Оптимальные рефакторинги для сопровождаемости микросервисных систем

Першин Никита Вдадилавович

Студент

Физтех-школа прикладной математики и информатики, Московский физико-технический институт, Москва, Россия

E-mail: pershin.nv@phystech.edu

Научный руководитель *— Хританков Антон Сергеевич*

Сопровождаемость — степень эффективности с которой в систему можно вносить изменения для ее улучшения, исправления или адаптации к новым требованиям [1]. Темой данной работы является поиск оптимальных изменений структуры микросервисной системы для улучшения сопровождаемости. Актуальность работы обусловлена широким использованием микросервисной архитектуры и негативным влиянием плохой сопровождаемсоти на процесс разработки программного обеспечения. Текущие исследования фокусируются на двух аспектах: поиске высокоуровневых архитектурных проблем, посредством динамического анализа системы, и формировании фреймворка для поиска, анализа и внедрения изменений в архитектуру.

Рефакторинг — изменение структуры микросервисной системы, не затрагивающее поведение системы в целом.

Для поиска оптимальных рефакторингов используются следующие метрики сопровождаемости [1]: абсолютная важность сервиса (ACS), метрика связности интерфейса сервиса (TSIC), количество оконечных точек сервиса (WSIC), сумма всех операций для каждой из оконечных точек сервиса (TRS), метрика сбалансированности сервисов (CB)

Микросервисная система представляется в виде графа $G = (V, E)$, где вершины V сопоставлены элементам системы, а ребра E - вызовам элементов системы. Элементами микросервисной системы являются: элемент интерфейса REST API, метод для взаимодействия с REST API, адаптер источника данных, метод, содержащий бизнес-логику сервиса. Для поиска потенциальных рефакторингов задается полносвязный граф $H = (V, J)$, где вершины V сопоставлены элементам системы, а ребра J - связям между элементами.

На ребрах графа H задана метрика сходства Жаккара

$$J(i, j) = \frac{S_i \cap S_j}{S_i \cup S_j} \tag{1}$$

где $S_i \subset V$ - множество вершин, для которых существует исходящий из/ входящий в $v_i \in V$ путь в графах G и G^T, где G^T – транспонированный граф, все ребра которого заменены на обратные им.

При генерации рефакторингов рассматриваются три случая: первый — объединение сервисов, второй — изменение структуры системы без изменения количества сервисов, третий — разделение микросервиса на два. При объединении двух сервисов вызов оконечной точки внешнего сервиса заменяется вызовом метода внутри сервиса. При создании рефакторингов в остальных случаях используется эвристический алгоритм Feduccia-Mattheyses, который ищет разбиение графа H с минимальным разрезом [2]. Во втором случае за начальное разбиение берется текущее состояние системы.

Каждое из сгенерированных изменений является узлом в графе рефакторингов. Граф рефакторингов — ориентированный ациклический граф, на ребрах которого задано значение трудозатрат на проведение рефакторинга, а на вершинах значение метрики сопровождаемости после применения рефакторинга.

Тогда задача нахождения оптимального изменения задается следующим образом. Найти множество Парето-оптимальных решений, минимизируя трудозатраты и максимизируя сопровождаемость системы.

Для проверки алгоритма будет проведен эксперимент на примере системы микрсосервисов с открытым исходным кодом. Цель эксперимента — определить, позволяет ли предложенный алгоритм находить рефакторинги, повышающие метрики сопровождаемости системы.

Результатом работы алгоритма яаляется множество рефакторингов, применив один из которых разработчик сможет повысить сопровождаемость системы.

Литература

1. Bogner J., Automatically measuring the maintainability of service- and microservice-based systems: a literature review, In Proceedings of the 27th International Workshop on Software Measurement and 12th International Conference on Software Process and Product Measurement 107–115. 2017, October.
2. Träff J.L., Direct graph k-partitioning with a Kernighan–Lin like heuristic. Operations Research Letters, 2006, 621–629