Explore the Maintenance Measures and Methods of Computer Software Engineering

Can Yan

Xihua University, Chengdu 610039, China.

Abstract: With the continuous development of computer software engineering, maintenance has become a key challenge. This paper takes the software maintainability as the background, takes the actual case as an example, analyzes the maintenance measures and methods, and puts forward the targeted solutions. The findings aim to provide software engineers with practical guidelines to optimize maintenance processes and improve software quality and long-term usability.

Keywords: Computer Software Engineering; Maintenance Measures; Maintenance Methods; Software Maintainability

Introduction

In the field of computer software engineering, as software systems continue to evolve, maintenance becomes a key challenge in ensuring system reliability and continuous innovation. In order to solve the increasingly complex maintenance needs, this paper deeply explores the maintenance measures and methods of computer software engineering, aiming to deal with the changing application background and improve the maintainability and long-term performance of the software system.

1. Significance of computer software engineering maintenance

1.1 Continue the software life cycle

Upgrades and changes in hardware platforms are inevitable, and such changes can have an impact on the performance and compatibility of software systems. Therefore, software maintenance needs to constantly pay attention to the development trend of hardware technology, the necessary adjustment and optimization of the software, to ensure that it can maintain efficient and stable operation on the new hardware platform. This also includes the elimination of old technologies and the introduction of new technologies to keep the software systems advanced and available. The update and evolution of operating systems are also considerations. As the OS versions are updated, the software systems may face compatibility challenges. Software maintenance requires timely applies to the new operating system environment and repairs problems that may be caused by operating system changes to ensure the normal operation and safety of the software. Changes in business requirements are one of the most common challenges in software maintenance. As the business environment changes, the user needs may change, and new functional requirements may emerge. One of the goals of software maintenance is to enable the software system to adapt to these changes through flexible design and modifications. This may include optimization of existing features, development of new features, and adjustments to the entire system architecture. Maintenance also includes updates and additions to existing documents to maintain consistency with the actual code. This is essential for the understanding and work efficiency of the subsequent maintenance personnel.

1.2 Improve and optimize the software quality

Error repair includes not only solving known problems, but also actively responding to user feedback and real-time monitoring of system performance, as well as quickly locating and fixing potential problems by means such as logging. Performance optimization is another key aspect. Regular performance testing and analysis can identify performance bottlenecks and bottleneck causes in the software system. Optimizing algorithms, improving database queries, and reducing resource consumption can all be used to improve the response speed and overall performance of the software. Performance optimization not only improves the user experience, but also helps to improve the scalability of software systems and adapt to the growth of user scale. Introducing new technologies and functions is also an important task of software maintenance. Continuous advances in technology and changes in business requirements may require software systems to adopt new technology stacks or add new functional modules. By introducing advanced technology, the software can better adapt to the new environment and needs and remain competitive. At the same time, adding new features can also improve the market value of the software to meet the evolving needs of users. Software maintenance should also focus on code maintainability. By reconstructing and optimizing the code, we can eliminate the redundancy and complexity, and make the code more clear and concise. This not only helps to reduce the difficulty of future maintenance, but also improves the efficiency of team members to understand and modify the code.

2. Computer software engineering maintenance measures

2.1 Establish a sound document system

Requirements documentation is the cornerstone of software development. Detailed and accurate requirements documentation ensures a consistent understanding of the business requirements by the team, providing a clear direction for subsequent design and development work. The architecture design document focuses on the overall system structure, including the system module division, the interaction between modules, etc., which helps the team to understand the overall blueprint of the system. The detailed design document is more specific, covering the detailed design of each module, including data structure, algorithm selection, etc. Database design documents focus on data storage and management to ensure that the structure of the database meets the system requirements and ensure the integrity and consistency of data. The software installation and deployment document is a key link to ensure the successful operation of the software. It describes the installation steps and configuration requirements of the software in detail, and provides convenience for the deployment personnel. The user manual is a guide to end-users, detailing the software's functions, operating methods, and solutions to common problems, enabling the user to quickly learn and take full advantage of the software's capabilities. To ensure the effectiveness of these documents, regular updates and maintenance are essential[1]. As the project progresses, requirements may change, and system design may need to be adjusted, and timely updated documentation enables consistent understanding across the entire team and avoid lag and inconsistency of information.

2.2 Modular design to reduce the coupling

Modular design is a key software design concept, designed to achieve the design goal of high cohesion and low coupling by dividing the software system into logically independent and functional self-contained modules. The high cohesin modules are closely related and perform a single responsibility, while the low coupling emphasizes the independence between modules and interact through simple and clear interfaces to reduce the dependence between each other. Within the module, the strong correlation and tight combination of functions enable each module to perform specific tasks independently, which helps to improve the maintainability and reusability of the module. With loose functions between modules and simple data exchange, this design makes the system more flexible and easy to adapt to changes. This also means that when new functions need to be added, new modules can be added without affecting the rest of the system, limiting the diffusion range of changes and improving the maintainability of the system. However, caution is needed in the process of module separation, and too meticulous separation may lead to increased complexity of the system, making collaboration and communication between modules difficult[2]. Therefore, selecting the appropriate module granularity is crucial. Reasonable module granularity should not only ensure that a complete business model can be established within the module, but also ensure that the coupling degree between the modules is minimized to form the best mode of cohesion and coupling.

2.3 Adopt the code specification to improve the readability

Naming specification is one of the cornerstones of a programming specification. The naming rules of class names, method names and variable names should be concise and descriptive, avoiding using abbreviation and single character naming to improve the self-interpretability of the code. A clear and consistent naming style helps reduce the code to read, making it easier for developers to understand the code. Format specification is another important aspect. When formulating specifications, we need to pay attention to code indentation, function physique, annotation specification, etc. Consistent indentation and formats can clarify the code structure, reduce ambiguity, and improve readability. The annotation specification should include interpretation of critical business logic, algorithms, and recording of code changes to make it easier for team members to understand the design and use of the code. The implementation of the code specification is not only the responsibility of an individual developer, but also the responsibility of the whole team. By promoting and training the code specifications on the team, a team culture is formed by ensuring that developers jointly follow the specifications when writing the code. Code review can also be more convenient and quick because the entire team follows the same specifications, making it easier for team members to understand and evaluate each other's code. During the software maintenance phase, using consistent code specifications can significantly improve efficiency.

2.4 Build an automatic test system

The foundation of the automated test system is the perfect test cases. These test cases should cover all aspects of the software, including functionality, performance, security, etc. Test cases are designed to fully consider the integrity of the code coverage to ensure that as many code paths are covered as possible, thus improving the comprehensiveness and depth of the test. The choice of an automated framework is crucial. A powerful automation framework can improve the maintainability and scalability of the test code. With an automated framework, the test cases can be run continuously, and the results can be checked in a timely manner. This not only provides timely feedback for every modification of the software, but also provides a stable testing base for the development team. The automated testing system should also support continuous integration, meaning that testing can be seamlessly integrated with other stages of the development process[3]. By automatically triggering the test process, the regression test of the software can be made more easily and more efficient. This automated integration approach helps to identify potential problems early, thus reducing the cost of fixing problems later in the software development cycle. The operation of large-scale automated test cases not only helps to verify the correctness of the software modifications, but also releases the energy of the testing team. By automatically running routine tests, test teams can focus on developing more complex and innovative new test cases.

Conclusion

The in-depth study in the computer software engineering maintenance has laid a foundation for improving the reliability of the software system. In the future, with the continuous evolution of technology, we will continue to explore more advanced maintenance measures and methods to adapt to the increasingly complex software environment. It is expected that these efforts will provide more innovative and efficient solutions in the field of software engineering and promote the sustainable development of software systems.

References

[1] Zhang Jie, Wang Yanmei, Han Qiang. Explore the maintenance measures and methods of computer software engineering [J]. Computer Knowledge and Technology, 2022 (8): 62-64.

[2] Wang Zhe. Research on the maintenance measures and methods of computer software engineering [J]. Information and Computer (theoretical edition), 2020 (18): 90-92.

[3] Sun Zhixuan, Wang Xueying. Analyzing the maintenance measures and methods of computer software engineering [J]. A Guide to Family Life, 2018 (12): 87.