

Decoupling and re-architecture design and implementation of a component-based university budget management system

Wang, Hai ✉

Nanning University, China



ISSN: 2243-7738
Online ISSN: 2243-7746

Received: 3 February 2025
Available Online: 5 April 2025

Revised: 20 March 2025
DOI: 10.5861/ijrset.2025.25001

Accepted: 1 April 2025

OPEN ACCESS

Abstract

This study addresses the high module coupling and poor scalability issues in university budget management systems by proposing a decoupling and re-architecture solution based on component-based technology. The research first analyzes core requirements such as multi-department collaboration, hierarchical approval workflows, and dynamic adjustments in university budget management. It then adopts a microservices architecture and containerized deployment strategy to decompose the system into independent, reusable components. These components interact through standardized interfaces, achieving separation of business logic and data processing, thereby enhancing system flexibility and scalability. The study successfully constructs a component-based university budget management system, demonstrating significant application effects in pilot universities, including a 45%-62% reduction in budget approval cycles and a 58% improvement in cross-departmental collaboration efficiency. Additionally, innovative contributions such as domain-driven component partitioning and configurable workflow engine design are proposed, providing technical support for further optimization and upgrading of university budget management systems. This research is of great significance for promoting the transformation of universities towards data-driven financial governance.

Keywords: university budget management, component-based technology, decoupling and re-architecture, microservices architecture

Decoupling and re-architecture design and implementation of a component-based university budget management system

1. Introduction

Research Background - As a core link in the optimal allocation of educational resources, university budget management is characterized by multi-department collaboration, complex data dimensions, and frequent dynamic adjustments. With the expansion of higher education and increasing demands for financial refinement, traditional budget management systems have gradually exposed issues such as high module coupling, poor scalability, and high maintenance costs. For instance, N University requires switching between multiple independent subsystems for budget preparation, execution, and performance evaluation, leading to severe data silos and redundant data entry. When adding new interdisciplinary projects, the system requires days of downtime for code-level modifications, failing to meet agile management needs. These issues highlight the structural contradiction where information technology iteration lags behind business demands.

Research Significance - Component-based technology provides a "Lego-style" technical architecture for budget management through modular encapsulation, standardized interfaces, and plug-and-play mechanisms. Its core value lies in two dimensions: first, through decoupled design of business and service components, core modules such as budgeting algorithms and approval workflow engines can be independently iterated, compressing traditional system upgrade cycles from monthly to weekly; second, the open ecosystem built on component reuse enables flexible integration with heterogeneous platforms such as financial cloud platforms and research management systems, providing a digital foundation for universities to achieve business-finance integration. This technological transformation not only aligns with the technical requirements of the Ministry of Finance's Budget Management Integration Standards but also provides a technical pivot for universities to transition towards data-driven financial governance.

Domestic and International Research Status - Internationally, component-based technology has been widely applied in enterprise-level systems. For example, SAP has reconstructed ERP core modules using a microservices architecture to support millions of transactions per second. In the education sector, MIT has developed a configurable budget component library to balance departmental budget autonomy with global oversight. Domestic research exhibits a "practice-driven" characteristic: Tsinghua University's budget management system built on Spring Cloud supports multi-campus budget collaboration through component orchestration; Fudan University's "budget component marketplace" model allows third-party developers to access customized analysis plugins. Technological evolution paths indicate that university budget systems are undergoing a paradigm shift from monolithic architectures to SOA, and then to cloud-native component-based architectures, with containerization technologies (Docker) and API gateways maturing as key enablers.

2. System Requirements and Design Objectives

At the beginning of system construction, it is crucial to clarify system requirements and design objectives. The following elaborates on the requirements and objectives of the university budget management system, exploring core needs in multi-dimensional collaboration and full lifecycle management from a business perspective, and analyzing technical requirements to define design goals.

Analysis of University Budget Management Business Needs - University budget management exhibits characteristics of "multi-dimensional collaboration and full lifecycle management," with core needs including:

Multi-department collaboration: Supporting data linkage across 8-12 departments including the Finance Office, Research Office, Personnel Office, Academic Affairs Office, Logistics and Infrastructure Office, and

Student Affairs Office. For example, personnel budgeting requires interfacing with salary data from the HR system, and research project budgets need real-time synchronization with project approval information from the research management system.

Hierarchical approval workflows: Constructing a three-tier structure of "departmental preliminary review - functional department re-review - final university leadership approval," with different budget types (e.g., basic expenditures, special construction projects) triggering differentiated approval paths. For instance, equipment procurement exceeding 500,000 RMB requires an additional countersignature node from the State-Owned Assets Office.

Dynamic adjustment mechanisms: Supporting mid-year budget additions, reductions, and reallocations between categories, requiring the system to record adjustment trajectories and automatically trigger recalculations of related indicators. For example, when a college reallocates 30% of its "international conference fees" balance to "equipment maintenance fees," the performance evaluation indicator library must be synchronously updated.

Technical Requirements

High scalability: Ensuring that new component integration does not affect existing operations. For example, adding a "research funding lump sum management" module only requires extending the budget allocation algorithm component without modifying core approval processes.

Module reusability: Developing a universal component asset library. For instance, the approval workflow engine component has been reused by Shanghai Jiao Tong University, Huazhong University of Science and Technology, and three other institutions. The report generation component supports adaptive formatting for both standard Ministry of Finance final accounts reports and local customized reports.

System integration capabilities: Enabling two-way data interaction with student management systems (student population forecasting), asset systems (equipment depreciation calculations), and non-tax financial platforms through an API gateway, with average interface response times controlled within 500ms.

Design Objectives

Loosely coupled component architecture: Defining component lifecycles based on OSGi specifications and employing dependency injection (DI) for inter-component communication, ensuring that component failure isolation radii do not exceed two associated components.

Balancing process standardization and personalization: Establishing a BPMN 2.0 standard process template library while allowing configuration tools (e.g., Drools rule engine) to adjust node thresholds. For example, setting "50,000 RMB" as the threshold for college-level autonomous approval versus submission to university-level approval, with colleges able to configure specific thresholds within the [30,000, 80,000] RMB range based on disciplinary characteristics.

This design clarifies interface specifications through component contracts and employs consumer-driven contract testing (CDC) to ensure component evolution compatibility, reserving technical interfaces for future cloud-native architecture evolution.

3. Component-Based System Design

System architecture design is key to ensuring system efficiency, scalability, and maintainability. The following details the overall system architecture design, combining layered and microservices architectures with containerized deployment strategies to support complex university budget management business needs.

Overall Architecture Design - The system adopts a layered architecture of "presentation layer - business

logic layer - data layer," combined with a microservices architecture and containerized deployment strategy:

Presentation layer: Built on Vue.js for responsive front-end interfaces, including budget submission, approval dashboards, and analysis panels, with load balancing via Nginx.

Business logic layer: Divided into six major categories of microservices components (see 3.2), constructed using the Spring Cloud framework, with service discovery via Eureka and circuit breaking/degradation through Hystrix.

Data layer: Utilizes MySQL for structured data storage with sharding and partitioning, Redis for caching frequently accessed data, and Elasticsearch for full-text search and complex analytical queries.

Containerized deployment: Encapsulating components using Docker and enabling automatic scaling through Kubernetes. For example, during peak budget preparation periods at the end of semesters, the number of approval service instances is automatically increased to 15.

Core Component Division

Component Name	Function Description	Interaction Method
Budget Submission Component	Provides project submission templates, data validation rule libraries, multi-role collaborative submission	RESTful API + WebSocket real-time validation
Approval Workflow Engine Component	Parses BPMN process definition files, drives approval node transitions, supports countersignatures, additional signatures, and rollbacks	Asynchronous message queues (Kafka)
Execution Monitoring Component	Real-time collection of budget expenditure data, comparison with planned progress, triggering threshold warnings (e.g., <30% execution rate highlighted in red)	GRPC streaming transmission
Data Analysis and Visualization Component	Provides OLAP analysis, trend forecasting, cockpit displays, supports drill-down queries	GraphQL flexible querying

Key Technical Implementations

Service interface standardization: Defining interfaces using OpenAPI 3.0 specifications, generating Swagger documentation, and implementing contract testing via WireMock.

Dynamic configuration management: Based on the Apollo configuration center, enabling dynamic updates of approval rules and budget templates. For example, N University can customize allocation rules for "disciplinary team construction fees."

RBAC permission model: Constructing a "role-permission-resource" triplet model for fine-grained control. For example, distinguishing data access scopes between "college budget officers" and "finance office auditors," with JWT tokens enabling permission propagation.

Database Design - Key table structure design

Budget Project Table: project_id (primary key), project_name, department_id, budget_amount, status, create_time

Execution Record Table: record_id (primary key), project_id, expense_date, amount, voucher_code, audit_status

User Permission Table: user_id (foreign key), role_id (foreign key), data_scope (JSON format storing accessible college codes)

The system implements ORM mapping using MyBatis-Plus and adopts ShardingJDBC for sharded storage of budget execution data, ensuring stable query response times for tens of millions of records within 200ms.

4. System Implementation and Testing

In the process of building an efficient and scalable university budget management system, technology selection and development environment configuration are crucial. The following elaborates on the system's development environment and technology stack, including backend and frontend technology selections and infrastructure configurations, laying a solid foundation for subsequent system development and testing.

Development Environment and Technology Stack

Backend technology stack

SpringBoot: As the microservices base framework, integrated with the Spring Cloud ecosystem (Eureka, Hystrix, Feign) to support rapid RESTful API construction.

Database layer: MyBatis-Plus simplifies ORM operations, and ShardingSphere enables sharding and partitioning of budget execution data.

Containerization: Docker is used for image construction (individual component image sizes controlled within 500MB), with Kubernetes implementing service orchestration and self-healing.

Frontend technology stack

Vue.js: Builds responsive interfaces based on Element UI, using Axios for API request encapsulation.

Dynamic form engine: Implements metadata-driven form rendering based on Avue.js, supporting JSON configuration-based development.

Infrastructure - Alibaba Cloud ACK container service is selected, configured with 3 d2.4xlarge nodes as the computing resource pool. Jenkins is used to implement CI/CD pipelines, automating the entire process from code submission to production deployment.

Core Function Implementation Examples

Dynamic form engine implementation

Metadata design: Defines field types (e.g., number input boxes, dropdown selection boxes), validation rules (e.g., "funding amount > 0"), and layout configurations via JSON Schema.

Rendering engine: Parses metadata based on Avue.js to dynamically generate Vue component trees, supporting formula calculations (e.g., "total budget = personnel fees + equipment fees" automatically summed).

Rule engine: Integrates Drools to execute complex business rules. For example, "automatically adds 'indirect costs' field when budget type = research project."

Multi-level approval process componentization

Process definition: Describes approval nodes and gateway conditions (e.g., "amount > 1 million RMB → university-level approval") using BPMN XML, stored in GitLab version control.

Component encapsulation: Encapsulates process parsing, node routing, and state persistence into independent JAR packages, providing standardized APIs such as startProcess(bpmnId) and approve(taskId).

Dynamic loading: Implements hot updates of process definitions through Spring Cloud Config. After a college modifies approval thresholds, changes take effect within 5 seconds across all relevant process instances.

Test Plan Design

Unit testing

Uses JUnit5 + Mockito to perform isolated testing of the Service layer, covering boundary conditions (e.g., budget overrun scenarios). Generates coverage reports via Jacoco, ensuring core components (e.g., approval engine) achieve >85% coverage.

Integration testing

Builds API test suites based on Postman to verify cross-component transaction consistency (e.g., automatic triggering of approval processes after budget submission). Uses TestNG to implement multi-service parallel testing, simulating 200 concurrent users performing mixed operations.

Performance stress testing

Uses JMeter to simulate 500 concurrent users performing budget submission operations for 30 minutes. Monitors metrics including average response time (target <1.5s), error rate (target <0.5%), and TPS (target >120).

Test Result Analysis

Functional coverage

Executed 472 test cases, covering all 38 functional modules, with a defect fix rate of 100%. The dynamic form engine supports 12 field types and 23 validation rules, with the approval process component successfully parsing 37 BPMN elements.

Performance metrics

Response time: Under 500 concurrent users, the average response time for budget submissions was 1.23s (P95=1.82s).

Throughput: The system achieved a peak TPS of 145, with resource utilization (CPU/Memory) stable below 65%.

Stability: No service interruptions occurred during 7×24-hour stress testing, with Kubernetes auto-scaling response delays <30s.

Test results demonstrate that the component-based architecture enables horizontal scalability. After adding two service nodes, response times decreased by 28%, fully verifying the achievement of design objectives.

5. Application Effects and Comparative Analysis

In verifying the practical application effects and advantages of the university budget management system, application cases from pilot universities provide strong evidence. The following takes N University as an example to showcase post-launch data handling, process optimization results, and highlights the system's design and technical advantages through comparison with traditional systems.

Application Case in Pilot Universities - Taking H University as an example, with an annual budget of 820 million RMB involving 23 colleges and 8 functional departments. After system launch:

Data handling capacity: Stored 120,000 historical budget project records, with an average of 2,500 new execution records added daily, and system response times stable within 800ms.

Process optimization comparison:

Metric	Traditional System	Component-Based System	Optimization Rate
Approval cycle	14-21 days	5-8 days	↓45%-62%
Cross-departmental collaboration frequency	4.3 times/project	1.8 times/project	↓58%
Data error rate	3.2%	0.9%	↓72%

Comparative Advantages Over Traditional Systems

Development efficiency improvement

Through component reuse (e.g., approval engine, form engine reuse rate >75%), new project demand development cycles were shortened from 45 person-days to 12 person-days. After containerized deployment, environment setup time was reduced from 2.5 hours to 15 minutes.

Maintenance cost reduction

The microservices architecture supports independent component upgrades, reducing the impact scope of single functional iterations by 82%. Dynamic configuration management eliminates code modifications for rule changes, reducing annual operational costs by approximately 40% (based on 300,000 RMB/year IT labor costs).

User Satisfaction Survey - A questionnaire survey was conducted among three user groups: functional departments, finance offices, and colleges (sample size n=127), with results as follows:

Functional departments (n=41)

System usability score: 4.6/5 (traditional system 3.2/5)

Function satisfaction: 92% responded "fully satisfied" or "basically satisfied"

Finance offices (n=35)

Audit efficiency improvement recognition: 4.7/5

Data visualization function usage: 3.8 times/person daily

College users (n=51)

Process transparency satisfaction: 4.5/5

Mobile compatibility evaluation: 4.3/5

Comparative analysis conclusion - The component-based architecture provides "high cohesion, low coupling" characteristics, demonstrating stronger flexibility and scalability in addressing complex university budget management needs. Compared to traditional monolithic architectures, business response speeds improved by over 60%, and operational complexity decreased by approximately 55%, validating the practical value of technical selections.

6. Conclusion and Outlook

This study explores innovative paths for university budget management systems, successfully constructing a highly flexible and scalable system architecture through the introduction of component-based technology. The following elaborates on breakthroughs achieved in technical practice, management efficiency, and economic value; further summarizes innovative points to provide references for research and practice in related fields.

Summary of Research Results - By introducing component-based technology, this study constructs a decoupled architecture for university budget management systems, achieving the following breakthroughs:

- Technical practice: Completing standardized encapsulation of core components such as dynamic form engines and multi-level approval processes, enabling "hot-swappable" functional expansion.
- Management efficiency: In pilot university applications, budget approval cycles were shortened by 45%-62%, and cross-departmental collaboration efficiency improved by 58%.
- Economic value: Through component reuse and containerized deployment, system construction costs were reduced by approximately 35%, and operational efficiency improved by over 60%.

Summary of Innovative Points

- Domain-driven component partitioning: Proposing a three-tier partitioning model of "business domain - subdomain - functional unit," subdividing budget management into eight subdomains such as fund allocation, execution monitoring, and performance analysis. Each subdomain corresponds to independent component packages, resolving the modification diffusion issue caused by tight coupling in traditional systems.
- Configurable workflow engine: Designing a BPMN 2.0-based graphical configuration interface supporting drag-and-drop arrangement of approval nodes and conditional gateways. Combined with rule engines, a "configuration-as-business" model is realized, shortening college process adjustments from two weeks to two hours.

Future Improvement Directions

- Intelligent budget forecasting: Integrating machine learning models (e.g., LSTM neural networks) to train on historical budget data, research project outputs, and other multivariate features, enabling intelligent pre-filling of departmental budget needs and anomaly warnings. Expected to reduce budget preparation time by 40%.
- Cross-platform mobile adaptation: Reconstructing mobile components using the Flutter framework to achieve over 90% code reuse across iOS/Android/Web platforms. Focusing on optimizing offline processing capabilities for approval operations, ensuring process node submissions can be completed in network-free environments.

Outlook - With the acceleration of educational digital transformation, the "Lego-style" architecture advantages of component-based budget management systems will become increasingly prominent. In the future, deep integration with financial middle platforms and research management systems can be explored to build a unified platform covering the entire university funding chain, providing data support for strategic decision-making.

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

7. References

- Chen, J. (2025). Optimization and innovation of financial budget management. *China Electronic Business*, (04), 96-98.
- Guo, X. (2025). Optimization research on financial budget management based on RPA. *China Collective Economy*, (07), 165-168.
- Liu, S. (2025). Application of management accounting in college financial management under the background of high-quality development. *China Township and Village Enterprises Accounting*, (03), 192-194.

Shao, Y. (2025). Construction and implementation effect analysis of performance-oriented financial budget system. *Township Enterprise Herald*, (04), 120-122.

Yao, W. (2025). Research on the construction of financial management and internal control system in colleges and universities from the perspective of new economy. *China Convention and Exhibition*, (05), 94-96.

