Research on Aerospace Software Quality Improvement Based on Software Test Data Analysis

Yadi Hu[1*], *Wei* Zhao[2], *Jinlong* Tao[1], *Dacheng* Feng[1], *and Yiqian* Zheng[1]

[1]Beijing Institute of Computer Technology and Applications, Beijing, China
[2]The First Military Representative Office of Air Force Equipment Department in Beijing, Beijing, China

Abstract. This paper addresses the issue of underutilized software test data in current aerospace software quality management. Through analyzing the current third-party software evaluation process, the paper proposes to optimize software evaluation process and management to enhance software development quality and software test quality. The paper proposes an aerospace software quality improvement framework based on software test data analysis, which provides a reference scheme for aerospace software quality management. Overall, the paper emphasizes the importance of software test data analysis in improving aerospace software quality and provides a comprehensive solution for addressing problems identified in aerospace software quality management.

1 Introduction

The aerospace industry in China has experienced significant development in recent years, marked by the successful completion of several major aerospace projects. This has resulted in the production of numerous aerospace models and a corresponding acceleration in the development of the aerospace software industry. Software has emerged as a critical component of the core capabilities of aerospace models, supporting essential functions like testing, launch, flight control, detection and identification, communication networking, and driving the normal operation of entire aerospace models. In the future, the "software definition" capability is set to become a vital driver of the development of intelligent aerospace [1].

Notably, recent years have seen a rapid expansion in the software scale of critical aerospace software, accompanied by a sharp increase in complexity and task volume. This has led to significant pressure on software development and quality management, which have struggled to keep pace with the growth in software size [2]. In this context, this paper aims to analyze the main problems facing aerospace software quality management, to take stock of the third-party evaluation management process of aerospace software, and to propose a framework for improving the quality of aerospace software based on software test data analysis. Given the rise of AI and big data, test data can be integrated into and effectively support the software development process through mapping of knowledge, machine learning, deep learning and other approaches [3]. This paper focuses on the improvement of aerospace software quality management through software test data analysis, analyzed from a quality management perspective, without analyzing specific software test data application forms.

The paper comprises four main parts. Chapter 2 explores the main challenges associated with aerospace software quality management. Chapter 3 outlines optimization ideas for evaluation process management. Chapter 4 delves deeper into the role that test data plays in software quality improvement based on insights from the third chapter. Lastly, Chapter 5 concludes the paper by summarizing the key findings and offering insights for future research in this area.

2 Current status of aerospace software quality management

2.1 Main Problems

The number and scale of software in aerospace models have grown significantly in recent years, with a corresponding improvement in the software engineering capabilities of various model developers. As a result, the quality of aerospace software has gradually become more standardized and characterized by engineering practices. However, it is crucial to note that aerospace software quality management remains an area that requires significant improvement [4].

Software developers generally lack professional software evaluation and analysis personnel, which limits their knowledge of the software quality of their own unit and the industry as a whole. This lack of insight is primarily caused by shortcomings in understanding the underlying causes of software design defects, resulting in an inability to improve software quality management at the management level. This issue leads to frequent

* Corresponding author: {yadipaper}@163.com

occurrences of the same type of problems, which are difficult to rectify efficiently [5].

Software testers, on the other hand, commonly focus on the quality of the software under test, often at the cost of neglecting the analysis and improvement of quality management levels based on test data. Due to this lack of attention, testers often display inadequate knowledge of their own unit and the industry's current status of software testing quality. They lack a comprehensive understanding of the common defect types and fail to recognize the vulnerable and high-risk defects of specific software, leading to deficiencies in test item granularity, test case granularity, and software test type distribution.

In conclusion, even with the improvement of aerospace software development practices, the aerospace software industry still faces significant challenges in quality management. The issues identified above must be addressed to improve the quality of aerospace software effectively.

2.2 Third-party Software Evaluation Process

China's aerospace software industry has seen significant development over the last two decades, with the successful establishment of a third-party evaluation system under the guidance of the "focusing on evaluation and promoting engineering" policy that was introduced twenty years ago. The end-stage quality control procedures of aerospace software development have helped to improve the overall quality of software engineering [2]. The third-party evaluation has played a crucial role in supporting the engineering of aerospace software development for over two decades, facilitating the rapid development of the aerospace industry in China.

As shown in Figure 1, the current third-party software evaluation process includes several stages: project establishment, test requirement analysis, test planning, test design and implementation, test execution, test summary, and delivery. In the test requirement analysis stage, the tester gathers user test requirement information by communicating with clients and reviewing relevant software design documents. The test requirement analysis includes defining the test requirements and test type, determining the test items and priorities, defining the adequacy of each test item, and finally, forming the test design document, Test Requirement Specification.

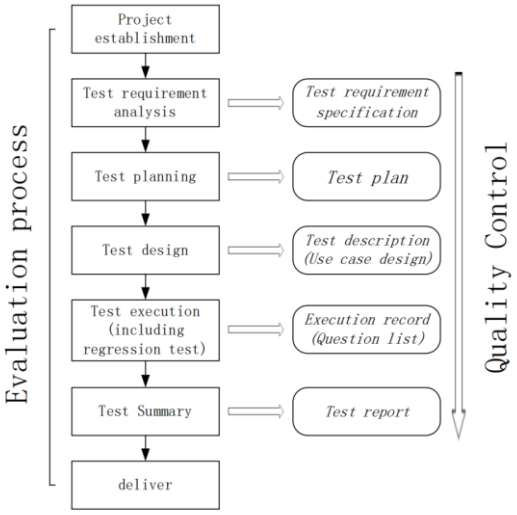

Fig. 1. Third-party software evaluation process.

In the test planning stage, the tester prepares the Test Plan according to the tested parts and design documents provided by the entrusting party and Test Requirements Specification. The Test Plan includes test strategies, technologies, and methods required for testing, estimation of test resources, and risk analysis, among other things.

At the test design and implementation stage, the tester follows the Test Requirements Specification and Test Plan to design the test. This includes designing test cases, preparing verification data, determining the execution sequence of use cases, confirming that the test environment is ready, and creating the Test Description document.

The test execution stage involves executing tests according to the test plan and test description and recording test execution results. The tester then determines whether the test case passes according to the actual test results, expected test results, and evaluation criteria for each test case. If a defect is identified, it is recorded in the software problem report, and the problem is communicated and confirmed with the developer's designer. After correction, the tester carries out regression testing.

In the test summary stage, the tester analyses and evaluates the test work and the tested parts based on the test documents, records, problem reports, and other factors. They prepare the Test Report according to the requirements of the test assignment and deliver it to the entrusting party. Additionally, they analyse and sort experiences and lessons learnt during testing.

It is evident that the current third-party software evaluation process primarily focuses on testing a single software product. This includes software test item division, test case design, test adequacy, and feedback processes for each software product tested, such as questionnaire feedback, confirmation, modification and regression. However, there is a lack of management mechanisms for communication and feedback of a larger range of test data.

3 Software Evaluation Process and Management Optimization

The current software evaluation process controls software and evaluation quality using various process documents. However, this control mechanism presents certain shortcomings, primarily due to the limited communication and feedback mechanism available. The data analysis generated during the software evaluation stage is not timely or effectively fed back to software developers, reducing their ability to improve.

To address the shortcomings of the communication and feedback mechanism within the current third-party software evaluation process, it is crucial to establish effective information communication and feedback channels that match the information acquisition capabilities of the software developer and the software evaluation party. This would enable the creation of a sound, collaborative, data-sharing model that promotes interoperability, data sharing and collaborative progress. As shown in Figure 2, software evaluators, after completing the software evaluation, should periodically summarize, analyze, and predict the data. Once communication mechanisms are established, evaluators periodically feed the analysis results to the software developer, allowing for their enhancement.

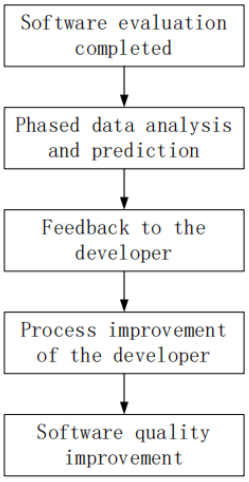

Fig. 2. Communication and feedback mechanism.

Currently, there are several mechanisms that may promote communications between software developers and software evaluators. For example, higher-level institutions lead case study summaries, or evaluators conduct analysis and research, resulting in paper publications [6]. However, such mechanisms can only play a limited role in promoting improvement. To optimize management, it is essential to establish a phased and normalized interconnection mechanism. It should not rely on one specific project, or on several specific projects, but on a more prolonged cooperation and exchange relationship. Consequently, management must not be carried out within a project-specific context. Instead, it is essential to establish a management consultation relationship between software developers and software evaluators regarding the communication and feedback mechanisms.

Simultaneously, it is necessary to create policies and regulations to ensure that the development of AI is ethical and complies with human rights. This is especially important in areas such as facial recognition and surveillance, where there is much debate around privacy concerns. It is also important to consider the potential impact of AI on employment, as it has the potential to replace certain types of jobs. Overall, the development and regulation of AI must be approached with caution and consideration for its potential impact on society.

4 Test Data Analysis and Software Quality Improvement

A crucial step towards software quality improvement is the establishment of an information feedback mechanism and the specification of test data that can promote software quality improvement. This chapter outlines how test data analysis can enhance software quality through the improvement of software development quality and software test quality.

4.1 Software Development Quality Improvement

Higher-level information feedback mechanisms enable the transmission of more macroscopic data than traditional communication feedback mechanisms from previous testing phases. This data can be analyzed from a certain amount of test data. In turn, evaluators can analyze and exchange software product quality evaluation data regularly to help developers comprehend the quality level and quality distribution of their software products, as shown in Figure 3.

For instance, the software defect rate per thousand lines of code, software average cycle complexity, and software requirement function point description granularity are metrics that can be summarized from software quality data derived from software testing. Macroscopic understanding of software development can help developers improve software development quality.

Moreover, evaluators can periodically and regularly summarize and exchange typical cases found during software testing to enhance developers' knowledge, understand the distribution of problems, and guide software problem-solving priorities. Examples of software problems include initialization problems, calculation problems, interruption problems, interface problems, logic processing problems, programming specification problems, etc. Such cases would foster a better understanding of software defects by highlighting the specific issues that should be addressed.

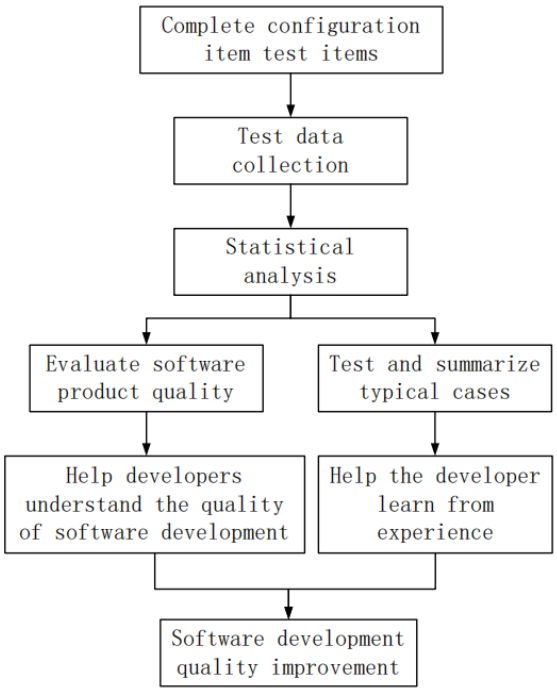

Fig. 3. Evaluation data analysis and software development quality improvement.

For example, the issue of evaluators identifying defects, while developers deem them unimportant, is a recurring problem. Analysis of the data over a three-year period revealed that serious issues go unaddressed, with the frequency of non-modification for important problems at 3% and 30% for general and recommended issues, respectively [7]. This phenomenon, also known as the "evaluator thinks it's a problem, and the developer thinks it can't be modified" situation, stems from poor communication, a lack of management systems, and unclear implementation of responsibilities.

Evaluators may not have conveyed the serious consequences of the software defect to the developer, while developers, including designers, may be overconfident in their software capabilities, and the actual approval process for non-modification issues may be too lenient. Upon receiving the statistical data and relevant analysis, developers can investigate and rectify the root cause of this phenomenon, thereby improving software development quality at the management level.

In summary, establishing an information feedback mechanism and analyzing test data can significantly improve software quality, thus, enhancing the trust and confidence of end-users. It is essential to prioritize software development quality, and rigorous evaluation mechanisms can help enforce the adherence to software quality protocols.

4.2 Software Test Quality Improvement

Software testers play a significant role in software testing as their ability, quality, and testing tools impact the quality of software testing. Test data is also applicable to improving software test quality. Software testers can analyze and communicate software test data to improve their test abilities, become more proficient in test tools, and enhance the consistency of the test scale and test quality.

Similarly to Section 4.1, test data comprises two parts. Besides typical cases, evaluators mainly communicate and monitor software test quality using data. As shown in Figure 4, these data help evaluators understand the quality level and quality distribution of the test process, such as software test item granularity, software test case granularity, software test type distribution, etc.

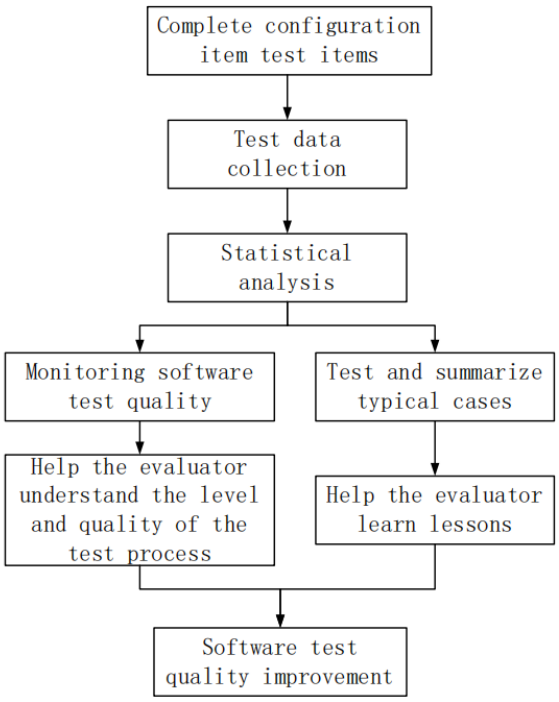

Fig. 4. Evaluation data analysis and software evaluation quality improvement.

5 Conclusion

The quality of aerospace software has always been a significant concern, and this paper analyzed the main problems facing aerospace software quality management. The paper also proposed a framework for improving aerospace software quality based on software test data analysis, which provides a reference scheme for aerospace software quality management.

The paper's framework emphasizes the importance of an information feedback mechanism to enable communication of software quality evaluation data between evaluators and developers. This communication enhances macroscopic understanding, facilitates problem-solving priorities, and improves software development quality. The proposed framework also highlights the importance of software test quality and the use of test data to improve test abilities and enhance the consistency of test scale and quality.

In conclusion, the proposed framework provides a comprehensive solution to improve aerospace software quality management by addressing identified problems and emphasizing the importance of using software test data analysis in achieving better software quality. This framework will significantly contribute to improving software quality, enhancing the trust and confidence of

end-users, and facilitating the growth and success of the aerospace industry.

The author thanks anonymous reviewers and editors for their valuable suggestions on revising and improving the work.

References

1. Xu F, Zhou X, Zhao J, Wu F, Lin Y, Xia Y. *Software-defined satellite technology concept and development*, in Journal of Beijing University of Aeronautics and Astronautics:1-12 (2022)
2. Cheng S. *Building a new system for aerospace software development to support the new leap in aerospace development in the 14th Five-Year Plan*, in National Defense Science and Technology Industry, 2020,(09):34-39 (2020)
3. Chen J, Zhao G, Sun J, Jing M. *Research on the construction of knowledge map for software testing of early warning detection equipment*, in Information Technology and Informatization,2022(03):66-69 (2022)
4. Sagheer M, Zafar T, Sirshar M. *A framework for software quality assurance using agile methodology*, in International Journal of Scientific & Technology Research, 2015, 4(2): 44-50 (2015)
5. Hooda I, Chhillar R S. *Software test process, testing types and techniques*, in International Journal of Computer Applications, 2015, 111(13) (2015)
6. Liu J. *Research on the implementation of improvement Electronic technology and software engineering*,2022(05):84-88 (2022)
7. Luan R, Yann Q, Li M, Xu Y, Chen L. *Some thoughts on improving the quality control of aerospace software evaluation process*, in Space industry management,2021,(08):28-30 (2021)