Journal of Information and Computing

Research on Cross-platform Software Development and Design Bbased on C++ Language

Zhou Kexin
Metron HK Limited, Kowloon, Hong Kong

Abstract: With the advent of the mobile Internet era, the demand for cross-platform software among users is increasing. Based on the C++ language, this paper studies the development and design process of cross-platform software. Firstly, the main advantages of cross-platform development are analyzed, and the technical route for cross-platform development based on C++ is expounded. Then, the cross-platform software development process is optimized from the requirements analysis, architecture design, coding implementation, testing and release and other aspects. Finally, the feasibility and effectiveness of the proposed method are verified through real project cases. The research shows that cross-platform software development based on C++ can significantly improve the development efficiency, reduce development cost, and broaden the application scope, warranting its promotion for application in more scenarios.

Keywords: C++; Cross-platform; Software development; Mobile application; Development process

I. Introduction

In recent years, with the rapid popularization of smart phones, tablets and other mobile terminal devices, the mobile application market has shown an explosive growth trend. At the same time, iOS, Android and other mobile operating systems coexist, and software developers are facing the problem of developing multiple versions of applications for different platforms, with heavy workload, low efficiency and high maintenance costs. Therefore, the cross-platform software development technology comes as The Times require. Through the concept of "one coding, multiple operations", the cross-platform software reuse is realized to improve the development efficiency and quality.

As an underlying system-level language, C++ is widely used in cross-platform software development due to its high-performance and portable characteristics. Many mainstream cross-platform development frameworks, such as Qt, Cocos2d-x, etc., are implemented based on C++. This paper will focus on the application of C++ language in cross-platform software development, systematically elaborate the cross-platform software design and development process based on C++, and analyze and demonstrate combined with actual project cases, in order to provide some reference for the industry.

II. Cross-platform software development process optimization based on C++

A. Requirements analysis and prototyping design stage

Any successful software product comes from accurate demand analysis and domain modeling. The same is true for cross-platform software development, which requires in-depth research on user needs at the initial stage of the project, and clarifying the target platform and functional positioning. Demand analysis is a top priority in determining the feasibility of a cross-platform solution. Through questionnaire survey, user interview, competitive product analysis and other methods, we can fully understand the real needs and pain points of users, what are the target user groups of the product, and what the core problems are expected to be solved. On the basis of clear requirements analysis, the software prototype design. A good prototype can intuitively present the interface layout, human-computer interaction, and business process of the software. It is a programmatic document to guide cross-platform development. Using Axure, Sketch and other professional prototype tools, the high-fidelity prototype was designed, and reached agreement through repeated review and modification, laying the foundation of cross-platform development.

B. Cross-platform framework selection and training

The implementation of cross-platform software development is inseparable from an excellent cross-platform development framework. Choosing the appropriate development framework is the key move to improve the engineering efficiency and guarantee the software quality. At present, the mainstream C++ cross-platform development frameworks in the industry include Qt, wxWidgets, Cocos2d-x, etc., which have their own characteristics in the development of language, graphics library, tool chain, performance, ecological support and other aspects. Technology selection should be based on the technical background and project needs of the team, and conduct a comprehensive evaluation according to the learning cost, development efficiency, operation performance, platform coverage, community activity and other dimensions. For example, Qt is widely popular in enterprise application development with its powerful UI capabilities, rich API support, and an active developer community. After selecting the framework, technical training should be organized in time to systematically learn the architecture, core mechanism and development specifications of the framework, so as to ensure that the development team understands the usage of the framework as soon as possible and avoid various common traps and misunderstandings.

C. Modular development and unit testing

Cross-platform projects often have complex functions and a large amount of code, which puts forward higher requirements for development efficiency and quality control. Using the idea of modular development, the complex system is divided into multiple modules with independent functions and low coupling degree, which is not only conducive to the division of labor and cooperation, but also conducive to unit testing. Through further packaging on the basis of the framework, a set of unified external interface is designed for each module to call. Use the design mode to optimize the module structure and improve the readability and maintainability of the code. And through code review and other ways, timely find and solve potential cross-platform compatibility problems. Another big advantage of modularity is testability. Using the unit test framework such as GoogleTest, Boost.Test to conduct comprehensive unit testing of key modules and interfaces. Write complete test cases to verify the logic of the code from the forward, reverse, boundary, abnormal and other perspectives. Unit testing is used as the "insurance link" in the development process to detect and repair defects early, and ensure software quality from the code source.

D. Platform integration and joint modulation test

After modular development and unit testing, each functional module has been basically completed. The next step is to integrate these modules into each target platform for joint debugging and integration testing. Cross-platform software often encounters various environmental dependence and compatibility problems in the integration stage, which requires developers to have rich debugging experience and strain ability. For example, file systems, multithreading, and network communication mechanisms may differ between different platforms, requiring conditional compilation and adding platform-related macro definitions. For another example, the adaptive effect of the UI layout on different sizes of screens requires repeated debugging to achieve the best effect. As a result, contingency testing is often the most challenging part of cross-platform development, which requires a lot of time and effort. When necessary, extreme scenarios such as weak grid and low power should be simulated to test the fault tolerance and stability of the software. Through careful testing, the problems in the integration stage are exposed and corrected in time, laying a solid foundation for the subsequent multi-platform release.

E. Multi-platform packaging and release

When the software function is stable and the joint adjustment test passes, it enters the packaging release stage. Different platforms have different release processes and specifications, such as iOS to be uploaded to App Store and Android to be released to major application markets. Different platforms have different requirements for the signature, reinforcement, and review of installation packages, and developers have to customize the packaging scripts and profiles one by one. For example, an iOS application must be signed with a certificate issued by Apple, and a Android application requires code confusion and reinforcement. At the same time, the application shelf process of different platforms varies greatly, with the review cycle ranging from several days to several weeks. Therefore, it is necessary to make multi-platform packaging planning in advance, and closely cooperate with relevant operation and marketing departments to timely update the software version information, innovative function highlights, and prepare publicity materials. Reasonable arrangement of the release pace, we should not only balance the development of resources, but also take care of the user experience. At the same time, it pays attention to the collection and sorting of user feedback, tracks the data indicators of key platforms, and accumulates first-hand information for the subsequent iterative optimization.

III. Case analysis of the cross-platform software developmentprojects

The authors team recently completed the development of a cross-platform education application. Here, we take this project as an example to analyze the engineering practice of cross-platform software development.

The project is aimed at primary and middle school students, and provides online exercises, exam models and other functions. It needs to be launched on three platforms: PC end, mobile end and web end, and realize account exchange and data synchronization. The project uses C++ as the main development language and conducts cross-platform implementation based on the Qt framework. First of all, in the early demand analysis link, the functional needs and non-functional requirements of each platform are defined through questionnaire survey, user interview, competitive product analysis and other methods. According to the UI interaction characteristics of different platforms, three prototype schemes of Mobile, Pad and Desktop were designed respectively. Then, organize the special training of Qt framework, focusing on learning its building system, signal slot mechanism, Model/View framework, Qt WebEngine module, etc., so as to make technical preparations for the subsequent modular development.

In general, the project is designed according to the architecture mode of "core library + platform plug-in". Among them, the core library includes the business model, data storage, network communication and other functions unrelated to the platform. On the basis of the core library, the interface layer plug-in corresponding to each platform is developed. Thanks to the cross-platform feature of the Qt framework, more than 80% of the code achieves multiplexing. According to the different requirements of different platforms, macro definition and conditional compilation mechanisms are adopted for personalized processing. At the same time, GoogleTest was introduced to improve the unit test of the core library, which significantly improved the code quality. Platform integration and joint adjustment are the difficulties of the project. For example, when the iOS platform connects with the system album, message push and other services, we encountered a lot of pits. You need to consult the Apple developer document and debug it for many times. After nearly a month of joint adjustment, the project has finally run steadily, and has been successively launched on Windows, macOS, iOS, Android, Web and other platforms. After the launch, I continued to pay attention to the operation data of various platforms, collect user feedback, and make iterative plans to carry out a new round of development according to the needs of business development.

Through the practice of this project, the author realized that the advantage of cross-platform software development is that the integration of multi-platform version into one project through highly reuse code, significantly improving the efficiency of development and maintenance. But at the same time, cross-platform projects also put forward higher requirements on the complexity of technology and management, requiring the development team to have full-stack skills and be familiar with the differences of different platforms. Only by building an agile and efficient R & D system can we navigate the increasingly complex cross-platform software engineering.

IV. Epilogue

In conclusion, the cross-platform software development based on C++ language can achieve the ideal goal of "Write once, run anywhere". By optimizing the development process and rationally using the cross-platform framework and tools, it can not only significantly reduce the development and maintenance costs, but also broaden the coverage of the software, laying the foundation for maximizing the vitality and commercial value of the software. Of course, cross-platform development is not a "silver bullet". While ensuring the development efficiency, we should also be alert to the risk of inconsistent experience brought about by platform differentiation. Only by accurately doing the demand analysis well, and taking into account the personalized design of different platforms, can we go more steadily and further on the cross-platform road. In the future, with the development of a new generation of cross-platform frameworks such as Flutter and React Native, the cross-end forms of software will be more abundant. As developers, we need to keep pace with The Times, explore new practices and new opportunities for cross-platform development, and bring more extreme product experience to users with innovative technologies.

References

[1] Kak A C .Programming with Objects: A Comparative Presentation of Object-Oriented Programming with C++ and Java[M]. Hoboken: John Wiley & Sons, Inc, 2011.

[2] Smith E. Introduction to the Tools of Scientific Computing[M]. Cham: Springer, 2021.

[3] Robert P, Matú D. Implementation C++/QT framework for CAN communication[J]. Transportation Research Procedia, 2023, 74:946-953.