Study on the Development Process of Several Small Micro-software under the GJB5000A Three-level System

Xiaoyu Yang¹ᵃ*, Shengrui Yuan²ᵇ, Xunyi Dang³ᶜ
¹CHINA NORTH VEHICLE RESEARCH INSTITUTE, BEIJING, CHINA
²CHINA NORTH VEHICLE RESEARCH INSTITUTE, BEIJING, CHINA
³CHINA NORTH VEHICLE RESEARCH INSTITUTE, BEIJING, CHINA

*Corresponding author’s: a-e-mail: yangxiaoyu@xywwgc5.onexmail.com, 15210280054@163.com
ᵇ-e-mail: yuanshengrui@163.com, c-e-mail: 12244270@qq.com

Abstract: In order to standardize and normalize the development process of small micro-software, based on the requirements of GJB5000A three-level system, this study is based on the software engineering management technology of process improvement, so as to adapt to the actual management requirements of software.

1. Introduction
At present, the software mainly developed in this study is embedded ones, and most of them belong to the software of driver or control class. In addition, the software scale is small, the development cycle is short. At the same time of implementing GJB 5000A three-level system, it is very necessary to study a development method that not only conforms to the requirements of software process improvement and equipment software specifications, but also conforms to the technical characteristics of small and micro embedded software.

2. Overview of Software Development Process

2.1 Waterfall Model
The waterfall model is used in most of the projects. Its characteristic is that under this development model, the software development process must be carried out in a linear way. That’s simple intuition. The project is divided into several phases, from top to bottom, with a fixed order of transition between the phases. And the output of the previous stage is taken as the input condition of the latter stage. In the end, each stage has a clear mission and needs to produce a definite result.

Because the development stage of the model is clearly defined, it is conducive to the review, verification, tracking, management and control of the project. Before writing code, you should fully emphasize and pay attention to requirements and design. In this process, the waste of time is avoided and the demands of users are guaranteed. However, this model is only applicable when the software development requirements in the early stages have been fully determined. In addition, the workflow of determining a model can only go down one by one, not back up. Then at the end of each phase, it is reviewed and validated, otherwise it will hide the problem and cause more serious consequences. In addition, during the development process, changes proposed by users are difficult to be responded to, thus increasing the risk of project development[1].
2.2 Incremental Model

The incremental model refers to software development by following an incremental approach. In other words, it combines the sequential features of waterfall model with the iterative features of prototype model to produce an incremental model with software components. In this process, a software product is presented as a set of incremental artifacts or modules that are released each time requirements are analyzed, designed, implemented, tested, and delivered until all artifacts are fully implemented. And in the actual development of software, the software system is decomposed into many incremental components according to functions, and developed and delivered one by one in the unit of components until all incremental components are developed fully. Eventually, these are integrated into the system and delivered to the user.

In incremental model, people can be assigned flexibly, and there is no need to invest a lot of human resources at the beginning. When the core module’s product is recognized, the manpower can be added to implement the next increment. During the development process, when the users’ requirement changes, the corresponding intermediate version can be obtained continuously according to the requirement changes, thus reducing the risk of development. However, since each component is gradually incorporated into the existing software system structure, adding components cannot destroy the system part of the existing components. Therefore, this requires the software to have an open architecture and pay more attention to the integrity of the software process control caused by changes in requirements.

2.3 Agile Development Model

The agile development model refers to a typical lightweight software development approach. It combines the advantages of many lightweight software development methods to emphasize people-oriented principle and highlight the characteristics of “Adaptability”. In addition, it can quickly adjust to various changes in the software development process, and minimize the cost and risk of software development. In fact, the agile development model focuses on the design of concepts and software architecture, thus simplifying the detailed design part of the software and leaving space for later adjustments[2].

In this study, agile development model is a software development method based on iterative thinking and can be delivered quickly, which advocates simple design. When architecting, each of iteration of a software project is designed to be “just right”. Then the design is gradually adjusted with the deep communication of users’ requirements. Finally, you can avoid unnecessary work and respond quickly to changing requirements. Compared with other development methods, agile development model adopts the technology of reconstitution, so as to constantly adjust the structure of the system, and finally make the system have strong ability to adapt to the changing demands.

2.4 Models Applicable to Small Micro-software

Considering the characteristics of small and micro software, like small scale, short development cycle and more reusable modules, adopting agile development model is the best choice. Generally speaking, small and micro software are embedded driver class or simple control class software, having simple logic and strong reusability. Therefore, using the agile development model, you can organize the asset base in GJB5000A by selecting the appropriate reusable modules based on the hardware types. At the same time, the reuse design according to the user needs can shorten the development cycle to the greatest extent, so as to reduce the development cost and risk.

3. Software Architecture Design Based on Agile Development Model

3.1 Principles of Design

Generally speaking, the outstanding advantage of agile development lies in its fast and iterative development approach. In detail, the first time the working software is delivered to customers, and then according to the communication with customers, feedback the use of software, and then
according to customer needs to adjust the structure of the software. In addition, it is a people-oriented, iterative and progressive development method. This though permeates all aspects of agile software development. For the design of software architecture, and also follow this principle, the software architecture design process based on agile development pattern is shown in the figure.

![Figure 1 Agile development iterative architecture design process](image)

More importantly, the development process of agile software is also a process of gradually perfecting software functions and gradually upgrading versions. Throughout the software development cycle, iterative refinements are used to modify, refine, and enrich the design solution so that the architecture is optimized to meet the user’s requirements for the software to the maximum extent. Therefore, gathering requirements and analyzing are performed prior to design, iteratively gathering users’ business requirements. Among them, each of iteration only analyzes the requirements in the current iteration, designing for the requirements of this iteration, and finally verifies the design. In agile software development, developing and testing are interleaved. In each of iteration, the developer completes the corresponding task, submitting it through rigorous testing, and validating the requirements with the customer, ultimately ensuring that the developing function is valuable.

3.2 Design Method

In agile development, software systems can adapt to a rapidly evolving and frequently changing environment through constant reconstitution. During reconstitution, we can improve existing designs by using design patterns.

3.2.1 Reconstitution

Next is reconstitution, an important technique in agile development. That is to say, on the basis of not changing the existing functions of the software, we can improve the quality and performance of the software by adjusting the program code, so as to make the design pattern and architecture of the program more reasonable, and finally improve the expansibility and maintainability of the software. In addition, agile development makes code cleaner and easier to maintain through constant reconstitution. At the same time, it does not require a lot of initial time for detailed planning. As a result of a cycle, customers’ demand changes in the products delivered. Therefore, the agile architecture needs to be reconstituted to meet customer needs[3].

3.2.2 Structure of Design

Design patterns are a set of code design experiences that have been used over and over again and that most people know about. They are also a summary of skills. In fact, reconstitution and design patterns may seem different, but they all have the same idea essentially. Design is important, but the way to achieve good design is different. In agile development, code can be reconstituted with help of using
design patterns. This makes your code more reusable, more understandable, and more reliable.

Figure 2  Agile architecture design patterns

4. Development Method of Small micro-software and its Combination with GJB5000A Three-level System

4.1 Introduction of GJB5000A Three-level System

Capability maturity model (CMM) for military software development adopts the hierarchical representation method, which is mainly divided into five levels: level 1 (or M1) is the initial level, level 2 (or M2) is managed level, level 3 (or M3) is defined level, level 4 (or M4) is quantitatively managed level, and level 5 (or M5) is optimized level. More generally, from a software development perspective, the process of maturity level 1 is usually arbitrary and unstructured; the projects of maturity level 2 maturity have ensured that their processes are planned and executed in accordance with guidelines; at maturity level 3, processes are well defined and understood, and described in terms of standards, procedures, tools, and methods[4].

Generally speaking, the standard contains 22 process domains and is classified by different levels of maturity, while each maturity level contains several process domains. As an organization raises the maturity level process, the process domains in the corresponding maturity level and the entire process domains in the lower maturity level should be used to incrementally achieve the specific and shared goals associated with several process domains. Among them, maturity level 2 includes a total of 7 process domains: configuration management, measurement and analysis, project monitoring, project planning, process and product quality assurance, demand management, and supplier agreement management. On the basis of maturity level 2, maturity level 3 adds 11 process domains, such as decision analysis and decision making, integrated project management, organizational process definition, organizational process focus, organizational training, product integration, demand development, risk management, technical solutions, confirmation and verification[5].

4.2 Development Mode of Integrated Internal Control Software

In fact, in the process of development, in line with the principles of integrating resources, professional management, reducing workload, and avoiding the influence of external factors, Small micro-software will undertake different software projects of the same model (system), the same cycle, the same development team, and the same life cycle, and finally carry out centralized management as only one project. In other words, the process of software engineering is still managed by multiple projects according to the requirements of the assignment book, forming work products of each project and independently implementing technical review of each project. In the process of software management, integrated management is implemented. Multiple software projects are regarded as one integrated large project, and then unified management is implemented to form a set of process records. Moreover, through the integrated management of project can carry out the professional management and reduce ineffective management. In addition, starting from improving efficiency and reducing workload, so as we can alleviate the shortage of human resources.

First, it is up to the project leader to make the software development plan, preparing the project schedule, and identifying the project risks. At the same time, the corresponding risk plan and emergency plan should be made, and the data management plan of the project should be prepared. In addition, development library, controlled library and product library are used for management and
maintenance to prepare the resource plan of project, project plan of stakeholder participation, review plan, plan of measurement and analysis, etc., and finally complete Project Software Development Plan, thus establishing project WBS.

The software development process is shown in the figure below.

Figure 3  Agile software development process

5. Conclusions
In this study, the agile model suitable for small micro-software is introduced. At the same time, the design method of software architecture in agile development mode and its implementation method in GJB5000A system are analyzed. Finally, these for the development process of small micro-software can be referenced.

References
[1] Han, likai, Gao yinsheng, Yang quan, (2013) Software Engineering. Tsinghua University Press, Beijing.
[2] Richard F. Schmidt, (2016) Software Engineering: Architecture-Driven Software Development. China machine press.
[3] Martlin, R.C, (2010) Agile Software Development: Principles, Patterns, and Practices. Posts & Telecom Press, Beijing.
[4] Shi, zhu, (2013) Military software development capability Maturity Model and its application. Standards Press of China, Beijing.
[5] Zhou zhenzhen, Wang haoyu, (2017) Discussion on software development management based on GJB5000A three-level process area. Software Tribune, 16(1): 133-134.