Hindawi
Discrete Dynamics in Nature and Society
Volume 2020, Article ID 1057902, 11 pages
https://doi.org/10.1155/2020/1057902

Research Article

Granularity Decision of Microservice Splitting in View of Maintainability and Its Innovation Effect in Government Data Sharing

Yan Li,[1] **Chun-Zi Wang**,[1] **Ying-chao Li**,[1,2] **and Jia Su**[3]

[1]*School of Management, Xi'an Polytechnic University, Xi'an 710048, China*
[2]*Emaplink Software Co., Ltd., Xi'an 710000, China*
[3]*School of Management, Xi'an University of Architecture & Technology, Xi'an 710055, China*

Correspondence should be addressed to Yan Li; sayidli@xpu.edu.cn

Received 18 May 2020; Revised 23 June 2020; Accepted 16 July 2020; Published 4 August 2020

Guest Editor: Chi-Hua Chen

The reasonable sharing and effective integration of data is the premise of improving the comprehensive governance ability of society through data empowerment. It can be determined that the fine-grained splitting of microservice framework can promote the reasonable sharing and effective integration of data. However, how to determine the granularity of microservice splitting is a multiparameter and multiobjective decision-making problem, which is also a key basic problem to be solved urgently both in academic research and application. From the perspective of application, this paper puts forward the criteria and basic framework that can guide the microservice splitting, and for the first time, based on the perspective of maintainability, it gives the decision criteria and methods of microservice splitting granularity. After the theoretical research, this paper also takes the provincial microservice governance of food and drug regulation as an example: the example shows that the framework and methods proposed can effectively improve data sharing and system expandability. Through microservice governance, collaborative social governance featuring "one network to achieve management objectives, one network to realize comprehensive business processing, one network for comprehensive view of all information" can be achieved. It is an exemplary mode worth popularization.

1. Introduction

As digitization evolves, humans are entering a finely resolved society, also known as the granular society [1] in which everyone must go through the transformation from "a rational person" to "a granular person." Technical innovation provides high-density and more detailed insight into all phenomena, and the high resolving process of digitization will also drive social system disintegration. As the result of this "resolving-disintegration" dual processes, population average value and outdated knowledge of the coarse-particle time will be phased out and the consequent "big-data mindset" will revolutionize the traditional philosophy of social governance. Thus, amid the granular time, how to innovate governance mode and modernize relative ability of the government is a significant issue urging for a solution.

At present, the world is witnessing an upsurging trend of rapid transformation from data into information and knowledge and the improvement of data-driven government governance ability [2, 3]. As China's reform enters the deep water zone, potential social contradictions and conflicts are looming. Thus, data-based speaking, decision, management, and innovation have become an effective way to enhance our comprehensive and collaborative social governance ability. On August 19, 2015, the State Council executive meeting adopted the Action Outline for Promoting Big Data Development, which defined how to transform government functions and optimize service modes through data sharing and integration from the national top-layer policy design level. Implementation Plan for the Integration and Sharing of Government Information System (General Office of the State Council [2017] No. 39) requires the combination of investigation and elimination to speed up zombie

information system removal and the integration and sharing of the internal information system of each department. On this basis, Notice on the Implementation of Government Information System Integration and Application Pilots (Department of High Technology Industry of National Development and Reform Commission [2017] No. 1714) issued by the National Development and Reform Commission also mentioned a couple of times that the foundation of information system integration and sharing is the integration of government data and the establishment of a unified sharing platform. This pointed out where the transformation and upgrading of China's e-government service ecosystem should go. Data integrated by e-government accounts for over 90% of the total amount of social data [4]. As the granular mode increasingly intensifies, data sharing and integration become the basic requirements for the synergy and accuracy of comprehensive social governance and represent a consensus reached by the academic and the industrial communities. At present, government data sharing and integration are mainly implemented through enterprise service bus (ESB) [3, 5, 6] and data center sharing [4, 7]. As a key component of service-oriented architecture (SOA), ESB gathers middleware, XML (Extensible Markup Language), Web services, and other technologies to support intra- or interenterprise heterogeneous systems in service, messaging, or event interactions. With standard open protocols and a proven application integration model, it acts as a connection hub linking various application systems for data sharing. However, responsiveness and maintenance cost are where its development constraints lie. The exchange service configuration is complex and users have to shoulder much workload. Service updates can only be achieved by reconstructing the original system, cleaning data, and transferring them to the terminal. These defects show that ESB can hardly meet exponentially growing interaction demands between information systems. The concept of data center sharing is to build a large centralized data resource pool level by level and realize data sharing between different organs through the data resource service catalog. The advantage is that it can form a unified data source and ensure data consistency, but this traditional data collecting method will inevitably involve data collection, comparison, cleaning, and heterogeneous data conversion of different organs and organizations, which makes real-time data hard to guarantee. Considering the authority and responsibility of departments, information sharing faces resistance and obstacles in large areas. Government data integration is greatly limited in many ways including information degradation in the two-way funnel filtering process of the upper and the lower levels in departments, information island caused by the interdepartment chimney system, and information obstruction at the terminal end led by the information asymmetry between the government and the public [7, 8].

Data sharing is the precondition of realizing the mutual function of big data and government governance. Therefore, government data sharing is a fundamental and key issue for both academic research and practical application. This paper builds an underlying platform of data sharing and service connection based on the distributed microservice architecture; the overall chapter structure is as follows. The first section is the introduction; the second section summarizes the history of software technology architecture; and on this basis, the third section clarifies the definition and advantages of microservice. In the fourth section, the paper introduces the framework and key technologies of the distributed microservice platform and for the first time discusses quantitative measurement indexes of microservice architecture. In the fifth section, the availability of the architecture is verified by the example of the collaborative social governance system of Shaanxi Medical Products Administration. The sixth section is the summary of the whole paper.

The main innovations and contributions of this paper are as follows: (1) systematically summarizes the differences and advantages between microservice architecture and other three mainstream architectures and clarifies the main application scenarios and functions of microservice; (2) for the first time, based on the perspective of maintainability, gives the decision-making standards and methods of microservice resolution granularity; and (3) from the perspective of application effect, takes the provincial food and drug supervision microservice governance. An example is given to demonstrate the significance of the proposed criteria and methods.

2. The Characteristics and Splitting Granularity of Microservices

As a systematic sketch, software architecture (SA) is the foundation of software practices and the series of principles and goals set by software systems. Though the concept is simple, it is hard to draw an accurate definition, even a descriptive one, so most engineers comprehend it from the intuitive perspective. Based on years of experience in software industry, this paper divides the evolution process of SA into monolithic, vertical, service-oriented, and microservice architectures.

Monolithic architecture (Figure 1) has no unified definition. Essentially, it organizes all basic applications into one module or unit [8] or, in other words, gathers all functions into a project during the development process and packs them into a WAR for further server deployment. Both time- and cost-saving, it is simple in structure and suitable for small-scale projects. Little interface interaction makes it relatively easy to optimize and upgrade execution performance of the system. However, disadvantages also exist: (1) all functions in one project impede the mutual expansion and maintenance work of multiple collaborators; (2) generally, system upgrade must rely on the clustering on both application and database levels, which is costly; and (3) mass data reading and writing technologies are still facing bottlenecks.

Derived from the concept of the vertical structure in nature (special vertical differentiation or stratification of communities), vertical architecture targets at the limitation of monolithic application architecture. As project complexity and the amount of users grew, the acceleration driven by cluster optimization in single applications tapered; thus,

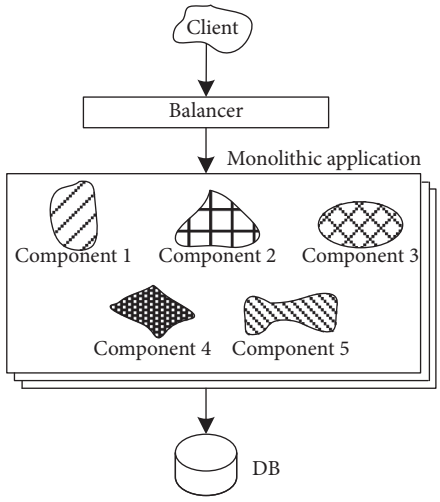

Figure 1: Framework of monolithic architecture.

people began to raise efficiency by vertically dividing large projects and splitting them into multiple monolithic architecture projects (Figure 2). Vertical architecture can, to some extent, alleviate the defects of traditional monolithic architecture. For example, vertical separation can resist the original monolithic project from excessively expanding, and different technologies can be used on different vertical projects, making collaboration and maintenance easy to operate by multiple people. But at root, it is still a continuation of monolithic architecture, so the essential problems have yet to be solved.

With the boom of vertical architecture applications, the interaction problem between different applications became salient. SOA gained popularity around 2000 (Figure 3). Accurately, it is an architecture style [9, 10]. In this service structure featuring coarseness and loose coupling, communication between services is achieved through simple and precisely defined interfaces, involving no underlying programmatic interfaces and communication models. Fundamental SOA consists of service provider, consumer, and registration. It abstracts repeatedly shared functions into modules and provides services for each system in the way of service. Through ESB, multiple services in the project communicate by means of Web service and RPC. The advantages of SOA are apparent: (1) the extraction of shared modules raises reusability and maintainability, thus promoting efficiency; (2) since ESB lowers the coupling degree between interfaces, different projects and services can adopt different technologies and clustering and the optimization plan can be tailored due to the characteristics of a certain service. Meanwhile, the vague boundary between system and service hinders SOA development and maintenance. Though with ESB, the unfixed and various service interface agreements still impede system maintenance. As a result, services extracted are too coarse, and the coupling degree of system and service is high.

As a new architecture concept put forward after 2010, microservice architecture has no unified definition at present [11–13]. The most representative one is proposed by Martin Fowler [12, 14] who thinks MSA is a specific designing method transforming software applications into independent and deployable service modules. It is structured in such a style that single applications are developed due to service abilities into microservices independently running in their own processes (Figure 4). Service communication is realized on the basis of lightweight protocols (such as RESTful), and each service can be independently deployed by fully automated mechanisms. This method, which splits out the service layer and extracts it into multiple small-scale services, has obvious advantages. Compared with traditional SOA, it performs split in a more refined way, which enhances resource recycling rate and efficiency. The decentralization principle and lightweight communication agreements adopted are more flexible than ESB, which makes possible more accurate service optimization plans after service simplification, improves system maintainability, and shortens product iteration cycle for the upgrading need of the Internet era. For some scholars [15], the cost of MSA service governance and service granularity promotion must keep balance, and distributed development means larger challenges for teams, requiring higher technical costs.

In practical service application, challenges come from both the service and the technique. This section draws a detailed summary of the features of the four architectures and analyzes their key distinctions (Table 1). In terms of hierarchy, monolithic and vertical architectures centralize functional modules of each hierarchy with high coupling degree; SOA uncouples multiple functional modules of vertical and horizontal hierarchies of three or more tiers, but public modules can only be shared on horizontal hierarchies, leading to unthorough uncoupling; the fully self-service flexibility achieved by simultaneous uncoupling on vertical and horizontal hierarchies represents the main characteristic of microservice architecture; however, when putting large projects into practice, development teams cannot comply with all the features and they must consider the integration of irreplaceable systems and promote the flexibility of full uncoupling within acceptable changing rate. Here, the rules of system microservice governance are summed up into two principles:

Principle 1. Regarding the existing service system as services which can participate in microservice architecture, namely, regarding the external systems of each monolithic architecture as independent services clustered by autonomous agents.

Principle 2. Based on the concept of microservices, using SOA architecture, which is controllable for the technical team, to proceed uncoupling upgrade. If the technical routes are closed, then adopt verified upgraded SOA architecture, if not, use the brand new microservice one to explore.

3. Microservice Framework and Key Technologies

Based on the principles of microservice governance, this section designs a whole set of microservice frameworks and common services and describes key functions in microservice development including configuration management, service discovery, circuit breaker, intelligent routing, micro

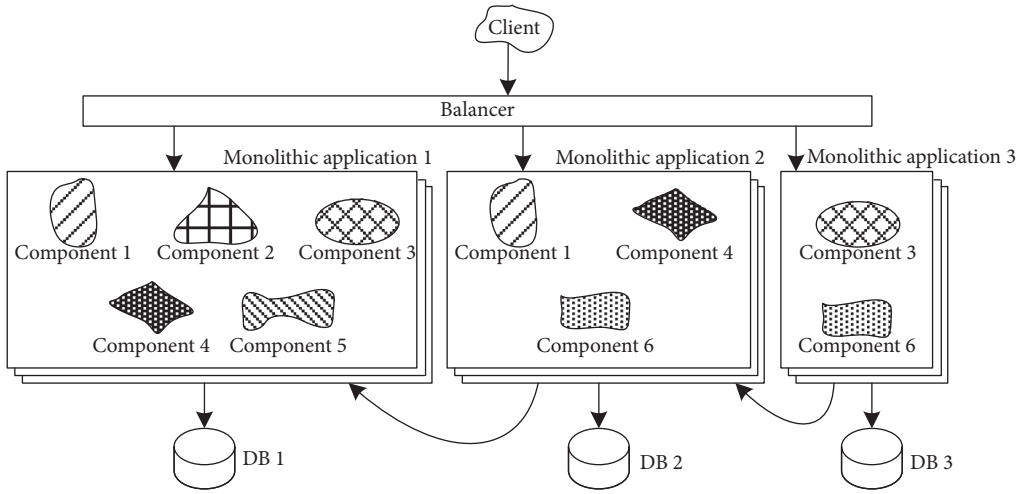

Figure 2: Framework of vertical architecture.

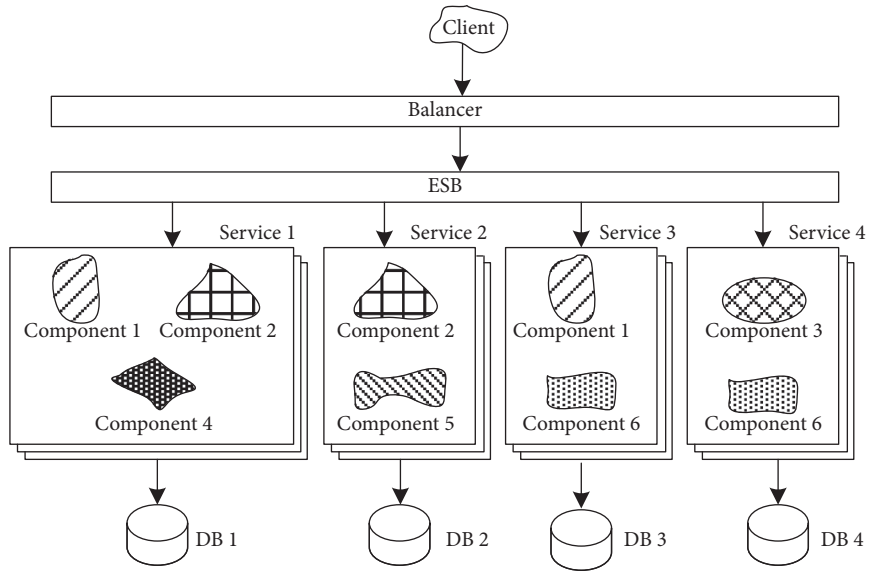

Figure 3: Framework of service-oriented architecture.

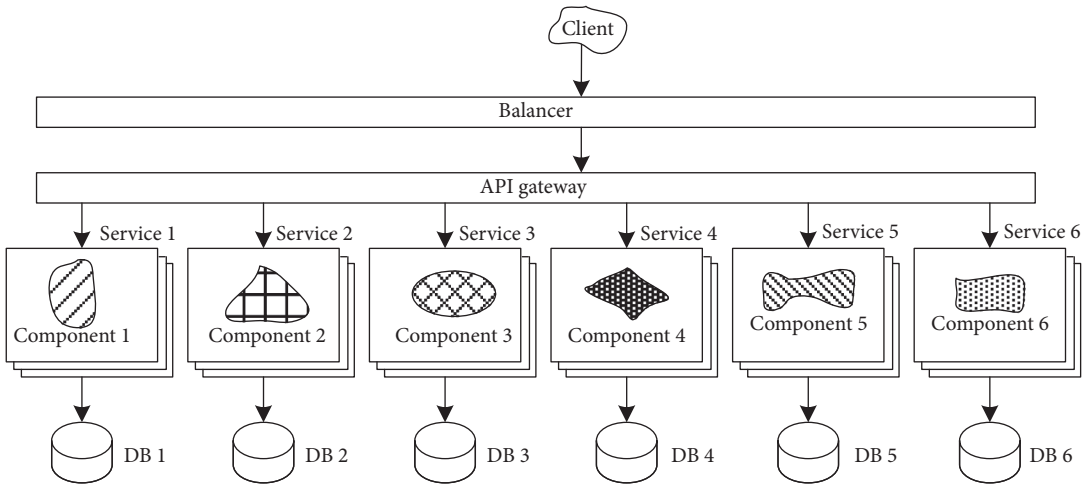

Figure 4: Framework of microservice architecture.

Table 1: Key differences between four architectures.

	Monolithic architecture	Monolithic–vertical architecture	SOA architecture	Microservices architecture
Time	Before 1990s	1990–2000	2000–2010	After 2010s
Feature	Tight coupling	Tight coupling	Light coupling	Decoupled
Advantage	(1) Simple architecture, short development cycle, and low cost (2) It is relatively easy to optimize and improve performance because of less interaction between interfaces	(1) Simple architecture, short development cycle, and low cost (2) Vertical splitting of original monomer projects should not be excessively enlarged (3) Different projects can adopt different technologies	(1) Extraction of common components improves reusability, maintainability, and efficiency (2) Different projects or services can adopt different technologies (3) ESB further reduces the coupling between system interfaces	(1) Finer granularity is more conducive to resource reuse and efficiency improvement (2) Decentralization and lightweight communication protocol (3) The system has strong maintainability and short iteration period
Disadvantage	(1) All functions are integrated in one project, which is not conducive to the coexpansion and maintenance of multiple collaborators (2) Performance optimization depends on clustering, which has bottlenecks in high cost (3) Technology stack constraints	(1) Simple decomposition of monolithic architecture (2) Not conducive to the development and maintenance of large-scale projects (3) Performance expansion has bottlenecks	(1) The boundary between system and service is blurred, which is not conducive to development and maintenance (2) The interface protocols of services are not fixed and there are many kinds of services (3) The extracted service granularity is too large and the coupling between system and service is high	(1) There should be a reasonable balance between the cost of service governance and the refinement of service granularity (2) Challenges to development team and high cost of technology
Scope of application	Small projects	Medium-scale project	Large project	Large projects with frequent and complex interactions

brokers, control bus, distributed sessions, and clustering state management. Also, it analyzes the relationship between the underlying technical principle, logical design, and modules so as to provide guidance and examples on how to rapidly promote microservice governance and realize service and data connection and sharing. Figure 5 shows the concise processing flow diagram of microservice. Key technologies consist of service registration and discovery, remote service calling, circuit-breaker mechanism, service link tracking, and annotation interfaces.

3.1. Service Registration and Discovery. The naming service exists as a basic service in microservice architecture, with name servers as the central node. Each service system defines a service name which is taken as the identifier by the system and registered at the name server. The server identifies each service system by its service name, which provides routing relays for mutual calls between multiple systems. The detailed process is as follows:

Step 1. When the service producer starts, it registers the service it provides at the service registration center

Step 2. When the service consumer starts, it subscribes to the service it needs at the service registration center

Step 3. The registration center returns the address information of the service provider to the consumer

Step 4. The consumer calls the service from the provider

3.2. Remote Service Calling. Remote service calling clarifies the calling protocol for intersystem services, which makes services calling external systems the same as proxy local services. Remote service calling uses the Hypertext Transfer Protocol (HTTP) POST protocol to establish an outgoing channel for data service call. As the HTTP protocol is stateless, JWS authentication method is adopted in system design to ensure secure access to internal resource services and prevent illegal access. Each remote calling must use the token, and the server will verify the token to assess the validity. The basic principles are as follows:

Step 1. The service caller performs remote calling requests according to the service name.

Step 2. The service center picks up the service provider from the service list due to the requested service name.

Step 3. The HTTP connection is established between the service caller and the finder, and data is sent through POST. After receiving the HTTP request, the service provider will first perform validity check and processes the corresponding service according to the request data if the check is passed. When the processing is completed, the result data is returned to the service caller.

3.3. Circuit-Breaker Mechanism. The function of circuit-breaker mechanism is to avoid system failure from spreading. When a system or a function cannot provide

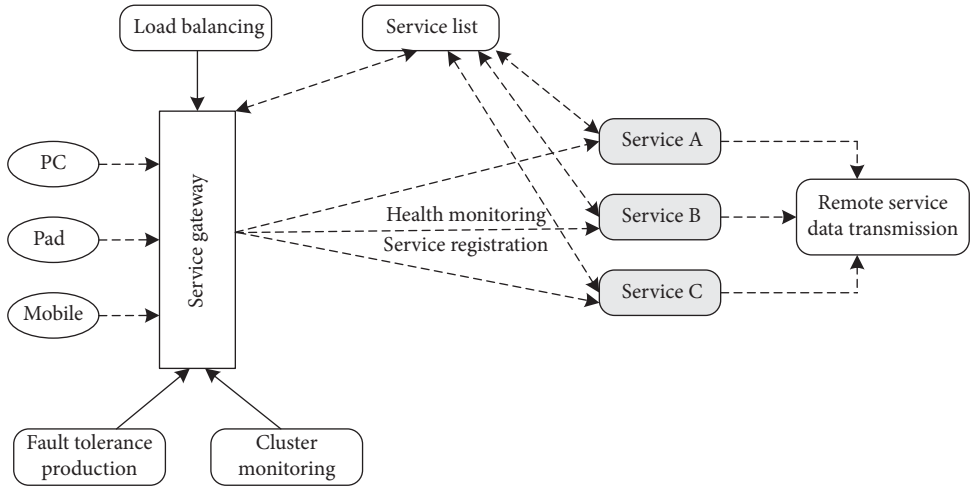

Figure 5: Concise request processing diagram of microservice architecture.

services due to failure, the system should be automatically taken offline from the entire distributed platform. In this way, large-area system crash caused by all requests waiting for a response at that function point can be avoided. The circuit-breaker mechanism is as follows:

Step 1. When a certain threshold is met (the default value is over 20 requests in 10 seconds).

Step 2. When the failure rate reaches a certain level (the default value is over 50% request failure in 10 seconds).

Step 3. When the above threshold is reached, the circuit breaker will turn on.

Step 4. When it turns on, all requests will not be forwarded.

Step 5. After a while (taking 5 seconds as the default value), the breaker is half-open at this moment and will let one of the requests be forwarded. If this succeeds, the circuit breaker will turn off; if not, it will maintain opened and Step 4 will be repeated.

3.4. Service Link Tracking. Service link tracking provides full-link tracking and monitoring during the completion of the entire function, which can clearly show the relationship between each service calling and accurately pinpoint when problems occur. When the system is running, the service link tracking module receives the real-time monitoring data of each microservice system. It mainly includes 4 components:

Component 1: collector, which receives or collects data transmitted by each application.

Component 2: storage, which stores data received or collected in internal storage by default. It currently supports Memory, MySQL, Cassandra, etc.

Component 3: API (query), which is responsible for querying data stored in storage and providing simple data obtained by JSON API, mainly for web UI.

Component 4: Web, which provides simple Web interfaces.

3.5. Annotation Interfaces

Service registration interface @EnableEurekaClient: as an annotation, this interface will go through scanning when the system starts. If the annotation storage is scanned, the configuration information of the naming server is automatically obtained from the configuration file to register the service system.

Remote service call interface @FeignClient: this interface appears as an annotation and the annotation is added when a remote service call is required. After the remote service name is input, the interface obtains the address of the remote service from the naming server and establishes the http connection.

Circuit-breaker interface @EnableHystrix: this interface starts the circuit breaking service, and the corresponding mechanism will start automatically.

Circuit-breaker detection interface @EnableHystrixDashboard: this interface launches the circuit breaking monitoring UI page and visualizes all relevant data on the interface.

Service link tracking interface @EnableZipkinServer: the service link interface starts the health monitoring service, collects trace information from the http protocol from collector. The client terminal calls/api/v1/spans or/api/v2/spans to report the trace information.

4. Microservice Splitting Granularity Decision and Calculation Formula

The core role of microservice architecture is to cope with the growing service capability within the system and the increasingly complex interaction demands between systems. After microservice governance is performed in accordance with the principles and frameworks in the first two sections

of this paper, papers evaluating the effectiveness are quite rare. Obviously, according to the fine-grained service decoupling method in this paper, service updates will be more frequent and the iteration speed will be raised. To ensure that software plays a greater role in overall value creation, the maintainability measurement of microservice system is the most important index.

Definition 1 (maintainability). Referring to the description of Rowe et al. [16], maintainability of microservice system can be defined as how the system can maintain effective and efficient when going through correction, refinement, expansion, or optimization.

Based on Definition 1, the operation of maintainability can be separated into two parts. One is performing correction and refinement when demands are stable; the other is performing expansion and optimization when demands change. Though lots of papers [17] have provided definitions on the maintainability of object-oriented system, there has been no unified conclusion when it comes to microservice system. Thus, on the basis of a large amount of literature research [18–20] and practical experience, this paper discusses from three dimensions related to maintainability, namely, size, coupling degree, and cohesion.

For convenience, we suppose microservice system Y is composed by n services, namely, $Y = \{S_1, S_2, \ldots, S_n\}$, $|Y| = n$. Service S_i has m interaction interfaces, namely, $S_i = \{\mathrm{SI}_1, \mathrm{SI}_2, \ldots, \mathrm{SI}_m\}$, $|S_i| = m$.

Definition 2 (size). The size of a microservice system is the sum size of all its subservices. Obviously, under the same condition, the larger the microservice system is, the lower its maintainability is. The traditional definition of size for a service or operation is related to Lines of Code (LOC) [17]. However, in this paper, the size of the service is represented by the weighted value of the operation amount contained in the exposed interface of a service S, namely, $S_{\mathrm{ws}} = \sum_{\mathrm{SI} \in S} \sum_{O \in \mathrm{SI}} w_l / mk$, in which $W_l (l = 1, 2, \ldots, k)$ is the weighted value of operations within a certain interface (the weighted value can be set according to the amount of parameters and the coarseness of the interface), and the size of microservice system Y is $Y_{\mathrm{ws}} = \sum_{S \in Y} S_{\mathrm{ws}} / |Y|$.

Definition 3 (coupling degree). Coupling degree reflects how much one service depends on others. The lower it is, the higher the maintainability of microservice system is. If service S_i relies on service S_j and vice versa, then this is called interdependence, which is what we have to avoid (and combine the two services into one) in practice. According to [19], S_{IOS} is the importance of a service, showing the amount of consumers depending on service S (the amount of clients of interface SI_i of the called service S). S_{DOS}, the dependence of a service, shows the amount of services depended on by service S (the amount of services of which more than one interface is called by service S). The criticality of a service $S_{\mathrm{COS}} = S_{\mathrm{IOS}} * S_{\mathrm{DOS}}$. Though lower coupling degree means higher maintainability, in the transformation of microservice system services, low coupling degree will constitute a bottleneck. That is because service S will always be called or calls other services and the coupling degree can, to a large extent, help system designers find unreasonable calling.

Definition 4 (cohesion). Cohesion refers to the contribution of the service operation to a certain task or function. There is a positive correlation between it and system maintainability. The connotation of cohesion is quite complex semantically, so it is hard to measure automatically. According to [19], the interface data cohesion (IDC) of service S is defined as the similarity of the parameter type of S's internal interface SI_m, namely, $S_{\mathrm{IDC}} = P_{\mathrm{type}}/m$, in which P_{type} is the data type of interface parameter. When $S_{\mathrm{IDC}} = 1$, the system cohesion is high. Interface usage cohesion (IUC) represents the ratio of the internal interface amount of service S called by the customer and the total amount of interface data of service S, namely, $S_{\mathrm{IUC}} = |\mathrm{SI}_{\mathrm{invok}}|/m$. Similarly, when $S_{\mathrm{IUC}} = 1$, the system cohesion is high. S_{TIC}, the total interface cohesion (TIC) of service S, equals $(S_{\mathrm{IDC}} + S_{\mathrm{IUC}})/2$.

5. Practical Microservice Governance

Shaanxi Medical Products Administration (hereinafter referred to as the Administration) has been using monolithic architecture and upgraded vertical architecture for information construction and connecting with systems of other provincial departments through Web service. By the end of 2016, it has formed a comprehensive service platform covering all aspects of food and drug regulatory system and generalized it to the whole province. However, with the growing functions of the system, the accumulating service data and the provincial-wide user group, the performance of the system has been under great pressure and gradually could not satisfy actual application needs any more. This was manifested in (1) slow response, which makes it failed to return data within a reasonable time and (2) insufficient concurrency support, which means when services are centralized in a certain period, abnormal conditions will happen such as service denial. Besides, as the system became more complex, the cost and risk of new function development, testing, and launch greatly increased. According to the overall requirements of the "Internet + government services" and the 13th Five-Year Plan on Food and Drug Safety of Shaanxi Province and the actual problems existing in the system, experts of the Administration discussed and decided to rearchitect the original system with the new architecture of distributed microservice governance. In this way, the service system can achieve infinite horizontal scaling and gain support in adapting to upgraded demands. The platform also provides the unified data interaction service that truly integrates small systems into a large one.

5.1. Goal of Microservice Governance. According to the requirements of China Food and Drug Administration and the informatization of provincial food and drug administrations, the service framework of the food and drug regulatory system is shown in Figure 6. The system is

mainly composed of three components: the comprehensive service platform, the enterprise service platform, and the collaborative social governance platform. Among them, the first platform is accessed through the unified identity authentication system and serves as the main working platform for supervisors at all levels. After the enterprise service platform is registered and logged in by enterprise users, it can conduct service declaration, process progress inquiry, and receive and give feedback on regulatory information. The third platform requires no registration and login, and one can check or query through the Administration's website or WeChat. It integrates with the newly revised website of the Administration, making data query and utilization more convenient.

Based on the microservice concise processing framework of Figure 5 in Section 3, microservice upgrade and transformation are performed on the original Shaanxi provincial regulatory information system of small-scale food workshops, catering units, and stall keepers. The key tasks and goals include database table management, service splitting of monolithic architecture, workflow customization, and establishment of microservice supporting platforms.

(1) Database table management is to reasonably split some basic information tables that affect performance, making each table as responsive as possible. In terms of the repetitive, unreasonable fields left over from the past, uniform standards are defined. This part mainly contains splitting the enterprise principal information basic table (subtables and the main table are associated through Universally Unique Identifier (UUID) by its uniquely identified subject), reencoding organizational structure (managing the organizational hierarchy by the custom encoding format, and listing administrative region codes as a common attribute in order to flexibly adjust the organization level), splitting the service library (isolate the service library table and sync and share common basic data. Data that belongs to a specific module is regularized, cut from the intersect with other service system data), and replacing triggers with active service calling (use active remote service calling to replace triggers in data upgrade and sync, as the large use of the latter may cause database deadlock)

(2) According to the actual service classification of the Administration, services of the original monolithic system are split into multiple subsystems that can function and be deployed independently, including management of basic user authority, regulatory files, administrative licensing, daily inspection, inspection and handling, random inspection, public notice, and external network report. The relationship of different service systems is teased out. Each service system provides the services they need to release externally so that other systems can call those services remotely. To facilitate management, service systems are not allowed to implement the functions of others. A unified standard for external services is adopted and JSON Web Token (JWT) authentication is used for service calling

(3) In order to improve the flexibility and adaptability of the administrative licensing process, a workflow engine is utilized to realize licensing process customization. More specifically, by defining BussinessKey, enterprise users are linked with process definition which is then connected with the license type through keywords of the predefined license type. The process node is associated with the actual system processing function through predefined keywords

(4) The establishment of microservice supporting system is the core part of microservice transformation. From the perspective of service transformation, each service system provides one or more specific services. In traditional architectures, these systems are totally independent and can hardly utilize specific services of each other. That is to say, many functions are repeatedly established. Through open-source framework, the system of the Administration can achieve basis platform functions such as naming, remote service calling and load balancing, circuit-breaker mechanism, and service self-recovery and service link tracking (Figure 7).

5.2. Results of Microservice Splitting Granularity Decision. According to the microservice framework in Section 3 and the microservice governance goals of the Administration in Section 5.1, service analysis and service-oriented governance have been performed on the original regulatory information system. The system was divided into 12 microservices including administrative licensing, daily supervision, double random inspection, inspection and enforcement, random inspection, risk and credit rating, regulatory archives, data analysis, system maintenance, system monitoring, and information submission. Calculation results of the maintainability indicators of each microservice are shown in Table 2, which shows that microservice transformation greatly raises the scalability and maintainability of the system.

The system of Shaanxi food and drug supervision completed the start-up construction of government procurement procedures in July 2017 and passed the expert acceptance on July 4, 2018. As of June 30, 2018, the system has covered 14 county-level bureaus directly subordinate to the municipal and provincial governments, 104 district and county bureaus, more than 1,400 regulatory agencies, 10,715 supervisors, and 284,544 enterprise users, namely, nearly all food and drug supervisors and food and drug regulatory targets of the province. The supervisory terminal is logged in by 300,000 people per month and the enterprise terminal by 140,000. In total, the system produces 13.64G structural data and 890G unstructured data.

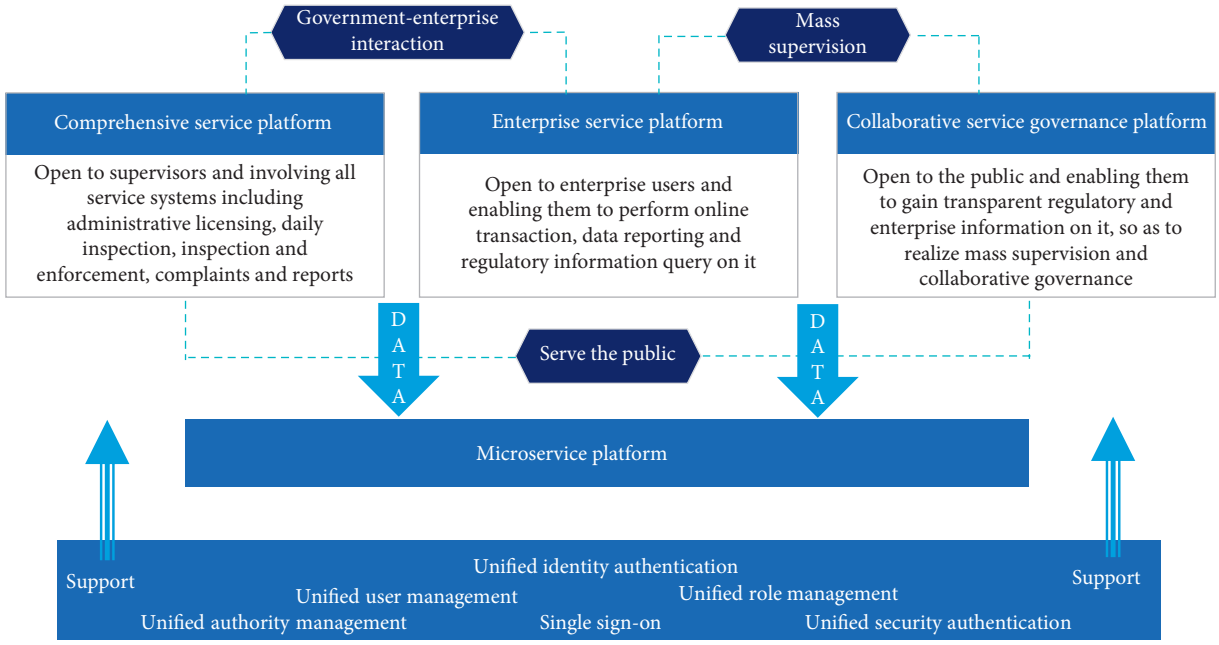

Figure 6: Service framework of the provincial food and drug regulatory system.

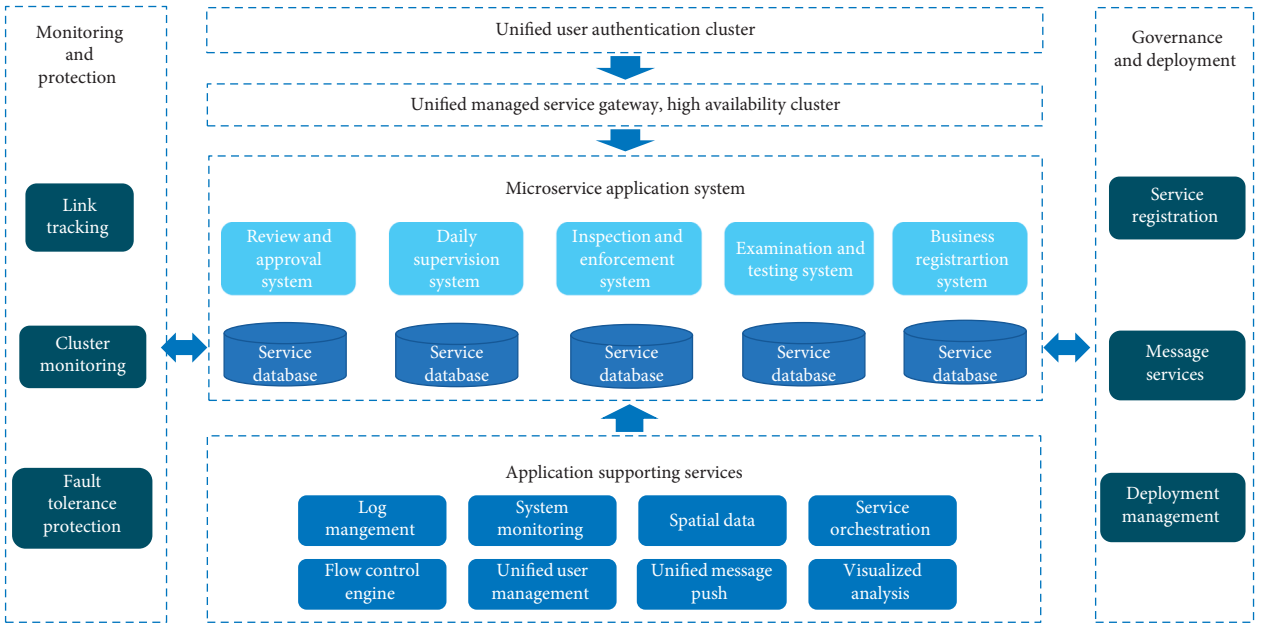

Figure 7: Framework of the microservice-supporting platform.

The comprehensive service system of the Administration implemented the value effect of "three portals, four terminals" (Figure 8). The three portals mean the one-network-based governance for government regulation, the one-network-based transaction for enterprise businesses, and the one-network-based communication for public service, which has achieved the "unified deployment in the province scale and hierarchical application on the prefectural and municipal level" through microservice transformation. The four terminals mean that due to the microservice terminal adaptation function, the system can be accessed through multiple channels including PC, Android, and IOS mobile terminals and WeChat official accounts. The microservice transformation result of Shaanxi province has won a large round of applause from acceptance review experts and insiders and appeared in the special reports in the "Annual Review during the Two Sessions" program of CCTV and Office of the Central Cyberspace Affairs Commission. As one of the "2018 National Intelligent Regulation Exemplary Cases," it has been generalized and replicated in 11 provinces and cities.

Table 2: Calculation results of the maintainability indicators.

No.	Service name	Interface number	Operation number	S_{ws}	S_{IOS}	S_{DOS}	S_{COS}	S_{IDC}	S_{IUC}	S_{TIC}
S_1	Administrative licensing (API)	5	8	0.35	8	3	24	0.9	0.60	0.75
S_2	Daily supervision (XML)	3	5	0.54	5	5	25	0.9	1.67	1.28
S_3	Double random check (API)	3	3	0.67	3	5	15	0.8	1.67	1.23
S_4	Enforcement of inspection (WSDL)	4	5	0.45	2	5	10	1	1.25	1.13
S_5	Sampling inspection (JSON)	2	5	0.70	2	3	6	0.7	1.50	1.10
S_6	Risk rating (API)	3	3	0.67	2	3	6	1	1.00	1.00
S_7	Credit rating (API)	3	3	0.67	2	3	6	1	1.00	1.00
S_8	Regulatory archives (API)	5	2	0.70	10	2	20	1	0.40	0.70
S_9	Data analysis (API)	2	10	0.60	2	10	20	0.3	5.00	2.65
S_{10}	System maintenance (XML)	3	5	0.53	2	10	20	1	3.33	2.17
S_{11}	System monitoring (XML)	2	10	0.60	1	10	10	1	5.00	3.00
S_{12}	Information submission (WSDL)	2	2	1.00	1	8	8	1	4.00	2.50

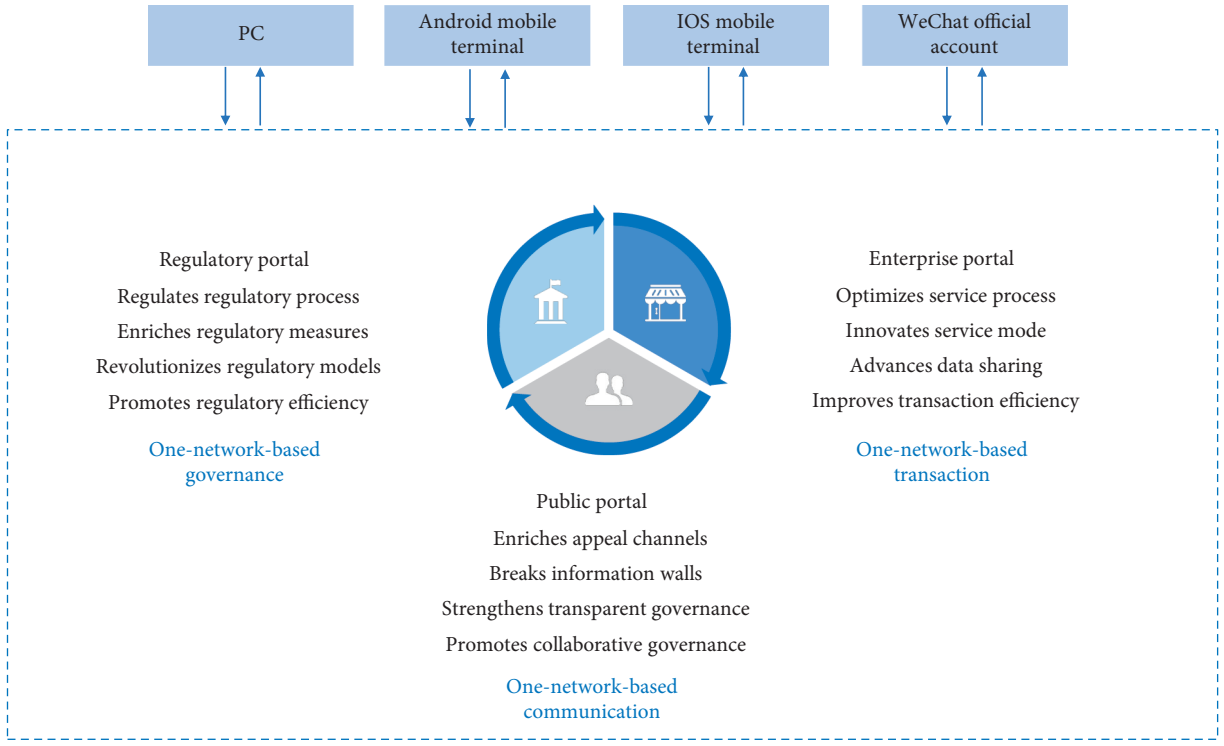

Figure 8: Value effect of "three portals, four terminals."

6. Conclusion and Prospect

The data-driven concept can innovate the comprehensive social governance mechanism effectively, and the premise of utilizing data-intensive research paradigm is the full sharing and integration of data. This paper first summarizes the critical meanings of government data sharing, introduces the four architectures of data integrated governance in a systematic way, and analyzes their differences, advantages, and disadvantages. Then, it provides the basic framework and the maintainability measure indexes of microservice from the perspective of practical application. In the end, it takes the microservice governance of a provincial food and drug regulatory system as an example to verify the framework and measure indexes proposed, which, as the practical result shows, promote data sharing and the expandability and maintainability of the system. Through microservice governance, collaborative social governance featuring "one-network-based governance, transaction, and communication" can be achieved. It is an exemplary mode worth popularization.

Microservice architecture is a practical technology derived from applications, most of which are based on individual experience or understanding for granularity segmentation of microservices. This paper firstly analyzed and discussed the split decision-making process of microservice from a rigorous theoretical analysis perspective, which has certain innovation significance in this field. However, the theoretical decision-making in this paper is only from a simple application and maintenance perspective, without considering some other factors. Therefore, more factors need to be integrated for rigorous theoretical analysis and cases in the subsequent research.

Data Availability

The data used to test the matching decision model of this study are included within the article. Additional data can be provided by the corresponding author upon request.

Conflicts of Interest

The authors declare no conflicts of interest.

Acknowledgments

This research was supported by the Shaanxi Provincial Department of Education serving the Special Project of Local Enterprises under Grant no. 2019TJ034; Research Project of Key Projects of Statistical Science in Shaanxi Province under Grant no. 19JC015; Xi'an Science and Technology Plan University Talent Service Enterprise Project under Grant no. GXYD7.7; and Xi'an Polytechnic University Doctoral Research Start-Up Fund under Grant no. 107020342.

References

[1] C. Kucklick, *The Granular Society*, CITIC Group Press, Beijing, China, 2017.

[2] C.-h. Wang, "Evolution and innovation of "comprehensive governance" in China," *Journal of Beijing Administrative College*, vol. 2, pp. 42–46, 2015.

[3] C.-H. Chen, "A cell probe-based method for vehicle speed estimation," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E103.A, no. 1, pp. 265–267, 2020.

[4] Z.-m. Zeng and Y. Qian-wen, "Research on the data curation system of urban public security oriented to the fourth paradigm," *Information Studies: Theory & Application*, vol. 2, pp. 82–87, 2018.

[5] Ye Tian, Bo Yuan, and Li Ting-li, "A massive and heterogeneous data storage and sharing strategy for Internet of things," *Acta Electronica Sinica*, vol. 44, no. 2, pp. 247–257, 2016.

[6] L. I. Xiao-tao, H. U. Xiao-hui, G. U. O. Xiao-li, and W.-n. LU, "Complicated information sharing technology based on metadata," *Systems Engineering and Electronics*, vol. 37, no. 3, pp. 700–706, 2015.

[7] L. Ming, "Large data technology and public security information sharing ability," *E-government.*vol. 6, pp. 10–19, 2014.

[8] B. Lake, "*An empirical evaluation of an agile modular software development approach: a case study with ericsson*," M.S. thesis, Stockholm University, Stockholm, Sweden, 2012.

[9] R. Arnon, *SOA Patterns*, Manning Shelter Island, Shelter Island, NY, USA, 2012.

[10] D. Sprot and L. Wilkes, *Understanding Service Oriented Architecture*, Microsoft Developer Network, 2004, https://msdn.microsoft.com/en-us/library/aa480021.aspx#aj1soa_topic5.

[11] J. Thönes, Micro-services, IEEE Software, ISSN 0740-7459/15, 2015.

[12] C.-H. Chen, F. Song, F.-J. Hwang, and L. Wu, "A probability density function generator based on neural networks," *Physica A: Statistical Mechanics and Its Applications*, vol. 541, Article ID 123344, 2020.

[13] G. Gruman and A. Morrison, "Micro-services: the resurgence of SOA principles and an alternative to the monolith," *Technology Forecast: Rethinking Integration*, vol. 1, 2014.

[14] M. Fowler, "Micro-services," 2014, http://martinfowler.com/articles/micro-services.html.

[15] M. Vianden, H. Lichter, and A. Steffens, "Experience on a micro-service-based reference architecture for measurement systems," in *Proceedings of the 21st Asia-Pacific Software Engineering Conference*, IEEE, Jeju, South Korea, 2014.

[16] D. Rowe, J. Leaney, and D. Lowe, "Defining systems architecture evolvability-a taxonomy of change," in *Proceedings of the International Conference and Workshop: Engineering of Computer-Based Systems*, pp. 45–52, Jerusalem, Israel, December 1998.

[17] M. Perepletchikov, C. Ryan, and F. Keith, "Comparing the impact of service-oriented and object-oriented paradigms on the structural properties of software," *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 3762 LNCS*, pp. 431–441, Springer, Berlin, Germany, 2005.

[18] C.-H. Chen, F.-J. Hwang, and H.-Y. Kung, "Travel time prediction system based on data clustering for waste collection vehicles," *IEICE Transactions on Information and Systems*, vol. E102-D, no. 7, pp. 1374–1383, 2019.

[19] M. Perepletchikov, C. Ryan, and F. Keith, "Cohesion metrics for predicting maintainability of service-oriented software," in *Proceedings of the Seventh International Conference on Quality Software (QSIC 2007)*, IEEE, Portland, OR, USA, pp. 328–335, 2007.

[20] D. Rud, A. Schmietendorf, and R. R. Dumke, "Product metrics for service-oriented infrastructures," in *Proceedings of the IWSM/MetriKon*, Montreal, Canada, 2006.