based side-channel analysis," *ETRI J.*, vol. 42, no. 2, pp. 292–304, Apr. 2020.

[22] W. Driel, J. W. Bikker, and M. Tijink, "System software reliability," in *Proc. 21st Int. Conf. Thermal, Mech. Multi-Phys. Simulation Exp. Microelectron. Microsyst. (EuroSimE)*, Kraków, Poland, Jul. 2020, pp. 1–5.

[23] T. Yaghoobi, "Selection of optimal software reliability growth model using a diversity index," *Soft Comput.*, vol. 25, no. 7, pp. 5339–5353, Jan. 2021.

[24] Y. Xu and D. Pi, "Complex software reliability allocation based on hybrid intelligent optimization algorithm," *Ruan Jian Xue Bao/J. Softw.*, vol. 29, pp. 2632–2648, Sep. 2018.

[25] Y. Tohma, K. Tokunaga, S. Nagase, and Y. Murata, "Structural approach to the estimation of the number of residual software faults based on the hyper-geometric distribution," *IEEE Trans. Softw. Eng.*, vol. 15, no. 3, pp. 345–355, Mar. 1989.

[26] Y. Tohma, H. Yamano, M. Ohba, and R. Jacoby, "The estimation of parameters of the hypergeometric distribution and its application to the software reliability growth model," *IEEE Trans. Softw. Eng.*, vol. 17, no. 5, pp. 483–489, May 1991.

[27] Z. Jelinski and P. Moranda, "Software reliability research," in *Statistical Computer Performance Evaluation*. New York, NY, USA: Academic, 1972, pp. 465–484.

[28] J. Liu, Y. Hu, X. Wang, and M. Xu, "Novel nonlinear least square parameter estimation method for Jelinski–Moranda model," *Xitong Fangzhen Xuebao/J. Syst. Simul.*, vol. 20, no. 18, pp. 4791–4794, Sep. 2008.

[29] A. L. Goel and K. Okumoto, "Time-dependent error-detection rate model for software reliability and other performance measures," *IEEE Trans. Rel.*, vol. R-28, no. 3, pp. 206–211, Aug. 1979.

[30] H. Okamura and T. Dohi, "Application of EM algorithm to NHPP-based software reliability assessment with generalized failure count data," *Mathematics*, vol. 9, no. 9, pp. 1–18, Apr. 2021.

[31] S. Li, "A useful parametric family to characterize NHPP-based software reliability models," in *Proc. 51st Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Taipei, Taiwan, Jun. 2021, pp. 23–24.

[32] H. Okamura, Y. Watanabe, T. Dohi, and S. Osaki, "An estimation of software reliability models based on EM algorithm," *Electron. Commun. Jpn. II, Fundam. Electron. Sci.*, vol. 86, no. 6, pp. 29–37, Jun. 2003.

[33] H. Okamura and T. Dohi, "SRATS: Software reliability assessment tool on spreadsheet," in *Proc. 24th Int. Symp. Softw. Rel. Eng. (ISSRE)*, Pasadena, CA, USA, Nov. 2013, pp. 100–117.

[34] H. Okamura and T. Dohi, "Phase-type software reliability model: Parameter estimation algorithms with grouped data," *Ann. Oper. Res.*, vol. 244, no. 1, pp. 177–208, Apr. 2015.

[35] V. Nagaraju, V. Shekar, J. Steakelum, M. Luperon, Y. Shi, and L. Fiondella, "Practical software reliability engineering with the software failure and reliability assessment tool (SFRAT)," *SoftwareX*, vol. 10, Jul. 2019, Art. no. 100357.

[36] J. Aubertine, K. Chen, V. Nagaraju, and L. Fiondella, "A covariate software tool to guide test activity allocation," *SoftwareX*, vol. 17, Jan. 2022, Art. no. 100909.

[37] M. Khari, P. Kumar, D. Burgos, and R. G. Crespo, "Optimized test suites for automated testing using different optimization techniques," *Soft Comput.*, vol. 22, no. 24, pp. 8341–8352, Aug. 2017.

[38] W. Liu, J. Yang, Y. Song, X. Yu, and S. Zhao, "Research on software quality evaluation method based on process evaluation and test results," in *Proc. 6th Int. Conf. Dependable Syst. Their Appl. (DSA)*, Harbin, China, Jan. 2020, pp. 480–483.

[39] *Software and Systems Engineering—Software Testing—Part 2: Test Processes*, Standard ISO/IEC/IEEE 29119-2:2021(E), Oct. 2021, pp. 1–64.

[40] *Systems and Software Engineering—Software Testing—Part 2: Test Processes*, Standard GB/T 38634.2, Apr. 2020, pp. 1–56.

[41] *Systems and Software Engineering—Life Cycle Processes—Requirements Engineering*, ISO/IEC/IEEE 29148:2018(E), Nov. 2018, pp. 1–104.

[42] J. Erasmus, I. Vanderfeesten, K. Traganos, and P. Grefen, "Using business process models for the specification of manufacturing operations," *Comput. Ind.*, vol. 123, Dec. 2020, Art. no. 103297.

[43] H. Wang, M. Yang, L. Jiang, J. Xing, Q. Yang, and F. Yan, "Test case prioritization for service-oriented workflow applications: A perspective of modification impact analysis," *IEEE Access*, vol. 8, pp. 101260–101273, 2020.

[44] M. Uddin, S. Islam, and A. Al-Nemrat, "A dynamic access control model using authorising workflow and task-role-based access control," *IEEE Access*, vol. 7, pp. 166676–166689, 2019.

[45] *Software Engineering—Capabilities of Software Testing Tools*, Standard ISO/IEC/IEEE 30130:2016, Feb. 2016, pp. 1–27.

[46] Q. Han and D. Yang, "Hierarchical information entropy system model for TWfMS," *Entropy*, vol. 20, no. 10, p. 732, Sep. 2018.

[47] *Reliability of System and Software—Part 3: Testing Method*, Standard GB/T 29832.3, Nov. 2013, pp. 1–15.

[48] R. Peng, Y. Li, W. Zhang, and Q. Hu, "Testing effort dependent software reliability model for imperfect debugging process considering both detection and correction," *Rel. Eng. Syst. Saf.*, vol. 126, pp. 37–43, Jun. 2014.

[49] C. Zhang, H. Liu, R. Bai, K. Wang, J. Wang, W. Lv, and F. Meng, "Review on fault detection rate in reliability model," *Ruan Jian Xue Bao/J. Softw.*, vol. 31, no. 9, pp. 2802–2825, Sep. 2020.

[50] X. Xiao and T. Dohi, "Interval estimation for non-parametric NHPP-based software reliability model via simulation-based bootstrap," in *Proc. 8th Int. Conf. Dependable Syst. Their Appl. (DSA)*, Yinchuan, China, Aug. 2021, pp. 265–270.

[51] S. Li, T. Dohi, and H. Okamura, "A comprehensive evaluation for burr-type NHPP-based software reliability models," in *Proc. 8th Int. Conf. Dependable Syst. Their Appl. (DSA)*, Yinchuan, China, Aug. 2021, pp. 1–11.

[52] D. H. Lee, I. H. Chang, and H. Pham, "Software reliability model with dependent failures and SPRT," *Mathematics*, vol. 8, no. 8, p. 1366, Aug. 2020.

[53] I. W. Burr, "Cumulative frequency functions," *Ann. Math. Statist.*, vol. 13, no. 2, pp. 215–232, Jan. 1942.

[54] K. Y. Song, I. H. Chang, and H. Pham, "An NHPP software reliability model with S-shaped growth curve subject to random operating environments and optimal release time," *Appl. Sci.*, vol. 7, no. 12, p. 1304, Dec. 2017.

[55] S. Z. Ke and C. Y. Huang, "Software reliability prediction and management: A multiple change-point model approach," *Qual. Rel. Eng. Int.*, vol. 36, no. 5, pp. 1678–1707, Jul. 2020.

[56] K. Pillai and V. S. S. Nair, "A model for software development effort and cost estimation," *IEEE Trans. Softw. Eng.*, vol. 23, no. 8, pp. 485–497, Aug. 1997.

[57] Q. Li and H. Pham, "NHPP software reliability model considering the uncertainty of operating environments with imperfect debugging and testing coverage," *Appl. Math. Model.*, vol. 51, pp. 68–85, Nov. 2017.

[58] C. Zhang, Y. Yuan, W. Jiang, Z. Sun, Y. Ding, M. Fan, W. Li, Y. Wen, W. Song, and K. Liu, "Software reliability model related to total number of faults under imperfect debugging," in *Proc. Int. Conf. Intell. Automat. Soft Comput.*, Chicago, IL, USA, May 2021, pp. 48–60.

NAN LI received the B.Eng. degree in Internet of Things engineering from Dalian Minzu University, Dalian, China, in 2019. She is currently pursuing the M.Eng. degree with the School of Computer Science and Engineering, North Minzu University, Yinchuan, China. Her research interests include workflow technology and standardization technology in the field of software testing.

QIANG HAN (Member, IEEE) received the B.S. degree in computer science and technology from the Minzu University of China, Beijing, China, in 1995, the M.S. degree in computer application technology from Peking University, Beijing, in 2009, and the Ph.D. degree in computer science and technology from the Beijing University of Posts and Telecommunications, Beijing, in 2013.

He is currently a Professor with the School of Computer Science and Engineering, North Minzu University, Yinchuan, China. His research interests include trustworthy software and workflow technology.

YANGYANG ZHANG received the B.S. degree from Beijing Technology and Business University, Beijing, China, in 1998, and the M.S. degree from the Beijing University of Posts and Telecommunications, Beijing, in 2009.

She is currently a Professorate Senior Engineer, the Director of Software Engineering Research Office of Software Application, and the Service Research Center, China Electronic Standardization Institute. She is also an ISO Registered Technical Expert, the Head of ISO/IECJTC1/SC7 China Delegation, and the Co-Editor of more than ten international standards, such as ISO/IEC 25023. Her research interests include information technology standardization strategy, software engineering standardization, and software quality measurement.

CONG LI received the B.Eng. degree in computer science and technology from the Science and Technology College, North China Electric Power University, Baoding, China, in 2019. She is currently pursuing the M.S. degree with the School of Computer Science and Engineering, North Minzu University, Yinchuan, China. Her research interests include workflow technology, design and analysis of intelligent systems, and cryptology.

YU HE received the B.Eng. degree in computer science and technology from Hebei GEO University, Shijiazhuang, China, in 2016, and the M.Eng. degree in computer technology from North Minzu University, Yinchuan, China, in 2022. His research interests include workflow technology and software testing.

HAIDE LIU received the B.Eng. degree in computer science and technology from Taishan University, Taian, China, in 2018, and the M.Eng. degree in computer technology from North Minzu University, Yinchuan, China, in 2021. His research interests include workflow technology, information system and software engineering, and software architecture.

ZIJIAN MAO received the B.Eng. degree in software engineering from the Nantong Institute of Technology, Nantong, China, in 2019. He is currently pursuing the M.Eng. degree with the School of Computer Science and Engineering, North Minzu University, Yinchuan, China. His research interests include workflow technology and Petri nets theory.