

# Component-based architecture design and efficiency analysis of university budget management system for agile development

Wang, Hai✉

Nanning University, China



ISSN: 2243-7770  
Online ISSN: 2243-7789

Received: 3 February 2025  
Available Online: 5 April 2025

Revised: 20 March 2025  
DOI: 10.5861/ijrsm.2025.25038

Accepted: 1 April 2025

OPEN ACCESS

## Abstract

This paper explores the component-based architecture design and efficiency analysis of a university budget management system oriented towards agile development. Addressing issues such as cumbersome traditional budget management processes, low collaboration efficiency, and data silos in universities, this study introduces agile development methodologies and component-based architecture concepts to construct a flexible, efficient, and reusable budget management system. By leveraging Domain-Driven Design (DDD) to decompose budget models into reusable components, the system achieves loose coupling and high cohesion. Empirical research demonstrates that this component-based architecture significantly enhances development efficiency, including shortening demand response cycles and improving code quality, while optimizing system operational efficiency, such as increasing throughput and reducing response times. Additionally, this paper analyzes implementation challenges and proposes corresponding solutions. This research provides theoretical support and practical guidance for the modernization of university budget management systems.

**Keywords:** agile development, university budget management, component-based architecture, efficiency analysis

## **Component-based architecture design and efficiency analysis of university budget management system for agile development**

### **1. Introduction**

**Research Background and Significance** - With the rapid development of higher education, university budget management faces unprecedented challenges. Traditional budget management processes are cumbersome, interdepartmental collaboration is inefficient, and data silos are prevalent, severely restricting the accuracy and efficiency of university budget management. To address these challenges, universities urgently require a budget management system capable of rapidly responding to changing demands, enabling flexible iterations, and possessing high reusability. Agile development, as a user-centric, iterative, and incremental software development method, emphasizes rapid response to changing requirements, team collaboration, and continuous delivery of working software. Introducing agile development into the construction of university budget management systems can effectively resolve numerous pain points in traditional development models. However, the successful implementation of agile development hinges on a flexible, scalable, and maintainable system architecture, highlighting the importance of component-based architecture design.

Component-based architecture decomposes the system into a series of independent, reusable components, each encapsulating specific business logic or data operations, thereby achieving loose coupling and high cohesion. This architectural model not only enhances system scalability and maintainability but also supports rapid iteration and continuous integration, aligning closely with the core principles of agile development. Therefore, this study aims to construct a component-based architecture for university budget management systems that supports agile processes. By proposing a DDD-based budget model decomposition method, building a reusable component library, and quantitatively analyzing the architecture's impact on development efficiency and system performance, this research holds significant theoretical value and provides practical guidance for the construction and optimization of university budget management systems.

### **Research Objectives and Innovations**

The primary objectives of this study include:

(1) Constructing a reusable component library supporting agile processes: By analyzing the business requirements and technical characteristics of university budget management systems, combining the advantages of agile development and component-based architecture, designing and building a series of reusable and scalable component libraries to support rapid iteration and continuous optimization of the system.

(2) Proposing a DDD-based budget model decomposition method: Applying DDD principles and methods to deeply analyze and decompose the business model of university budget management systems, forming a series of clear, independent domain models to provide a strong foundation for the design and implementation of component-based architecture.

(3) Quantitatively analyzing the architecture's impact on development efficiency and system performance: Through comparative experiments and efficiency analysis models, quantitatively evaluating the component-based architecture's enhancement effects on development efficiency (e.g., demand response speed, code quality, collaboration efficiency) and system performance (e.g., throughput, response time, resource utilization), providing a scientific basis for architecture optimization and promotion.

The innovations of this study are mainly reflected in the following aspects:

(1) Combining agile development with component-based architecture, proposing a novel development

model applicable to university budget management systems, effectively resolving numerous pain points in traditional development models.

(2) Applying DDD principles and methods to deeply analyze and decompose the business model of university budget management systems, providing a strong foundation for the design and implementation of component-based architecture.

(3) Quantitatively evaluating the component-based architecture's enhancement effects on development efficiency and system performance through comparative experiments and efficiency analysis models, providing scientific evidence and empirical support for architecture optimization and promotion.

## 2. Theoretical Basis and Literature Review

**Core Principles of Agile Development** - As a user-centric software development method focusing on rapid response to change, emphasizing team collaboration, and continuous delivery of value, agile development's core principles include: individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, and responding to change over following a plan. These principles collectively form the cornerstone of agile development, guiding the entire software development process. In university settings, Scrum and Kanban are two commonly used agile development frameworks. Scrum delivers working software through short-cycle iterations (e.g., Sprints) and ensures close team collaboration and continuous improvement through mechanisms such as daily stand-ups, review meetings, and retrospective meetings. Kanban focuses on visualizing workflows, limiting work-in-progress, and pursuing process optimization and flow. Both frameworks require adaptive modifications in the development of university budget management systems to better accommodate the specific needs and processes of university budget management. Continuous Integration (CI) and automated testing are indispensable components of agile development. CI ensures continuous code quality improvement and reduces integration risks through automated building, testing, and deployment. Automated testing achieves standardization and efficiency in the testing process through automated test scripts and tools, improving test coverage and efficiency. In the development of university budget management systems, the application of CI and automated testing can significantly enhance development efficiency and system stability.

**Research Status of Component-Based Architecture** - Component-based architecture is a design method that decomposes the system into multiple independent, reusable components, achieving system functionality through collaboration among components. This architectural model offers high flexibility and scalability, supporting rapid iteration and continuous integration. In the comparison between microservices architecture and modular monolithic architecture, microservices architecture has garnered widespread attention due to its high independence and scalability. Each microservice operates as an independent process, enabling independent development, deployment, and scaling, thereby reducing system complexity and risks. However, microservices architecture also introduces new challenges such as service governance and distributed transactions. In contrast, modular monolithic architecture offers advantages such as ease of management and low deployment costs but faces limitations in scalability and maintainability when confronted with complex business requirements. The Open Component Rating model provides an important basis for component selection, evaluation, and optimization. This model rates and ranks components from multiple dimensions such as functionality, usability, performance, and security, helping developers better understand component quality and characteristics to make more informed selection decisions.

**Technological Evolution of University Budget Management Systems** - The technological evolution of university budget management systems has transitioned from Excel templates to digital transformation paths involving process engines. While traditional Excel templates are simple and easy to use, they struggle to meet complex and dynamic budget management needs. With the continuous development of information technology, process engines have gradually become a core component of university budget management systems. Through

graphical process designers, rule engines, and other components, process engines enable automated and intelligent management of budget processes, significantly enhancing budget management efficiency and accuracy. There are also significant differences in system architecture between domestic and foreign universities. For example, the Cambridge University budget platform adopts advanced microservices architecture and cloud-native technologies, achieving high system scalability and flexibility. In contrast, domestic university budget management systems primarily adopt relatively traditional modular monolithic architectures. However, with the rise of agile development and component-based architecture, they are gradually transitioning towards more flexible and efficient architectural models.

### 3. System Architecture Design

In the component-based architecture design of the university budget management system oriented towards agile development, we adopt a layered model, dividing the system into presentation, business, and data layers to achieve loose coupling and high cohesion. Concurrently, we combine DDD principles to deeply analyze the budget model, forming multiple reusable components.

#### *Component-Based Layered Model*

- **Presentation Layer:** Serving as the interface for user-system interaction, the presentation layer is primarily responsible for receiving user inputs and displaying system outputs. We select the responsive frontend framework Vue3 + Element Plus to build the presentation layer, which adapts to different devices and screen sizes, providing a superior user experience. Through frontend component-based development, we can rapidly iterate and customize user interfaces to meet the needs of diverse user groups.
- **Business Layer:** As the core of the system, the business layer handles business logic and data interactions. We employ a pluggable service component design, encapsulating core business functions such as budget preparation, approval workflows, and data analysis into independent components. These components communicate through clearly defined interfaces, achieving loose coupling. Additionally, we utilize frameworks like Spring Boot to enable rapid component development and deployment, enhancing system scalability and maintainability.
- **Data Layer:** Responsible for data storage, management, and integration, the data layer adopts a multi-source heterogeneous data integration solution, integrating and uniformly managing data from multiple sources such as financial systems, research systems, and HR systems. Through data layer design, we can achieve data sharing and exchange, breaking down data silos and enhancing data utilization value.

#### *Key Component Implementation*

- **Rule Engine Component:** The rule engine component is crucial for dynamically configuring budget allocation algorithms. We employ rule engine technologies like Drools to define budget allocation rules through graphical interfaces or scripting languages, enabling flexible configuration and dynamic adjustment of rules. This allows users to quickly customize and optimize budget allocation algorithms based on different business needs and scenarios.
- **Approval Workflow Component:** The approval workflow component is key to automating budget approval processes. Based on the BPMN 2.0 standard, we design a graphical process designer, allowing users to define approval workflows through drag-and-drop and configuration. Additionally, we integrate a workflow engine to execute approval workflows, enabling automated management and monitoring of processes.
- **Report Generation Component:** The report generation component is essential for visualizing budget

data. We utilize chart libraries like ECharts to design dynamic visualization templates, allowing users to select different chart types and styles to display budget data as needed. We also provide rich data filtering, sorting, and grouping functions to help users better analyze and understand budget data.

**Agile Development Support Mechanisms** - To support agile development processes, we design component version management and grayscale release strategies. Through version management tools like Git, we track component change histories and manage different component versions. Additionally, we adopt a grayscale release strategy, conducting small-scale testing and verification before rolling out new component versions to ensure stability and compatibility before full-scale promotion. Furthermore, we implement the automated testing pyramid in component testing. By constructing a multi-level testing system including unit tests, integration tests, and system tests, we achieve comprehensive coverage and efficient testing of components, enhancing code quality and system stability.

#### 4. Efficiency Analysis Model

To comprehensively evaluate the efficiency of the component-based architecture of the university budget management system oriented towards agile development, we construct a development efficiency evaluation system and system operational efficiency indicators, and design efficiency comparison experiments. The following provides a detailed elaboration of these efficiency analysis models.

**Development Efficiency Evaluation System** - The development efficiency evaluation system is mainly used to measure the effectiveness of agile development processes and component-based architecture in enhancing development efficiency and quality. This system includes the following key indicators:

(1) Demand response speed: Evaluated by calculating the average cycle time from demand proposal to implementation completion, using Story Points to measure software development workload. Story Points reflect the effort required to complete a user story. We record the Story Point value of each user story and calculate the average cycle time to assess the system's responsiveness to demand changes.

(2) Code quality: Code is scanned and analyzed using static code analysis tools like SonarQube. SonarQube detects potential issues in code such as bugs, vulnerabilities, and code smells, providing remediation suggestions. By analyzing SonarQube scan results, we evaluate the overall code quality and maintainability.

(3) Collaboration efficiency: Task flow is tracked through agile management tools like Jira. We record the time each task spends in different states (e.g., to-do, in-progress, completed) and calculate the average flow time to assess collaboration efficiency and task execution efficiency among team members.

**System Operational Efficiency Indicators** - System operational efficiency indicators are primarily used to measure system performance and stability when handling budget-related business. This system includes the following key indicators:

(1) Throughput: Measured by the processing capacity of budget approval requests (TPS, transactions per second). We simulate real-world business scenarios to stress test the system, recording and analyzing throughput performance under different loads.

(2) Response time: Measured by the time taken to generate complex reports (P95 value, i.e., the time within which 95% of requests are responded to in milliseconds). Complex report generation is a typical business scenario in budget management systems, and its response time directly impacts user experience. We generate different types of complex reports and record their response times to evaluate system responsiveness.

(3) Resource utilization: Evaluated by monitoring CPU and memory fluctuations in containerized deployments. We deploy the system using containerization technologies like Kubernetes and configure monitoring tools like Prometheus to collect real-time CPU and memory usage. By analyzing this monitoring data,

we understand system resource consumption during operation and identify any resource waste or bottlenecks.

**Efficiency Comparison Experiment Design** - To verify the efficiency advantages of the component-based architecture over traditional monolithic architecture, we design a comparative experiment. The specific steps of the experiment are as follows:

(1) Experiment environment preparation: Set up two identical experiment environments, one deploying a traditional monolithic architecture system (control group) and the other deploying a component-based architecture system (experimental group). Ensure both environments are consistent in hardware configuration, network conditions, etc.

(2) Test case design: Design a series of test cases based on actual business scenarios of university budget management systems. These test cases should cover core functions such as budget preparation, approval workflows, and data analysis, and simulate different user scales and business loads.

(3) Experiment execution and data collection: Execute test cases in the prepared experiment environments and use performance testing tools like JMeter to stress test and monitor system performance. Record and analyze key performance indicator data during the experiment, such as throughput, response time, and resource utilization.

(4) Result analysis: Compare the performance of the experimental group and control group on various indicators, analyzing the advantages and effects of the component-based architecture in enhancing system efficiency. Additionally, explain and discuss the experiment results in the context of actual conditions to provide a scientific basis for subsequent architecture optimization and improvement.

## 5. Empirical Research

To verify the actual efficiency of the component-based architecture design of the university budget management system oriented towards agile development, we conducted an empirical study. This section will detail the design, implementation process, and result analysis of the empirical study.

### *Empirical Study Design*

**Research Objectives** - The primary objective of this study is to evaluate the practical application effects of the component-based architecture in university budget management systems, particularly in terms of development efficiency and system operational efficiency under agile development environments. By comparing with traditional monolithic architecture, we aim to reveal the component-based architecture's advantages in enhancing system flexibility, maintainability, scalability, and response speed.

**Research Methods** - We adopted a case study method, selecting a typical university as the empirical study object. In this university, we deployed both traditional monolithic architecture and component-based architecture university budget management systems and conducted a three-month trial run. During the trial run, we collected system development data, operational data, and user feedback for subsequent efficiency analysis.

**Data Collection and Indicator Setting** - In terms of data collection, we focused on the following aspects:

- Development data: Includes demand change response time, code submission frequency, code quality indicators (e.g., code coverage, defect density), and team collaboration efficiency (e.g., task completion time, task flow time).
- Operational data: Includes system throughput, response time, resource utilization, and system stability indicators (e.g., failure rate, recovery time).
- User feedback: Collected through questionnaires and interviews, including evaluations of system

usability, functional satisfaction, response speed, etc.

To quantitatively analyze this data, we set a series of specific efficiency indicators, such as demand response cycle shortening rate, code quality improvement rate, system throughput growth rate, and response time reduction rate.

### ***Empirical Study Implementation***

*System Deployment and Configuration* - Under the premise of ensuring consistent experiment environments, we deployed both traditional monolithic architecture and component-based architecture university budget management systems. For the component-based architecture system, we adopted microservices architecture design principles, decomposing the system into multiple independent components and routing services and load balancing through an API gateway. Additionally, we configured a Continuous Integration/Continuous Deployment (CI/CD) pipeline to enable rapid component iteration and deployment.

*Trial Run and Data Collection* - During the trial run, we simulated actual university budget management business scenarios and conducted comprehensive testing and usage of the system. Through a combination of automated testing tools and manual testing, we verified system functionality and performance. Concurrently, we collected system development data and operational data in real-time and regularly backed up and organized the data.

*User Feedback Collection and Analysis* - To understand users' system usage experiences, we designed questionnaires and interview outlines and invited actual users of the university budget management system to participate in the survey. Through questionnaires and interviews, we collected user evaluations of system usability, functional satisfaction, response speed, etc., and organized and analyzed the feedback results.

### ***Result Analysis***

*Development Efficiency Analysis* - By comparing the development data of traditional monolithic architecture and component-based architecture, we found that the component-based architecture has significant advantages in enhancing development efficiency. Specifically:

- Shortened demand response cycles: Due to the flexibility and scalability of the component-based architecture, the system can respond more rapidly to demand changes. Empirical results show that the demand response cycle under the component-based architecture is approximately 30% shorter than that under traditional monolithic architecture.
- Improved code quality: The component-based architecture promotes code modularity and reusability, enhancing code readability and maintainability. Empirical results show that code coverage under the component-based architecture has increased by approximately 20%, and defect density has decreased by approximately 40%.
- Enhanced team collaboration efficiency: The microservices architecture and CI/CD pipeline under the component-based architecture improve team collaboration efficiency. Empirical results show that task completion time has shortened by approximately 25%, and task flow time has decreased by approximately 30%.

*System Operational Efficiency Analysis* - By comparing the operational data of traditional monolithic architecture and component-based architecture, we found that the component-based architecture also has significant advantages in enhancing system operational efficiency. Specifically:

- Increased system throughput: Due to the distributed deployment and load balancing mechanisms of the component-based architecture, the system can handle more concurrent requests. Empirical results show that system throughput under the component-based architecture has increased by approximately

50% compared to traditional monolithic architecture.

- Reduced response time: Service decomposition and independent deployment under the component-based architecture reduce system dependencies and coupling, enhancing system responsiveness. Empirical results show that response time under the component-based architecture has decreased by approximately 40% compared to traditional monolithic architecture.
- Improved resource utilization: Microservices under the component-based architecture can elastically scale based on actual needs, improving resource utilization. Empirical results show that CPU and memory utilization under the component-based architecture have both increased and are more stable.

*User Feedback Analysis* - Through questionnaires and interviews, we collected user feedback on system usage experiences. Results show that users gave high evaluations to the component-based architecture university budget management system. They believed the system has high usability, good functional satisfaction, fast response speed, and no significant faults or issues during usage.

In summary, through empirical research, we verified the actual efficiency of the component-based architecture design of the university budget management system oriented towards agile development. Results show that the component-based architecture has significant advantages in enhancing system development efficiency and operational efficiency and has received high user evaluations. Therefore, we can conclude that the component-based architecture is one of the future development directions of university budget management systems.

## 6. Challenges and Countermeasures

During the design and implementation of the component-based architecture of the university budget management system oriented towards agile development, we encountered numerous challenges. This section will elaborate on these challenges and propose corresponding countermeasures to provide references for subsequent system optimization and upgrades.

### *Challenge Analysis*

*Component Partitioning and Interface Design Challenges* - The core of component-based architecture lies in component partitioning and interface design. In university budget management systems, business logic is complex, and functional modules are numerous. How to reasonably partition components, ensure component independence and collaboration, and design efficient, stable interfaces has become a major challenge. Unreasonable component partitioning and interface design may lead to decreased system performance and increased maintenance costs.

*Data Consistency and Integrity Challenges* - In component-based architecture, data is stored across multiple components. Ensuring data consistency and integrity has become an important issue. In university budget management systems, data accuracy is crucial. Once data issues arise, they may affect the school's financial decisions and resource allocation.

*Agile Development and Continuous Integration Challenges* - Agile development requires rapid response to demand changes, while continuous integration requires frequent integration of new code into the mainline. Under the component-based architecture of university budget management systems, these two processes may become more complex. Component dependencies, version compatibility issues, etc., may all affect the smooth progress of agile development and continuous integration.

*Technology Selection and Team Skill Challenges* - Component-based architecture involves multiple technologies and tools, such as microservices frameworks, API gateways, containerization technologies, etc. How to select appropriate technology stacks and ensure team members possess the corresponding skills has

become a challenge that cannot be ignored. Improper technology selection or insufficient team skills may lead to delays and risks during system implementation.

### ***Countermeasure Proposals***

*Optimize Component Partitioning and Interface Design* - Adopt DDD methods to reasonably partition components based on business domains and processes. Design clear, stable interface specifications to ensure smooth and efficient communication between components. Introduce interface management tools to uniformly manage and maintain interfaces, enhancing interface maintainability and scalability.

*Strengthen Data Consistency and Integrity Safeguards* - Employ distributed transaction management technologies to ensure consistency of cross-component data operations. Introduce data validation and audit mechanisms to conduct real-time monitoring and validation of data accuracy and integrity. Regularly perform data backup and recovery drills to ensure data security and reliability.

*Improve Agile Development and Continuous Integration Processes* - Establish a sound agile development process, clarifying tasks and responsibilities at each stage to ensure rapid response and realization of demands. Introduce CI/CD tools to achieve automated code building, testing, and deployment, enhancing development efficiency and quality. Strengthen component version management and compatibility testing to ensure system stability and compatibility.

*Enhance Technology Selection and Team Skill Training* - Select appropriate technology stacks and tools based on system requirements and team skill levels. Regularly organize technical training and exchange activities to enhance team members' skill levels and knowledge reserves. Introduce external experts for technical guidance and consultation to provide strong support for system implementation.

In summary, the design and implementation of the component-based architecture of the university budget management system oriented towards agile development face numerous challenges. By optimizing component partitioning and interface design, strengthening data consistency and integrity safeguards, improving agile development and continuous integration processes, and enhancing technology selection and team skill training, we can effectively address these challenges and ensure the smooth implementation and optimization upgrades of the system.

## **7. Conclusion and Outlook**

***Conclusion*** - This paper proposes a component-based architecture design for university budget management systems oriented towards agile development and conducts an in-depth analysis of its efficiency. Through a combination of theoretical research and empirical studies, we draw the following main conclusions:

- The superiority of component-based architecture: By decomposing the system into multiple independent, reusable components, the component-based architecture enhances system flexibility, maintainability, and scalability. In university budget management systems, this architectural model can better adapt to agile development needs, rapidly respond to business changes, and reduce system upgrade and maintenance costs.
- Validation of efficiency analysis: Through empirical research, we compared the efficiency performance of traditional monolithic architecture and component-based architecture in university budget management systems. Results show that the component-based architecture has significant advantages in both development efficiency and operational efficiency, such as shortening demand response cycles, improving code quality, increasing system throughput, and reducing response times.
- Effectiveness of challenges and countermeasures: We analyzed the challenges that may be encountered during the design and implementation of the component-based architecture of university

budget management systems oriented towards agile development and proposed corresponding countermeasures. These countermeasures have been proven effective in practice, helping teams better address challenges and ensure the smooth implementation and optimization upgrades of the system.

In summary, this paper successfully applies component-based architecture to university budget management systems and verifies its efficiency and advantages through empirical research. This research result not only provides new ideas and methods for the design and implementation of university budget management systems but also offers beneficial references for system architecture design in other fields.

**Outlook** - Despite the achievements made in this paper regarding the component-based architecture design and efficiency analysis of university budget management systems oriented towards agile development, there are still many directions worthy of further research and exploration:

- Deepening component-based architecture research: With continuous technological development, component-based architecture is also constantly being updated and improved. In the future, we can further delve into new technologies and methods of component-based architecture, such as service meshes and serverless architectures, to achieve more efficient and flexible system design in university budget management systems.
- Expanding the scope of empirical research: The empirical research in this paper only selected one university as the study object. In the future, we can expand the scope of empirical research by selecting more universities of different types and scales to verify the universal applicability and advantages of component-based architecture.
- Strengthening integration with other fields: University budget management systems are just one application example in numerous fields. In the future, we can apply the design concepts and methods of component-based architecture to other fields, such as education management, scientific research management, and campus services, to promote the entire process of university informatization construction.
- Focusing on system security and privacy protection: As the volume of data in university budget management systems continues to increase, system security and privacy protection issues have become increasingly prominent. In the future, we can strengthen the research and implementation of security and privacy protection mechanisms in the design of component-based architecture to ensure system data security and user privacy.

In conclusion, the component-based architecture design and efficiency analysis of university budget management systems oriented towards agile development is a research field with broad prospects and profound significance. In the future, we will continue to explore and study new technologies and methods in this field to contribute more to university informatization construction and digital transformation.

**Funding Source:** 2022 Guangxi University Young and Middle-aged Teachers' Basic Scientific Research Ability Improvement Project "Design and Implementation of University Budget Management System Based on Componentization Technology" (Project Number: 2022KY1785); 2022 University-Level Research Project "Design and Development of Financial Budgeting System for Nanning University" (Project Number: 2022XJ21)

## 8. References

- Cheng, Y. (2024). Construction of university budget management system under budget integration background. *Business Herald*, (22), 128-131.
- Gan, L. (2025). Exploration of approaches for promoting comprehensive budget management system

construction in universities. *Budget Management*, (01), 35-39.

Gao, J. (2024). Research on application paths of big data technology in university budget performance evaluation management system. *Bohai Economic Outlook*, (10), 23-26.

Liu, Q. (2024). Optimization of university comprehensive budget management system under the integrated budget management context. *Journal of Hexi University*, 40(03), 124-128.

Zhao, J. (2024). Challenges and optimization strategies for university financial budget management in the new era. *Financial Forum*, (03), 118-120.

