Article

Research on Collaborative Development Mode of C# And Python in Medical Device Software Development

Minkang Zhang [1,*]

[1] Bio-Rad Laboratories, Clinical Diagnostics Group (CDG) Software Team, CA, 94547, USA
* Correspondence: Minkang Zhang, Bio-Rad Laboratories, Clinical Diagnostics Group (CDG) Software Team, CA, 94547, USA

Abstract: C# and Python are two widely used programming languages, each with unique advantages, and they play a crucial role in the development of medical device software. C# is particularly effective in designing and implementing graphical user interfaces (GUIs) and managing core business logic, providing a stable and high-performance environment for software front-end and system integration. Python, on the other hand, excels in data analysis, scientific computing, algorithm development, and machine learning, offering flexibility and rapid prototyping capabilities that are essential for advanced data-driven functionalities in medical devices. This paper presents a comprehensive study on the joint development model of C# and Python, focusing on the design of an effective cooperative framework that leverages the strengths of both languages in the field of medical device software development. The proposed framework provides a structured approach for integrating front-end GUI design, core process management, and advanced computational modules, and details the interaction and cooperation between C# and Python during the development lifecycle with specific task-oriented guidance. To validate the proposed approach, integration testing and performance evaluation were conducted to ensure the reliability, efficiency, and robustness of the system. The results demonstrate that the combined use of C# and Python not only improves software performance and maintainability but also enhances the flexibility and scalability of medical device applications. Finally, the practical implementation of this joint development model in real-world projects confirms its effectiveness, providing a valuable reference for future development of high-quality, data-driven, and user-friendly medical device software.

Keywords: C#; Python; medical device software; joint development framework; data analysis; GUI design

Received: 13 November 2025
Revised: 22 November 2025
Accepted: 02 December 2025
Published: 09 December 2025

1. Introduction

With the continuous advancement of medical technology, the development and application of medical device software have become increasingly critical for ensuring the quality of patient care, enhancing diagnostic accuracy, and optimizing treatment plan design. Modern medical devices demand software systems that are highly functional, efficient, well-structured, scalable, and easy to maintain, while also meeting stringent safety and regulatory requirements.

In this context, C# and Python, two programming languages with distinct advantages, play complementary roles in medical device software development. C# is particularly well-suited for designing user-friendly graphical interfaces and managing the control of instrument functions, offering high performance, stability, and integration capabilities for core system operations. Python, on the other hand, excels in data analysis, machine learning, algorithm development, and rapid prototyping, providing flexibility and computational power for advanced functionalities such as predictive modeling, real-time data processing, and intelligent decision support [1].

The effective integration of C# and Python enables developers to leverage the strengths of both languages, improving the efficiency of software development, enhancing system reliability, and addressing quality control challenges in medical device applications. However, achieving a reasonable and optimized division of labor between the two languages, coordinating their interactions, and ensuring smooth system interoperability remain key challenges. This article aims to explore a joint development framework that maximizes the synergistic potential of C# and Python, offering practical guidance for accelerating medical device software development while maintaining high standards of performance, reliability, and maintainability [2].

2. C# And Python Language Features

2.1. Language Features of C#

C# is a high-performance object-oriented programming language developed by Microsoft, often used for Windows software based application development, characterized by efficiency, simplicity and ease of use. The biggest feature is that the type system is strict, which can ensure the validity of types at compile time and greatly reduce the possibility of type problems at run time. In addition, C# also supports object-oriented programming ideas, can inherit, encapsulate, polymorphism, and facilitate developers to organize and reuse code. At the same time, C# has a built-in garbage collector to automatically allocate and recycle memory to prevent memory leaks. In the face of parallelization, C# provides a powerful consistent multi-threading, which makes it easy for developers to carry out efficient asynchronous programming, which is more suitable for the writing of applications with strong requirements for high performance. Because of its easy maintenance, debugging characteristics, C# is usually used for large-scale enterprise applications, desktop applications and Internet applications development, more suitable for the development of complex user interface UI, control of various devices applications. Sure, C# can be used. NETCore technology to achieve cross-platform development, can run in Windows, Linux, MacOS and other platforms, further enhance the actual development scope of C# [3].

2.2. Python Language Features

Python is a high-level and analytical scripting language that is concise, easy to read, and efficient. Python uses a concise and intuitive syntax that allows complex operations to be performed with a small amount of code, so Python is used in fields such as data science, AI, web development, and more. Python's dynamically typed architecture, in which the type of a variable is not determined until run time, makes the code more flexible. Python has a large number of predefined libraries that meet various scenarios, such as file processing, web page compilation, database connection, and so on. At the same time, rich third-party libraries, especially for a large number of applications in the field of data science, such as data analysis, scientific computing and machine learning, further enhance the powerful functions of Python, such as NumPy, Pandas, TensorFlow, etc., making it extremely easy to use. Python supports hybrid programming, such as object Oriented programming (OOP) and functional programming (FP), resulting in high development efficiency [4].

3. Design of Collaborative Development Mode between C# And Python

3.1. Collaborative Development Framework

In the development of medical equipment software, it is necessary to effectively combine the advantages of C# and Python and have good flexibility and maintainability. Because C#, a statically strongly typed language, is easy to design high-quality GUIs, operating hardware, and business rules parts, Python shows great advantages in data management, algorithmic programming, and machine learning. Therefore, in order for C# to interface seamlessly with Python, the framework should adopt a microservice architecture, splitting each function into separate services, each service is only focused on a specific task, for example C# is responsible for generating the user interface and obtaining real-time data, and Python is responsible for data parsing and modeling. Use the RESTfulAPI or gRPC interface to let C# and server-side Python modules access and exchange information with each other. In addition, the framework also needs to have a good scalability and maintainability, in order to cope with future project changes or technical changes brought about by the function of a module. Finally, if the team is able to work well with each other, the framework should also include complete interface descriptions and uniform coding specifications so that different programmers can work smoothly [5].

3.2. Data Exchange Mechanism

In the collaborative development of C# and Python, the data exchange mode ensures the smooth cooperation of functional modules. Common data exchange methods include JSON, XML and Protobuf, among which JSON is the most common communication between C# and Python because of its simplicity and easy parsing. C# sends the data via JSON to Python, which parses the data through the corresponding library, such as the json module. XML is suitable for the exchange of complex data structures, especially when there are many levels of data. Message queuing (such as RabbitMQ or Kafka) is an efficient, asynchronous way to exchange data, avoiding the performance bottlenecks of traditional synchronous communication, and is suitable for high concurrency scenarios. It guarantees the orderliness and stability of the message, and ensures the stability and efficiency of the system in real-time data processing. In addition, C# and Python modules can also store and read data through a shared database (such as MySQL/MongoDB), which is suitable for application scenarios where large amounts of historical and real-time data need to be stored. Because the database can both record data and efficiently search and update data, it is very suitable for dealing with the huge storage of historical data and real-time requirements of medical equipment software, as shown in Table 1.

Table 1. Describes the data exchange mechanism and application scenarios.

Data exchange mechanism	Description	Application scenario
JSON	Lightweight data format, easy to parse, widely used for cross-language communication	Suitable for simple and small data exchanges, such as communication between C# and Python
XML	Text storage, easy to establish hierarchical relationships, suitable for complex structured data exchange.	For complex data exchange, data that requires a clear hierarchy
Protobuf	Binary format, provides high performance and compact size, suitable for low latency data transfer	When efficient, low-latency data transfer is required, especially between microservices
Message queue	Asynchronous messaging system for breaking coupling between modules and resolving high concurrency	Used in high-concurrency scenarios to ensure non-blocking communication
Archive	The shared storage system can efficiently record and read massive data	It is suitable for storing large amounts of historical data and real-time data, especially data access in medical device software

Based on the table above, it can be seen that these data exchange mechanisms can meet the requirements in different scenarios through their respective advantages.

3.3. Cross-Language Communication

One of the most important and difficult aspects of mixed programming in C# and python is the effective communication between the two languages. Two common ways to interact with C# and python are RESTful apis, gRPC, and shared memory. The RESTfulAPI is usually used, C# modules can provide restful service apis using the http protocol, and Python modules use http requests to call apis for communication. For more efficient communication and lower latency, you can choose the gRPC protocol, which requires ProtocolBuffers (Protobuf). This protocol has the advantages of strong performance and cross-language protocol. gRPC can also support HTTP/2. There is also the ability of gRPC to have two two-way streams, which is most effective for frequent data update applications. If you want to pursue better performance, you can choose shared memory and custom message processing logic, so that the data transmission between C# and Python modules is more smooth and complete the task smoothly, improve the system response speed and reliability. Table 2 shows common communication mechanisms in C# and Python co-development, including RESTful apis, gRPC, and shared memory, and details the features, benefits, and scenarios for each mechanism.

Table 2. Advantages and application scenarios of cross-language communication.

Communication mechanism	Description	advantage	Application scenario
RESTful API	Based on the HTTP protocol communication mode, C# provides a RESTful service interface, and Python accesses it through HTTP requests	It is convenient to implement and easy to debug	It is suitable for small - and medium-sized data exchange scenarios with low latency requirements
gRPC	High performance communication mode based on Protobuf protocol, support HTTP/2 and two-way traffic, suitable for real-time data update	Efficient speed, low delay	Suitable for scenarios requiring high-frequency data exchange, real-time updates, and low latency
Shared memory	The shared memory is used for direct communication without network transmission, and is suitable for scenarios that require high performance	Efficient and fast communication mode, suitable for low latency and high throughput tasks, reducing data transmission delay	It is suitable for high-performance scenarios that require high real-time performance and large amount of data

It can be concluded from the above table that the reasonable selection and combination of these mechanisms can achieve efficient communication between C# and Python modules, and meet the performance requirements in different scenarios.

3.4. Microservice Architecture

Microservices architecture is a design method for breaking down a large software architecture into multiple independent and autonomous services. Each service performs a specific task and can be developed, tested, installed, and upgraded independently. The combination of C# and Python programming means that C# and Python function modules can be targeted in the microservice architecture, so that they can execute independently and communicate with each other through apis or message channels. The main characteristics of microservices architecture are high flexibility and easy maintenance. Since each module operates independently, module expansions and modifications do not affect other modules. For example, if one of the Python modules needs to be more efficient, you can just modify it without interfering with the C# module. In addition, the microservice architecture is also suitable for container deployment environments, and C# and Python module services can be packaged using Docker containers and run and control independently in different environments. The use of microservice architecture model can make the medical device software have good modular support, realize the cooperation of modules, reduce the complexity of system structure and cause technical debt.

4. Practice Path of Collaborative Development Mode in Medical Device Software Development

4.1. Allocation of Module Functions Based on C# And Python Advantages

In the development of medical equipment software, the collaborative development of C# and Python should first allocate module functions reasonably according to their respective advantages. Considering that C# has a powerful effect in GUI creation, but also can well control the hardware device and a large number of concurrent operations, so it is more suitable for the front desk user interaction, device interface management and real-time data processing and other functions; Python has great advantages in big data algorithms and machine learning, especially in medical data processing, image analysis, and model establishment. Through reasonable functional division, the expertise of both can be brought into play to ensure the efficient and sustainable development of the system. In short, the C# part is mainly to achieve server-side or client-side GUI interface generation and interaction with users and devices to ensure efficient and fast user response. The Python part mainly implements a large number of calculations on the back-end server side, such as image processing, data construction, and predictive analysis. This design ensures that each part avoids the duplication of functions, reduces the difficulty of module integration, and improves the work efficiency.

4.2. Selecting a Suitable System Integration Solution

In C# and Python co-development, choosing the right integration method is the key to ensure effective collaboration between modules of the two programming languages. Common integration methods include RESTfulAPI, gRPC, messaging, shared database, etc. Each integration method has its own applicable scenarios and requirements. The RESTfulAPI is a simple, generic integration approach for situations where formal request and answer interactions are required. C# uses HTTP to provide an API interface, and Python uses HTTP requests for information exchange. This method is easy to implement and easy to debug, and is suitable for user interface interaction and device management. gRPC is more suitable for scenarios that require extreme efficiency. The HTTP/ 2-based gRPC combined with Protobuf encoding can achieve faster transmission speed and lower latency, which is suitable for applications where a large number of data transfers occur, such as real-time monitoring and device refresh. Messaging (RabbitMQ, Kafka, etc.), suitable for concurrent, non-synchronous processing. C# and Python modules are run separately, which can avoid the performance lag caused by synchronous work and ensure that the system still runs normally under high load. An integrated database is another form of integration that allows you to use a database to pass data between C# and Python modules, thus ensuring data consistency within the system. Proper selection of integration options can facilitate C# and Python collaboration to improve the efficiency and reliability of medical device apps.

4.3. Integration and Performance Testing of Module Collaboration

In the development of medical equipment software, the integration of modules and performance testing are the key links to ensure the efficient and stable operation of the system. Since C# and Python are jointly encoded, it is necessary to carefully check the information transmission and data exchange between different modules in the two programming languages, so as to verify the integration and performance of the entire system. The main purpose is to confirm that C# and Python module indirect port calls are correct, whether the transmission data is appropriate. The research team first tested C# and Python modules separately, confirming in the unit test that the functions of both modules had been achieved. On this basis, the data transmission and action are simulated under the state of field work, and whether C# and Python modules can be coupled properly is checked. Finally, the focus of attention is on the pressure test of the system, especially the test of system operation when the software of medical equipment often deals with large data streams and real-time processing, so as to find out whether C# and Python modules can cooperate well to ensure that the system can still run stable in large-scale parallel computing and high data concentration. The optimization method includes the optimization of Python data management processing process, using asynchronous operation or GPU acceleration method to improve the efficiency of data management; For C#, the drawing of the user interface can be optimized and user actions can be returned to avoid the situation of interface deadlock.

4.4. Iterative Development and Continuous Improvement of Medical Equipment Projects

In the development process of medical equipment project, selecting the appropriate development process and continuous improvement mechanism is the key to ensure the success of the project. The iterative development process, which is the early selection of the development phase and the design of a clear project objectives, technical approaches, and resource placement. Understand the general development line and target strategy of the whole project, and clarify the content of later research. During the implementation phase, the development team carries out specific functional design and practical application according to the pre-development plan to ensure that the design and application of the realized functional modules meet certain requirements. Continuous verification and testing are required to ensure that the generated functions and systems meet quality standards and have certain reliability. Select the review stage, complete the inspection of the project development process, complete the project on schedule and with high quality, find out the problems and make corresponding adjustments or improvements. At the same time, it is also adopting the principle of continuous improvement for long-term victory. Quality improvement will continuously obtain feedback and feed back to the system to make the system more stable and efficient, so that the medical device software can maintain excellent field use results. In the same way, feature optimization also enhances the functional experience of the software with each iteration to optimize the user experience. Deeper technological innovation can also be achieved through some new technologies such as AI and big data analysis to achieve intelligent and automatic execution of medical equipment software, so as to make products keep pace with The Times. Second, user feedback and adaptation mechanisms enable medical device software to respond to the changing business and technical environment in a timely manner to meet customer needs and adjust. By effectively handling customer feedback and flexibly responding to market changes and technological developments, medical device projects continue to have a competitive advantage, consistently deliver services, and ensure their long-term medical availability and stability (see Figure 1).

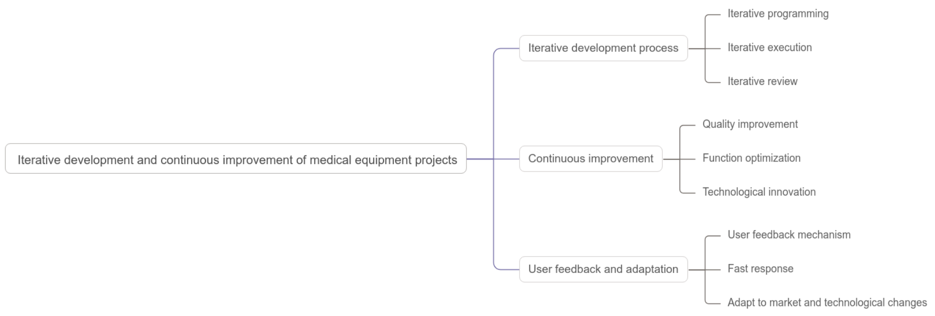

Figure 1. Iterative development and continuous improvement of medical equipment projects.

5. Conclusion

Based on the analysis of the joint application of C# and Python in medical device software development, this study proposes specific design principles and implementation strategies to optimize the performance and efficiency of such integrated systems. Key aspects include reasonable module allocation, the selection of cross-language communication schemes, thorough system integration and performance testing, and pilot verification procedures. These measures collectively ensure that the collaborative use of C# and Python maximizes the advantages of both languages, enhances software reliability, and supports the development of high-quality, user-friendly medical instrument applications.

With the continuous advancement of technology, the integration of C# and Python in medical device software is becoming increasingly prevalent, reflecting its growing importance in the field. This collaborative programming approach not only accelerates the development process but also facilitates more sophisticated functionalities, such as advanced data analysis, machine learning integration, and intelligent control systems. Furthermore, it contributes to improving software scalability, maintainability, and adaptability, which are crucial for meeting the evolving requirements of modern medical devices.

In conclusion, the joint programming of C# and Python represents a promising and practical approach for enhancing medical instrument software. Its application supports the advancement of medical technology, promotes the efficiency and accuracy of medical operations, and provides a solid foundation for future innovations in intelligent, data-driven medical systems.

References

1. F. Su, J. Wang, R. Lei, and D. Ren, "RETRACTED ARTICLE: Optical medical equipment diagnosis and clinical efficacy of medication in the treatment of autoimmune thyroiditis," *Optical and Quantum Electronics*, vol. 56, no. 3, p. 297, 2024. doi: 10.1007/s11082-023-05998-w
2. A. Hotz, E. Sprecher, L. Bastianelli, J. Rodean, I. Stringfellow, E. Barkoudah, and J. G. Berry, "Categorization of a universal coding system to distinguish use of durable medical equipment and supplies in pediatric patients," *JAMA Network Open*, vol. 6, no. 10, pp. e2339449-e2339449, 2023.
3. N. Assavakamhaenghan, W. Tanaphantaruk, P. Suwanworaboon, M. Choetkiertikul, and S. Tuarob, "Quantifying effectiveness of team recommendation for collaborative software development," *Automated Software Engineering*, vol. 29, no. 2, p. 51, 2022. doi: 10.1007/s10515-022-00357-7
4. D. Grana, and L. De Figueiredo, "SeReMpy: Seismic reservoir modeling Python library," *Geophysics*, vol. 86, no. 6, pp. F61-F69, 2021.
5. S. Roengtam, and A. Agustiyara, "Collaborative governance for forest land use policy implementation and development," *Cogent social sciences*, vol. 8, no. 1, p. 2073670, 2022. doi: 10.1080/23311886.2022.2073670