UDC 303.064
DOI https://doi.org/10.32782/tnv-tech.2023.4.6

THE MODEL OF INTELLIGENT ORCHESTRATION OF WEB SERVICES USING THE EXAMPLE OF STATISTICAL RESEARCH

Kasianchuk I. V. *– Postgraduate Student*
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute"
ORCID ID: 0009-0000-2215-149X

In organizations engaged in analytical research, the production process almost always involves statistical calculations, which serve as statistical functions of the business. The presence of tools for creating a service-oriented architecture raises questions about the development of publicly accessible means to perform such functions by assembling them into a specific plan or algorithm for solving a particular statistical task.

To implement such statistical programs, it is worth considering a service-oriented architecture for creating software services that implement the execution of statistical business functions, from which a statistical program can be composed using metadata. One notable feature of such architecture is the ability to distribute tasks according to a specific area of service responsibility, also known as a domain. This architecture employs an orchestrator – a main service to which requests are made directly by users, and from which the request should reach its intended recipient.

As noted in [1], orchestration becomes more complex with an increasing number of web services in an application. The main challenge lies in establishing the connection between new services, their tasks, responsibility zones, and the orchestrator. Often, such a change is not feasible without human intervention, particularly from a programming expert who needs to reprogram the main service, including deployment and configuration, which takes a significant amount of additional time.

As an alternative to hard-coded interaction between services, [1] explores an approach in which discovery occurs through intelligent orchestration. The core of this approach involves each service providing its semantic description, which the orchestrator analyzes during discovery, and according to which user requests are redirected.

This article provides an example of an approach to intelligent orchestration of services for conducting statistical research. Using the issue discussed in [2] as an illustration, the primary objective is to create an application for aggregating statistical data results over a specific time interval, incorporating intelligent service discovery to provide query-based result delivery.

Key words: *service-oriented architecture, orchestration, statistical research.*

Касьянчук І. В. Модель розумної оркестрації веб сервісів на прикладі статистичних досліджень

У організаціях, що займаються аналітичними дослідженнями, виробничий процес майже завжди включає статистичні розрахунки, які є статистичними функціями бізнесу. Наявність інструментів для створення архітектури, орієнтованої на сервіси, викликає питання щодо розробки загальнодоступних засобів для виконання таких функцій, шляхом їх об'єднання в конкретний план або алгоритм для вирішення певної статистичної задачі.

Для реалізації таких статистичних програм варто розглянути сервіс-орієнтовану архітектуру, для створення програмних сервісів, які реалізують виконання статистичних бізнес-функцій, з яких можна скласти статистичну програму, використовуючи метадані. Особливістю такої архітектури є можливість розподілення завдань згідно певної області відповідальності сервісу, яку ще називають доменом. Такій архітектурі властиве використання оркестратора – головного сервісу, до якого виконується запит безпосередньо від користувача, та від якого запит має дістатися адресата.

Як зазначено у [1], оркестрація ускладнюється зі збільшенням кількості веб-сервісів в додатку. Основною проблемою є встановлення зв'язку між новими сервісами, їх задачами та зонами відповідальності, та оркестратором. Часто така зміна неможлива без втручання людини, яка є експертом в області програмування, якій потрібно перепрограмувати головний сервіс, включно з його розгортанням та налагодженням що забирає багато додаткового часу.

Як альтернативу жорсткому (hard-coded) кодуванню взаємодії між сервісами, в [1] розглянуто підхід згідно з яким виявлення відбувається за рахунок розумної оркестрації. Основою підходу є надання кожним з сервісів свого семантичного опису, який аналізується оркестратором при виявленні, та відповідно до якого запит перенаправляється від користувача.

У даній статті запропоновано приклад підходу до розумної оркестрації сервісів для виконання статистичних досліджень. На прикладі проблематики дослідження з [2], за мету головну взято створення додатку для агрегації результатів статистичних даних на проміжку часу, з розумним виявлення сервісу для надання результату за запитом.

Ключові слова: *сервіс-орієнтована архітектура, оркестрація, статистичне дослідження.*

Introduction. With the increase in the scale of business systems and the volume of tasks addressed by such systems, the demands on software have also risen. As a result, the service-oriented architecture has gained prominence, decomposing a large monolithic application into several smaller services, each specialized in performing a specific task for the application.

An important challenge in designing a system with a service-oriented architecture is selecting the methods of interaction between services. Two classes of interaction methods are distinguished based on their types: orchestration and choreography. As the number of services in a system grows, the control logic becomes more complex, necessitating a flexible solution for establishing control over the system.

Managing a system with a large number of services, coupled with a significant number of business tasks and their scale, presents a non-trivial challenge that requires attention from the very stage of designing the system architecture.

Main part of the research. The approach proposed in [1] involves the use of semantic descriptions of services, which serve as the basis for determining the subsequent path of the query and processing the collected data. According to the authors' concept, services are divided into two groups: Semantic Registry Services (SRS) and Cloud Orchestrator Services (COS).

The role of SRS is to register data about the semantics of services in order to define the further query path. This group functions as a sort of knowledge base about services that belong to the COS category. When there are changes in semantics, administrators input these changes directly into this group. The main tasks of this group include:

- registering and describing services;
- configuring orchestration options;
- filtering and expanding services;
- automatic scaling;
- storing and defining workflows;
- extracting data from the knowledge base.

Additionally, this service includes a user interface for convenient interaction through a browser.

The task of the other COS group is to interpret the workflows defined using SRS. It also supports their execution, including providing intermediate results and handling errors. The COS group has access to the service knowledge base of SRS and includes tasks such as:

- deploying new and modified business process definitions;
- allocating and releasing memory in the cloud for process execution;
- automatic execution and termination of workflows;
- support for execution statistics and resource monitoring (memory, CPU time, etc.).

A crucial aspect of building an intelligent orchestration system is the thoughtfulness of ontological description contracts. The design must incorporate the notion that the orchestrator should be constantly available, and changing it to define a new contract entity is highly undesirable (Blue/Green deployment partially addresses the problem but doesn't bypass the human factor when developing a new version of the orchestrator).

A necessary condition for a contract is compatibility of the language through which intelligent services will interact with each other. Well-known and widely used solutions like WSDL, SOAP protocol, and UDDI registry have proven their worth in this field. However, for a local and small application, it's sufficient to select a format without unnecessary specifications. For instance, in this article, JSON format is proposed for interaction in the context of a statistical application.

To extract statistical data for a time period, an application will involve two main services and an orchestrator (Figure 1). The question of populating databases belonging to these services is not addressed in this article.

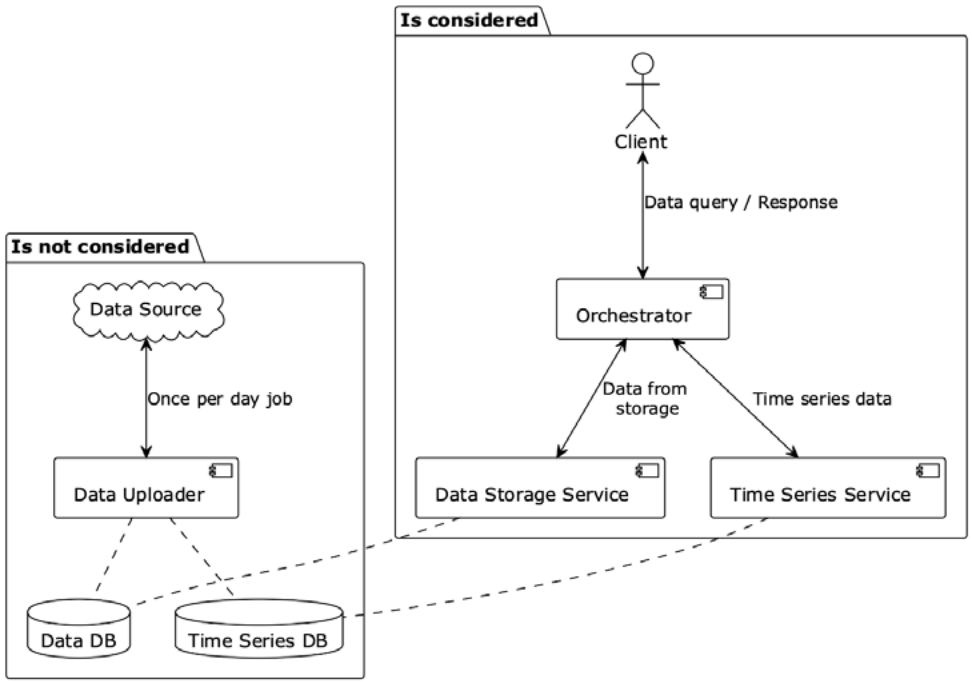

Fig. 1. Architecture of the application for extracting statistical data over time. This article does not cover the interaction between the data source and the database

The main purpose of the orchestrator is to divide the query based on service metadata, direct relevant queries to them, and aggregate the results.

In Figure 2, an example of metadata contract is presented, according to which intelligent discovery of service queries occurs. The idea is that a user provides a query with the data they expect to receive from the orchestrator. The query is then decomposed using information about registered services and executed sequentially. For example, to extract data about a statistical observation from the Data Storage service, identifiers of time periods are needed. The metadata of the service indicates that it is dependent

on the Time Series service, thus requiring the Time Series service to be queried first. After obtaining the necessary information from the Time Series service and validating it according to the schema, the intelligent orchestrator directs it to the data repository. As a result, the user receives aggregated query-specific data for further observations.

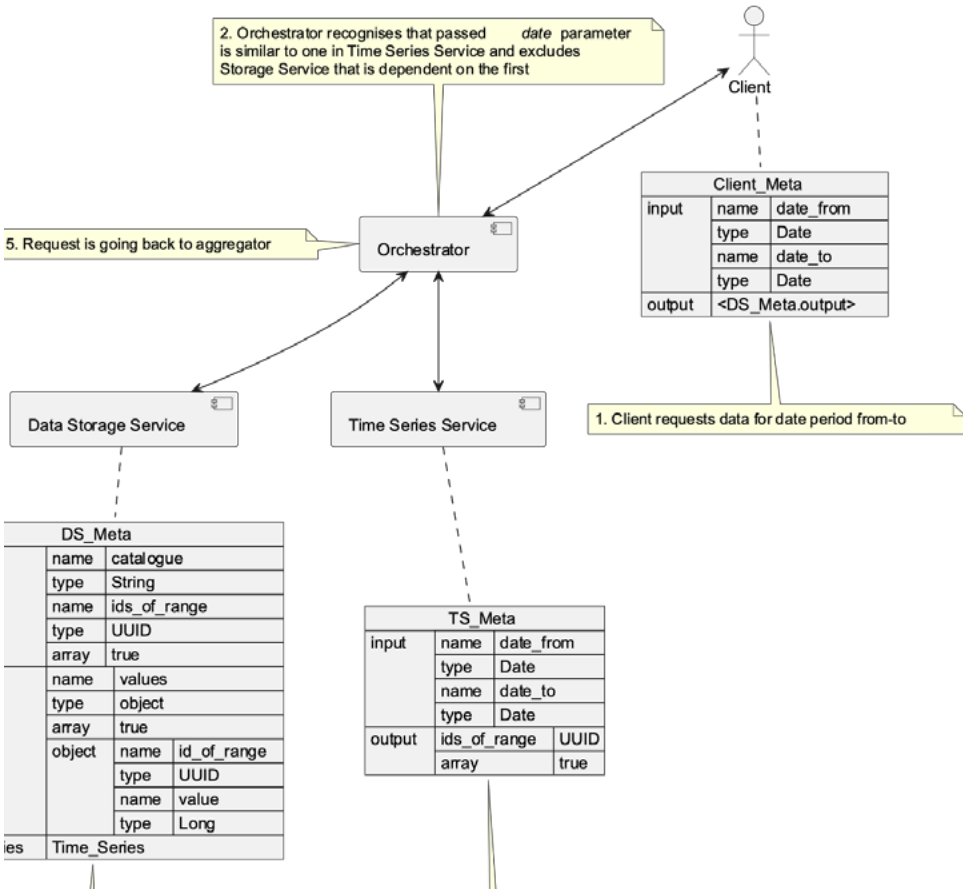

Fig. 2. Service Metadata for Obtaining Time-Sliced Information

Overall, the following fields can be distinguished in the service contract for obtaining statistical data:

- **input** – a list of input parameters for the service, including their names, data types, validation rules, and so on;
- **output** – the expected response from the service;
- **required** – determines whether the service needs to provide data. It indicates whether execution should stop in case of errors or absence of data;
- **dependencies** – a list of services that must be executed before the current service. This field allows building an execution graph for service requests;
- **version** – a number compared to the registered service description in the orchestrator. If it's different (usually higher), the old description is replaced with the provided one;

• **versionOfDependencies** – a list of associations in the format "dependency service name" = "version," needed in case the output data type of a dependent service changes;

• other additional fields depending on the task's specifics.

Another important aspect of orchestration is the correctness of service discovery. Since the contract of an individual service can be updated, it is necessary to validate it with each version change. This responsibility can be appropriately placed on the orchestrator, which has information about all services. Validation is proposed to be carried out based on the following criteria:

– **parameter names**: output parameters of a service must have names distinct from input parameters. Violating this rule can lead to parameter overlap in the orchestrator, resulting in incorrect execution of logic;

– **data types of parameters**: The output data types of services should match their corresponding input types. This rule leads to an issue when updating the contracts of two services simultaneously, where the input parameter of one becomes the output parameter of another. This scenario might require considering a mechanism to lock the contract's action using the `**versionOfDependencies**` parameter;

– input data of a non-optional service should not solely depend on the output parameters of optional services. It is necessary to account for a situation where all optional services return an error (no results), which means the mandatory service won't receive any input data.

Another advantage of a web service system with ontology-based orchestration is the possibility of parallelizing processes. Independent services can process information without waiting for results from others, which significantly improves execution time in multi-stage processes.

Further Research. The provided architecture example covers only a partial case of constructing a process for extracting statistical data over time. The list of potential improvements and research is not limited due to the diverse requirements of any information system. Among the issues not mentioned in this study, the following are worth highlighting:

– implementation of an aggregator for collected data from various services before returning the result;

– database population (data segregation) from data sources;

– implementation of service scalability, including parallel execution of algorithms on data;

– fault tolerance – orchestrator behavior in case of failure of a web service, including maintaining communication with it;

– process of updating the orchestrator's codebase;

– writing real data processing algorithms.

Conclusions. This article has presented an architecture example of intelligent service orchestration using the case of obtaining statistical data over time. The solution for service interaction through an orchestrator and the protocol for information exchange via metadata have been described.

Further research will explore expanding the system's capabilities for more complex data processing algorithms, as well as addressing other aspects such as fault tolerance, scalability, orchestrator updates, and more.

BIBLIOGRAPHY:

1. A. Petrenko and B. Bulakh, "Intelligent Service Discovery and Orchestration," 2018 IEEE First International Conference on System Analysis & Intelligent Computing (SAIC), Kyiv, Ukraine, 2018, pp. 1–5, doi: 10.1109/SAIC.2018.8516723.
2. Lumpova, T., & Kasianchuk, I. (2023). Finding a conceptual approach to developing an architecture of general-purpose services for economic researches. Technology Audit and Production Reserves, 3(4(71), 25–31. https://doi.org/10.15587/2706-5448.2023.283983
3. Лумпова Т. І. Перспективи створення статистичних сервісів загального користування для економічних досліджень / Т. І. Лумпова, І. В. Касьянчук. 2023. № 33. С. 26–39.

REFERENCES:

1. A. Petrenko and B. Bulakh (2018) "Intelligent Service Discovery and Orchestration", 2018 IEEE First International Conference on System Analysis & Intelligent Computing (SAIC), Kyiv, UKraine, pp. 1–5, doi: 10.1109/SAIC.2018.8516723.
2. Lumpova, T., & Kasianchuk, I. (2023). Finding a conceptual approach to developing an architecture of general-purpose services for economic researches. Technology Audit and Production Reserves, 3(4(71), 25–31. https://doi.org/10.15587/2706-5448.2023.283983
3. Lumpova T. I. (2023) Perspektyvy stvorennia statystychnykh servisiv zahalnoho korystuvannia dli ekonomichnykh doslidzhen / T. I. Lumpova, I. V. Kasianchuk. № 33. p. 26–39.