Region - Educational Research and Reviews, 2019, Volume1, Issue2
http://front-sci.com/journal/rerr
ISSN Online: 2661-4634
ISSN Print: 2661-4626

Method of Domestic Small and Medium-sized Software Projects

Jianfeng Bao

Zibo Vocational Institute, Zibo, 255314, China

Abstract: This paper mainly introduces the problems existing in the development process of small and medium-sized software projects in China, and analyzes the problems solved by the heavy load development method and the light load development method, and finally compares and chooses the two development methods.

Keywords: software development method; heavy load development method; light load development method

Due to the lack of optimized software development methods and scientific management methods, many domestic small and medium-sized software development projects have encountered many problems in the development process, which has resulted in failure to deliver qualified software on time, and project overruns have even failed. Some software companies try to improve this by learning and adopting some "popular" software development methods or quality control methods, but the results are often unsatisfactory. Often, old problems are solved while a series of new problems are generated. Investigating the reasons, we believe that these companies' failure to "appropriate remedies" in choosing software development methods is the key.

1. Problems in the Development of Domestic Small and Medium-sized Software Projects

Compared with developed countries, the level of software development in China is relatively low. Reflected in small and medium-sized software projects. First, there are many small and medium-sized projects developed by software companies in China. Second, many small and medium-sized software projects have more or less unsuccessful development, which often leads to budget overruns and delays. The quality cannot meet the design requirements. These problems can be roughly described in terms of external environment and internal management, and internal management includes methods, specifications, tools and how to ensure the implementation of specifications through scientific management. If these problems are not handled properly, the software will have low quality consequences.

1.1The difficulties brought by the external environment to the development of small and medium-sized domestic software projects are mainly manifested in tight schedules, heavy tasks, frequent changes in demand or environmental impacts on the project. The market outside the enterprise is not yet complete and the resources of the industry chain are unevenly distributed. Most software companies are relatively small in scale and poorly managed to form fierce competition.

The scale of small and medium software development projects determines the characteristics of short development time, fewer developers, and heavier tasks. Improper processing time and workload arrangements will cause software quality to decrease. Tight time will cause many unexpected problems in the development and there is almost no time to deal with. At this time, the patience and level of the management staff will be tested. Requirements are always changing, and frequent requirements changes have a greater impact on small and medium-sized software projects with shorter development times. There are many factors that cause changes in requirements, both from problems discovered during the development process due to accumulation of experience, as well as changes in software requirements and development environments; both for customer reasons and for the development team itself. In the end, changing requirements trigger changes in development and design behavior from outside the development team.

1.2In terms of internal management, small and medium software projects run by small and medium-sized software companies often do not use the most suitable tools and methods, and lack of advanced scientific concepts in management. Small and medium-sized projects due to the above problems make it difficult to capture user needs during development and new requirements are continuously generated, unit tests or integration tests are often skipped or shrunk, software integration is difficult, and there are still a large number of software bugs during trial operation .

2. Analysis of Problem Solving with Overloaded Methods

There are multiple methods for software development, including heavy and light loads. Small and medium-sized software projects, especially small and medium-sized projects developed by small and medium-sized software companies, should learn and learn from these methods during the development process until a new method that suits them best is produced. The reality is that many domestic software companies do not have their own scientific and applicable software development methods.

Starting from the earliest "waterfall" software development method, the overloaded software development method has played a significant role in solving software dangers and improving software quality. Generally speaking, overloaded software development methods can help solve problems in software project development as follows:

2.1Overloaded software development methods are known for their complete documentation, rigorous processes, and controls. By controlling the process under the conditions of time, it is easier to find and correct the work-arounds in the implementation.

2.2The overloaded software development method is suitable for larger development teams. The large development team consists of many people, and the role of the document can be fully reflected at this time.

2.3The overloaded software development method is suitable for large and complex projects. Large-scale projects place high demands on developer collaboration, and generally do not encourage free play. The overloaded method strictly controls the process, and can control the probability of errors in the software as much as possible.

2.4There are many good practice activities in overloaded methods, which are worth learning from various methods.

Overloaded software development methods, while solving some problems encountered in software development, also show some deficiencies in use, as shown in the following:

2.4.1Edge effects of reloading software development method documentation. Traditional methods require anticipating questions that customers may ask in the future and require practical understanding, which requires too much documentation. There are too many documents and it is not easy to keep updated, which will make it necessary to find useful information or find answers to questions through these documents very difficult.

2.4.2The ability of overloaded software development methods to adapt to frequently changing requirements is weak. Especially for small and medium-sized software projects, the development cycle itself is short. If the ability to adapt to changes is weak, the relative impact will be much greater. The overload method pursues the predictability of the project and the visibility of the process status. It does not pay enough attention to improve productivity, and a large amount of documentation places a great burden on developers. At the same time, rapid changes in demand and predictability cannot coexist.

2.4.3For small and medium software companies and software projects, the cost of using overloaded software development methods is relatively high.

3. Analysis of Problem Solving with Light Load Method

The purpose of using light-loaded agile methods to develop software is to introduce some practices, through the required discipline and feedback, to provide some principles to ensure the flexibility and maintainability of the software, let us know some design patterns and some use the balance law to solve special The principle of the problem. There are many advantages to developing software projects using light-load methods. Some of these advantages are common to various agile methods, and some are specific to some agile methods. These advantages are manifested in:

3.1The user satisfaction of agile software development methods is high. Agile software development methods work closely with business people and developers to frequently deliver working software, thereby increasing customer satisfaction.

3.2Agile software development methods quickly adapt to changes in requirements. Software requirements and development environments change as development progresses. Lightweight agile methods embrace change, and can adapt to the needs of changing needs through the adaptability of development organizations.

3.3The development efficiency of agile software development methods is relatively high. Agile methods focus on just enough principles. That is, on the premise of ensuring that software development has a successful output, minimize the activities and artifacts in the development process. And pursue the simplest design and require the simplest functions.

3.4Agile software development methods put less pressure on developers to write heavy documents. The light-load method uses code as its core content, and it seems unrealistic, at least not economical, to establish a complete and continuous requirements document.

3.5Agile methods provide a good foundation for requirements management. All user requirements are written on an index card or a constantly maintained feature list. The various agile methods are similar in this regard, the only difference is the requirement for details.

Although agile industry believes that agile methods can reduce management resources, increase productivity, improve software quality, and shorten release cycles. But in fact, agile methods are not necessarily a guarantee of success. Light-loaded agile methods are weak in the following areas:

3.5.1Agile methods need to be improved in dealing with non-functional related requirements.

3.5.2Light-loaded agile methods place too much emphasis on the importance of code, ignoring the importance of analysis and architectural design, and the lack of complete documentation affects the transfer and inheritance of knowledge within a company or organization.

3.5.3The paper story cards used in Extreme Programming are because they are more convenient to display, transfer, and distribute. But is this a step backwards in an increasingly electronic era?

3.5.4Many agile methods require on-site customers, which is difficult to do in real life.

3.5.5The requirements on developers are too high. Agile methods abandon all unimportant behaviors and allow developers to focus on development. When there are differences in the level of the software development team, too many novices or slow programmers will affect the progress of the entire team.

3.5.6Not suitable for large software team development, not suitable for large and complex software system development. The light load method emphasizes close communication. When the number of teams is large, pair-by-two communication becomes complicated and difficult.

3.5.7The light load method has less consideration in the sustainable development of software companies. The light-loaded software development method does not pay attention to documents, and uses code as the core document, hoping to pass information through close communication between people. Such an approach may make the gains and losses of development experience not recorded due to incomplete documentation, and the flow of personnel may cause the loss of information in the enterprise.

Conflicts of Interest

The author declares no conflicts of interest regarding the publication of this paper.

References

[1] Robert C.M. (2003). Agile Software Development--Principles, Patterns, and Practices, *Beijing: Tsinghua University Press*, 9.

[2] Zhong Q., He J.M. and Zhang W.Q. (2004). Discussion on Unified Process Based on Extreme Programming, Journal of Southwest China Normal University, 2, 209-212.