Формальный метод эволюции микросервисов на основании совместимости их API

Донская Софья Алексеевна

Студент

Факультет компьютерных наук, Национальный исследовательский университет «Высшая школа экономики», Москва, Россия

E-mail: sadonskaya@edu.hse.ru

Научный руководитель *— Хританков Антон Сергеевич*

Микросервисная архитектура является важным подходом в разработке современных программных систем, поскольку она позволяет реализовывать функциональность приложения в виде независимых компонентов, что повышает его гибкость и масштабируемость. Однако, разработка микросервисов требует тщательного подхода к управлению их эволюцией, чтобы сохранять их корректное взаимодействие [1].

Данные тезисы развивают представленные в [2] результаты анализа существующих решений для обеспечения совместимости микросервисов, эксперимента для выявления возможности представления взаимодействия микросервисов в виде графа и проверки сохранения совместимости при версионировании, а также представления API микросервисов в виде решетки концептов [3] с учетом принципа подстановки Барбары Лисков, согласно которому объекты суперкласса должны быть заменяемы объектами его подклассов, чтобы объекты подклассов были совместимы по поведению с объектами суперкласса [4].

Использование решеток концептов в REST API помогает обеспечить совместимость изменений. Принцип подстановки используется для обеспечения согласованности изменений с принципами подтипирования, наследования и полиморфизма [4].

Метод поддерживает следующие типы изменений. Во-первых, добавление новых точек подключения API (endpoints) как подтипов существующих, при условии совместимости их поведений. Во-вторых, изменение существующих точек подключения при их соответствии функциональным требованиям. В-третьих, удаление точек подключения при отсутствии нарушений функциональных требований API. Далее, добавление новых представлений ресурсов как подтипов существующих при условии совместимости их поведений. А также, изменение существующих представлений ресурсов при их соответствии

функциональным требованиям.

Также было выявлено, что в данном методе важны как спецификации типов, так и спецификации сигнатур. Спецификации типов определяют типы данных, которые могут быть переданы в API и возвращены из него. Спецификации сигнатур определяют методы, параметры и возвращаемые значения API. Вместе спецификации типа и сигнатур гарантируют, что API будет корректно определен и реализован, а также подтверждают, что изменения, внесенные в API, соответствуют требованиям принципа подстановки [4].

Для доказательства совместимости изменений в REST API используется комбинация математических доказательств, проверки с помощью внешнего инструмента и тестирования программы.

В данной работе был проведен эксперимент, в рамках которого были внесены изменения в проект PiggyMetrics. Для анализа был использован внешний инструмент jQAssistant. Совместимость вносимых изменений была подтверждена.

Таким образом, в работе предложен метод эволюции микросервисов. Данный метод основан на представлении API микросервисов в виде решеток концептов с учетом принципа подстановки. Метод применим для решения исходной проблемы эволюции микросервисов. Предложенный метод обеспечивает корректное версионирование сервисов посредством внесения поддерживаемых данным методом типов изменений. Дальнейшее исследование предполагает поиск подходящих для анализа проектов и реализацию метода для отслеживания их эволюции в рамках выявленных в данной работе типов изменений.

Литература

1. Alshuqayran N., Ali N., Evans R. A Systematic Mapping Study in Microservice Architecture // IEEE 9th International Conference on Service-Oriented Computing and Applications, Brighton, UK, 2016, P. 44–51
2. Донская С. А., Хританков А. С., Першин Н. В. О совместимости типов в рамках эволюции RESTful микросервисов // Сборник лучших докладов конференции ПКМ-2022, Санкт-Петербург, 2023, С. 312–316.
3. Ganter B., Wille R. Formal Concept Analysis: Mathematical Foundations. Springer Berlin, Heidelberg, 1999.
4. Liskov B. H., Wing J. M. Behavioral subtyping using invariants and constraints // Formal methods for distributed processing: a survey of object-oriented approaches. 2001, P. 254–280.