Proceedings of the 2nd International Conference on Software Engineering and Machine Learning
DOI: 10.54254/2755-2721/76/20240558

Research on software reliability enhancement methods based on iterative development

Shuang Huang

Hubei University of Economics, Wuhan, 430200, China

hshuang900@gmail.com

Abstract. Iterative development is an agile software development methodology that speeds up the development process and increases the flexibility of software delivery. However, the current software requirements change extremely fast, which leads to its increasing uncertainty, and iterative development also faces the challenge of software reliability. The software reliability enhancement method based on iterative development is researched and discussed for the current situation of insufficient reliability in the traditional software development process. In traditional software development, one-time delivery pressure and overall design complexity often lead to software functional defects and instability, affecting software reliability. The iterative development method, on the other hand, improves software reliability by decomposing the software development process into multiple iterative cycles, each of which contains requirements analysis, design, coding, and testing, updating the software functionality in smaller increments, and gradually improving the software product. This study proposes a software reliability improvement methodology by exploring and analyzing the key factors related to reliability in the iterative development process. This dissertation can establish a reliable and high-quality software development process through the steps of meticulous requirements planning and management, iterative development and testing, automated testing and continuous integration, code quality control, defect management and continuous improvement, user feedback and user engagement, and continuous learning and technology enhancement.

Keywords: iterative development, reliability, software testing.

1. Introduction

Iterative development is a software development methodology that improves the efficiency and quality of software development by decomposing the software development process into multiple smaller iteration cycles. Background information on iterative development dates back to the 1990s, when the traditional waterfall model development methodology presented some shortcomings in the face of complex and changing software requirements. The traditional waterfall model requires requirements analysis, design, coding, testing and maintenance to be completed sequentially in different phases of the software development lifecycle [1], but it has a drawback once the previous phase is completed, it is difficult to make changes to the subsequent phases if there are any problems. However, in real projects, software requirements often change over time, so the waterfall model often fails to adapt well to such changes, resulting in project delays, cost increases, or failure to meet end-user requirements. In order to solve these problems, iterative development methodology was introduced and is gradually becoming

DOI: 10.54254/2755-2721/76/20240558

widely used. Iterative development works by breaking down the entire software development process into several small cycles, each containing activities such as requirements analysis, design, implementation, testing, etc., with each cycle ending with a version of the software that can be deployed. This step-by-step iterative approach makes the software development process more flexible and controllable, which is very helpful in finding and fixing problems in a timely manner, better adapting to changes in requirements, improving the quality of the software, and enhancing user satisfaction. Iterative development has gained wide application in the computer field. However, the iterative development process lacks an independent testing phase, which makes the traditional reliability model of component software systems, also not fully applicable to componentized soft systems based on agile development [2], and the software reliability problem is often neglected. The aim of this study is to explore the main challenges of reliability during iterative development and to propose a comprehensive approach to improve software reliability.

2. Analysis of Iterative Development Methods in Development Environments

Many methods and techniques that have been developed in the field of iterative development over the past period of time. Scrum is one of them.

Scrum uses empirical process control methodology steps such as controlling, adapting and providing visibility to the complexity of a software development project based on a simple set of practices and rules. It provides an empirical approach to deal with the complexity of software development projects. Providing an empirical approach enables team members to work independently and centrally in a creative environment [3]. But scrum also has significant shortcomings in terms of software reliability. There is a possibility of inadequate or inaccurate requirements analysis on the product side. Since the requirements are formed based on ideas and feedback from stakeholders, the understanding and specification of the requirements may be at risk, leading to software reliability issues in subsequent iterations; on the planning meeting side, tasks in terms of quality requirements and stability may not be adequately taken into account in the iteration planning meetings. The focus may be more toward feature development and enhancements, ignoring the need for reliability, security, and other aspects of the software system. In terms of iteration cycles, time pressures within the iteration cycle may lead to inadequate testing and quality control. Teams may favor rapid delivery for user feedback, but shorter cycles and tight deadlines may limit adequate testing and verification of software reliability.

Kanban is a process management methodology rather than a specific software development methodology, but there may be some shortcomings in Kanban when it comes to software reliability. Kanban focuses on adaptability and flexibility to respond quickly to changes in requirements. However, frequent requirement changes can lead to software reliability challenges. If requirement changes occur unchecked, they can lead to unstable working conditions for the development team, affecting software quality and reliability.

Since these iterative development approaches can have some shortcomings and defects in software reliability, in order to succeed in a changing and competitive market, software development teams need to take steps to reduce the generation of defects, improve the reliability of the software and respond to user requirements in a timely manner For this purpose, an integrated iterative development workflow is proposed here.

3. Integrated Iterative Development Workflow

3.1. Requirements Planning and Management

The requirements analysis and planning phase is a crucial phase in the software development process, which provides the foundation and direction for the subsequent development and implementation of the project. In this phase, requirements are collected by communicating with customers, end-users, and business owners to understand needs and expectations and gather relevant information. Then the collected requirements should be analyzed and organized to ensure the completeness, consistency and traceability of the requirements, and to understand the business objectives and values behind the

DOI: 10.54254/2755-2721/76/20240558

requirements, to clarify the priority and importance of the requirements, and to guide the subsequent processing and implementation of the requirements according to the business values. People need to develop project milestones and plans, define project phases and cycles, and organize tasks and resources to determine the scope and boundaries of the project, and clarify which requirements will be included in the current iteration and which will be deferred to subsequent iterations. Finally, the requirements are validated to ensure that they are accurate.

3.2. Iterative Development

The first iteration gauge is conducted to determine the goal and scope of this iteration based on the results of the requirements analysis and planning phases. The iteration plan is developed to clarify the development time, how the staff is allocated, etc. The development team starts coding based on the requirement specifications or user cases so that the software functionality can be implemented. The coding is practiced according to the previously selected methodology, which ensures that it is clear code. After completion regular code reviews are conducted to ensure that the code is consistent.

3.3. Automated Testing and Continuous Integration

Automated testing and continuous integration are key practices in modern software development that increase development efficiency, reduce human error, speed up delivery cycles and ensure software quality [4]. The first step is to configure the automated test environment, which includes selecting the appropriate automated test tools and frameworks. Test scripts and test cases need to be written in order to ensure that test cases cover key features and scenarios.

The automation test scripts are then integrated using tools such as Jenkins and Travis CI. When encountering situations such as code commits, timed tasks or monitored code changes, set up triggers that trigger automated tests in response. Finally automate the execution of test scripts and generate test result reports. Validate the test result report and notify the development team if there are failed test cases to fix them. In this process, automated test scripts should be continuously monitored and maintained to prevent encountering some conditions to ensure its reliability.

3.4. Code Quality Control

During the software development process, software engineering employs strict quality control methods to ensure that the software meets predetermined quality requirements at all stages. These methods help to improve the reliability, stability and maintainability of the software, thereby reducing maintenance costs and software risks [5]. Among them, code quality control is an important part of software engineering.

Code quality control is the process of ensuring that the software code is maintainable, readable and extensible. Code quality control can be practiced through the following methods: code specification, comments and documentation, design patterns and architecture. Through these code quality control measures, code readability, maintainability and scalability can be improved, defects and technical liabilities can be reduced, and software quality and team development efficiency can be improved.

3.5. Defect management and continuous improvement

Software has become an important factor affecting the national economy, military, politics and even social life. Highly reliable and complex software systems rely heavily on the reliability of the software they employ. Software defects are the potential root causes of errors, failures, crashes, and even fatalities in related systems [6]. Defect management and continuous improvement are key and integral parts of the software development and testing process. They help identify, track and resolve problems in software and drive teams to continuously improve and enhance workflow and product quality. Defect management is the process of ensuring that software meets expected performance and quality by tracking, documenting, resolving, and verifying defects. Defects in the software are found through various tests, user feedback and code reviews, and then the defect details are recorded in the defect management tool, and then the defects are categorized according to the severity level, to determine

DOI: 10.54254/2755-2721/76/20240558

which one is the priority category to be resolved, and then the development team will resolve the defects, and the way of resolving the defects is also different for different defects. After resolving the defects, you need to verify whether the defects have been fixed completely, and finally close the defect record.

Continuous Improvement is a practice of continually pursuing optimization and enhancement, which aims to continuously improve work processes, methods, and product quality [7]. Through continuous analysis and improvement, teams are able to continually improve productivity and product quality, ultimately achieving continuous delivery, customer satisfaction and business success. Defect management, combined with continuous improvement, helps teams identify and resolve problems in a timely manner and continually improve software development and testing processes with an attitude of continuous improvement. Therefore continuous improvement is also an essential process.

3.6. User Feedback and Participation

Scrum and Kanban's biggest flaw is that the user's participation is not high [8], only in the early stage of understanding the user's needs, in the development process can not accept the user's suggestions and improvements, which makes the process is not very perfect in line with customer expectations, in order to solve this problem, to allow customers to actively participate in it.

Users can be invited to participate in product testing and trial, collect feedback and comments, carry out user research, understand user needs, habits and expectations, and guide product improvement and functional development. You can also make use of the user experience design methodology to put users at the center of the design process and design products that better meet users' needs and expectations. In dealing with user feedback, it is more important to respond to user feedback in a timely manner, expressing concern and willingness to solve problems. Analyze and classify the user feedback to find out the common problems and priority problems. Feedback tracking system can also be set up to track and record the progress of problem solving, and provide timely feedback to the user to solve the problem. Finally, user feedback is used as an important basis for product improvement, and corresponding improvement plans are formulated and implemented. This process through active interaction and participation with users, the team can better understand user needs, improve the product, improve user satisfaction, and lay a solid foundation for the long-term development of the product.

4. Implementation Case Study

This is a case study where an e-commerce team needed to develop an e-commerce platform and decided to manage the project using an integrated iterative development workflow. Through the integrated iterative development workflow, the team worked closely with stakeholders in each iteration to continuously develop and deliver new functionality, and from the feedback and retrospectives of each iteration to continuously improve the workflow and product quality. This agile development approach resulted in significant improvements in software reliability, a marked reduction in the number and severity of defects, and a team that was better able to respond to change and risk and deliver a high-quality e-commerce platform. The success of the real-world project examples demonstrated the effectiveness of the integrated iterative development workflow [9]. The results of the team's practice in finding significant improvements in software reliability and significant reductions in the number and severity of defects show that choosing and adopting an integrated iterative development workflow was the right decision and provides valuable experience and insights for the team's future work and project management. Continuing to maintain an attitude of teamwork, continuous improvement and learning will help to further enhance the efficiency and quality of the software development process.

5. Advantages and Disadvantages Analysis

Compared with the traditional waterfall development methodology, the iterative development approach shows significant advantages in reducing the defect generation rate and improving software stability. It has the ability to quickly adapt to changes in requirements, allowing the team to adjust development direction and priorities in a timely manner based on feedback. Having a usable version of the product for each iteration allows for quick demonstration of features and feedback to stakeholders, increasing

DOI: 10.54254/2755-2721/76/20240558

customer satisfaction. During the iteration process, the team continuously conducts code reviews, unit tests, and integration tests to ensure software quality and reliability. However, there are also some high demands on the team, requiring frequent communication and collaboration, and the time requirements of short-term iterations and the pressure of frequent deliveries are also challenging for the team. As shown in Figure 1 and Figure 2, the iterative development model is compared with the traditional waterfall model.

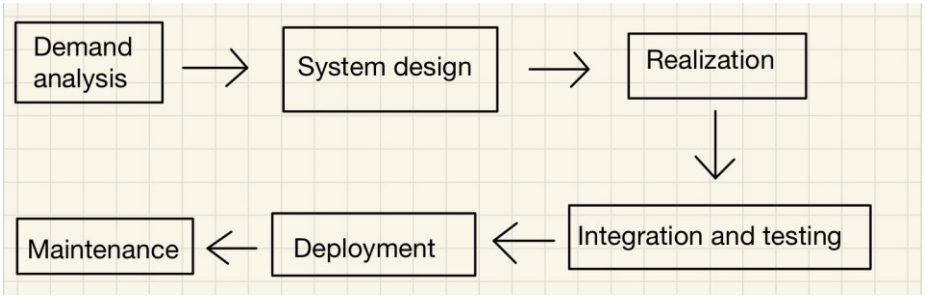

Figure 1. Waterfall model

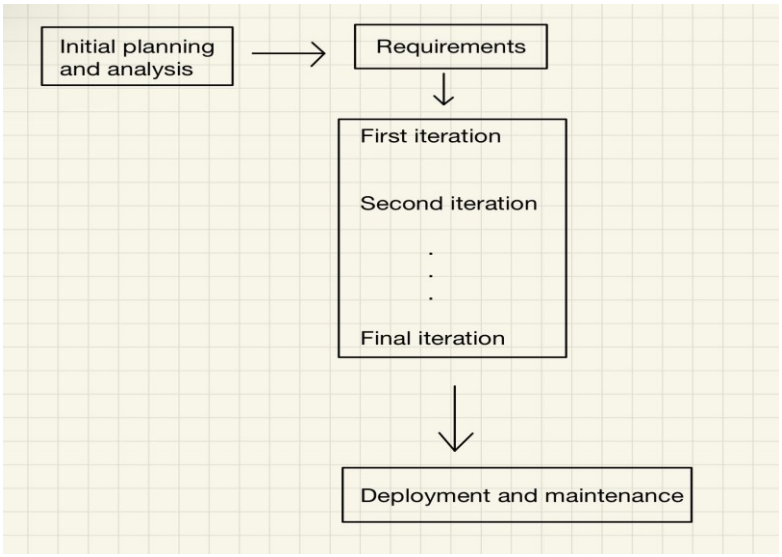

Figure 2. Iterative development model

Overall, an integrated iterative development workflow has significant advantages in terms of agility, delivery value, and quality assurance, but also needs to overcome communication challenges and time and resource pressures. The advantages and disadvantages should be balanced in practical application, and the team's strengths in the iterative development process should be better utilized according to the team's characteristics to maximize the efficiency and quality of software development.

6. Conclusion

Overall the software reliability improvement method based on iterative development is an important trend and direction in the current software development field. Iterative development not only emphasizes teamwork, code quality, risk assessment, defect management and continuous improvement in the software development process, but also improves the agility and flexibility of the software development team. At the same time, the flexibility and agile delivery characteristics of iterative development also provide more opportunities and space for software developers to help better respond to market changes and user needs, and increase the likelihood of project success. With the rapid development of artificial intelligence and machine learning technologies [10], future research can

DOI: 10.54254/2755-2721/76/20240558

explore how intelligent technologies can be applied to the iterative development process to automate tasks, intelligent decision-making and adaptive optimization to improve software development efficiency and quality. The combination of iterative development with agile development methods such as DevOps and continuous integration can also be explored in depth to further enhance the automation, collaboration, and continuous delivery capabilities of the software development process to better meet rapidly changing requirements.

In this paper, although the concepts, background information and importance of software reliability enhancement methods based on iterative development have been covered, there are still some shortcomings. In this paper, when introducing and discussing the software reliability enhancement method based on iterative development, there is insufficient theoretical support for the related theories, and there is a lack of related theoretical models and graphical explanations. There is a lack of authority in the use of cases, and the examples used around the world do not fully support the feasibility of the iterative development process. Although this study has elaborated the comprehensive process of iterative development to a certain extent, there are still some shortcomings that need to be further refined and improved. It is hoped that future research will be able to delve deeper on the basis of this paper, solve the existing problems, and contribute more valuable research results to the development of this field.

Acknowledgements

I would like to thank the Hubei University of Economics for the research project that provided significant support and funding for this study. The support of this project provided me with the necessary resources and platforms that enabled me to conduct in-depth research and complete this paper. I would like to express my heartfelt gratitude to the leadership of the college and related departments for their support and encouragement.

References

[1] Chen Xi, Mingkun Xu. Research and practice of test-driven development in software development [J]. Software, 2012(12):177-181.

[2] Wang S.A. Componentized software reliability research based on agile development [J]. Microelectronics and Computers, 2011(3): 119-122.

[3] Wensheng Hu, Ming Zhao, Jianfeng Yang, et al. Analysis of iterative strategy in agile development process [J]. Microelectronics and Computer, 2012(5):165-169.

[4] Yuanyuan Dang, Xiaolin Fu, Lixin Xu. An applied research on Scrum agile project management [J]. Journal of Intelligence, 2009(3):54-57,61.

[5] Chen Wei. Application of software engineering in enterprise informationization platform construction [J]. Electronic Communication and Computer Science, 2023(9).

[6] Qing Wang, Shujian Wu, Mingshu LI. Software defect prediction techniques [D]. Journal of Software, 2008.

[7] Guilin Zhang, Xiaomei Zhang, Ding Feng. Basic forms and key technologies of software engineering [J]. Computer program, 2013(10):93-94,117.

[8] Guoping Rong, He Zhang, Dong Shao, & Qing Wang. A review of software process and management methods. Journal of Software, 2018(1), 62-79.

[9] Zhang, X. J., Jiang, T. M., & Wang, X. F.. Research on methods to improve the reliability of computer networks [J]. Computer Engineering and Design, 2010(5):990-994.

[10] Lichun Kuang, Liu He, Yili Ren, et al. Application status and development trend of artificial intelligence in petroleum exploration and development [J]. Petroleum Exploration and Development, 2021(1): 1-11.