DOI: 10.15514/ISPRAS–2021–33(1)–6

Паттерны микросервисной архитектуры: многопрофильный обзор литературы

Х.А. Вальдивия, ORCID: 0000-0003-4270-0462 <zs15011636@estudiantes.uv.mx>
А. Лора-Гонсалес, ORCID: 0000-0002-0154-7361 <zs15011639@estudiantes.uv.mx>
К. Лимон, ORCID: 0000-0003-4654-636X <hlimon@uv.mx>
К. Кортес-Вердин, ORCID: 0000-0002-6453-180X <kcortes@uv.mx>
Х.О. Очаран-Эрнандес, ORCID: 0000-0002-2598-1445 <jocharan@uv.mx>
Университет Веракрузана
Мексика, 91000, Веракрус, Халапа

Аннотация. Микросервисная архитектура позволяет разрабатывать распределенные системы с использованием набора независимых, легко связываемых, и пригодных к совместному использованию сервисов, удовлетворяющих насущным потребностям облачных вычислений. Каждый микросервис может быть реализован в различных технологиях с использованием общих каналов связи, что приводит к созданию гетерогенных распределенных систем, демонстрирующих высокую масштабируемость, эксплуатационную надежность, производительность и интероперабельность. В настоящее время существует множество вариантов построения микросервисов; некоторые из них основаны на паттернах, задающих общие структуры для решения повторяющихся задач. Тем не менее, поскольку микросервисы являются сравнительно новым трендом, взаимосвязи между атрибутами качества, метриками и паттернами четко не определены, что является проблемой с позиции программной инженерии, поскольку понимание этих взаимосвязей является фундаментом правильного проектирования систем с использованием этой архитектуры. Настоящая статья направлена на расширение знаний о проектировании микросервисных систем, представляя многопрофильный систематический обзор литературы о паттернах, касающихся микросервисов, и связывая их с атрибутами качества и метриками.

Ключевые слова: микросервисы; распределенные системы; архитектурные паттерны, паттерны проектирования; атрибуты качества.

Для цитирования: Вальдивия Х.А., Лора-Гонсалес А., Лимон К., Кортес-Вердин К., Очаран-Эрнандес Х.О. Паттерны микросервисной архитектуры: многопрофильный обзор литературы. Труды ИСП РАН, том 33, вып. 1, 2021 г., стр. 81-96. DOI: 10.15514/ISPRAS–2021–33(1)–6

Patterns Related to Microservice Architecture: a Multivocal Literature Review

J.A. Valdivia, ORCID: 0000-0003-4270-0462 <zs15011636@estudiantes.uv.mx>
A. Lora-González, ORCID: 0000-0002-0154-7361 <zs15011639@estudiantes.uv.mx>
X. Limón, ORCID: 0000-0003-4654-636X <hlimon@uv.mx>
K. Cortes-Verdin, ORCID: 0000-0002-6453-180X <kcortes@uv.mx>
J.O. Ocharán-Hernández, ORCID: 0000-0002-2598-1445 <jocharan@uv.mx>
Universidad Veracruzana,
Xalapa, Veracruz, México, 91000

Abstract. A Microservice Architecture enables the development of distributed systems using a set of highly cohesive, independent, and collaborative services, ready for current cloud computing demands. Each microservice can be implemented in different technologies, sharing common communication channels, which results in heterogeneous distributed systems that exhibit high scalability, maintainability, performance, and interoperability. Currently, there are many options to build microservices; some of them led by patterns that establish common structures to solve recurrent problems. Nevertheless, as microservices are an emerging trend, the relationship between quality attributes, metrics, and patterns is not clearly defined, which is a concern from a software engineering point of view, since such understanding is fundamental to correctly design systems using this architecture. This paper aims to extend the knowledge on the design of microservices-based systems by presenting a multivocal systematic literature review for microservices related patterns, tying them together with quality attributes and metrics, as can be found in academic and industry research.

Keywords: Microservices; Distributed Systems; Architectural Patterns; Design Patterns; Quality Attributes

For citation: Valdivia J.A., Lora-González A., Limón X., Cortes-Verdin K., Ocharán-Hernández J.O. Patterns Related to Microservice Architecture: A Multivocal Literature Review. Trudy ISP RAN/Proc. ISP RAS, vol. 33, issue 1, 2021, pp. 81-96 (in Russian). DOI: 10.15514/ISPRAS–2021–33(1)–6.

1. Введение

Микросервисная архитектура (MicroService Architecture, MSA) возникла в ответ на быстрые технологические изменения, проблемы расширяемости и масштабируемости, а также на необходимость сокращения времени поставки программного обеспечения. Эти проблемы послужили мотивацией нескольких технологических тенденций, основанных на облачных вычислениях [1-4]. MSA можно понимать, как набор небольших, разнородных, легко связываемых и распределенных сервисов, которые используют общие каналы связи для взаимодействия друг с другом. Успешные MSA-системы Netflix, Amazon и SoundCloud [5-7] заложили основу растущего интереса к этому архитектурному стилю, позиционируя его как альтернативу сервисно-ориентированной архитектуре (Service Oriented Architecture, SOA), из которой и происходит MSA [8]. Кроме того, реализация микросервисов обеспечивает мелкоструктурные сервисы, поддерживает гетерогенные технологии и обладает высокой отказоустойчивостью [9]; подобные свойства можно перевести в атрибуты качества, такие как удобство эксплуатации, переносимость и надежность.

Атрибуты качества имеют первостепенное значение при разработке программного обеспечения, поскольку уровень их достижимости принципиально влияет на успех программного проекта. Выбор архитектурного стиля, наряду с архитектурными решениями, принятыми в процессе проектирования, обеспечивает достижение атрибутов качества [10]. Атрибут качества – это измеримое и поддающееся проверке свойство системы, которое помогает определить, насколько хорошо система отвечает потребностям заинтересованных лиц [11]. Решения архитектурных задач должны быть сбалансированными, следует соблюдать баланс между набором атрибутов качества, определенных для системы, и преимуществами предлагаемого решения [12].

С атрибутами качества связаны паттерны, в которых определяются наборы многократно используемых структур для решения схожих задач в разных контекстах и обеспечивается единый словарь для сообщества разработчиков программного обеспечения [13]. Структуры паттернов представляют собой расширение архитектурных элементов, используемых для построения системы; аналогично, комбинирование нескольких паттернов облегчают разработку более сложных систем [14].

Использование паттерна направлено на решение задачи или улучшение атрибута качества, связанного с требованиями [10]. Для достижения требуемого атрибута качества можно использовать один или несколько шаблонов; разные комбинации могут обеспечить разные уровни качества, но определение правильной комбинации может оказаться сложной задачей. Метрики уменьшают сложность количественной оценки свойств системы, предоставляя числовое значение как меру того, насколько близок атрибут к желаемому уровню [15]. Значение, полученное с применением метрики, позволяет сравнивать паттерны и выбирать правильный шаблон для достижения намеченных целей; также снижается уровень субъективности при оценке влияния выбранного решение на другие атрибуты качества.

Появление паттернов и их принятие для практического использования является результатом основанного на опыте высокого уровня зрелости в области разработкт программного обеспечения работы. Например, широко известная работа Бушманна (Frank Buschmann) и др. [13] явилась результатом опыта работы авторов в компании Siemens. Они создали каталог решений для совершенствования процесса архитектурного проектирования. Аналогично, публикация [12] подытоживает предыдущий опыт работы авторов, предоставляя набор объектно-ориентированных паттернов [16]. Что касается микросервисной архитектуры, то, насколько нам известно, коллекции паттернов ограничены, поскольку микросервисы являются новым трендом. Однако мы можем найти исследования (например, [17-19] с многочисленными архитектурными паттернами, предоставляющими некоторый диапазон альтернатив. Среди них только в последней работе обеспечивается соответствие между атрибутами качества и паттернами. Поэтому трудно идентифицировать подходящий шаблон, соответствующий заданному атрибуту качества. Кроме того, сложно выявить атрибуты качества, на которые влияет выбранный шаблон.

Отметим также, что нахождение доступных паттернов для микросервисов является сложной задачей, поскольку связанная с паттернами информации скудна и рассеяна по белой и серой литературе, то есть академическим и промышленным публикациям соответственно.

Цель нашей работы состоит в том, чтобы представить существующую в настоящее время коллекцию шаблонов MSA, соответствующих атрибутам качества с сопутствующими метриками, которые, в свою очередь, указывают на уровень достижения атрибута качества конкретным паттерном. Такие шаблоны были получены путем многоаспектного анализа как белой, так и серой литературы. Наша работа будет полезна специалистам-практикам, нуждающимся в понимании того, какое влияние на процесс разработки оказывает выбор того или иного шаблона MSA, а также исследователям для дальнейшей работы в этой области. В нашу задачу не входит классификация выявленных шаблонов, мы стремимся лишь связать шаблоны с соответствующими атрибутами качества и метриками.

Данная статья является развитием предыдущей работы [20], в которой представлена коллекция паттернов 2014-2018 гг., полученная на основе систематического анализа литературы, то есть рассматривалась только белая литература. Новый обзор охватывает и паттерны 2019 года, и в нем дополнительно рассматриваются публикации категории серой литературы с 2014 по 2019 год на основе использования должным образом скорректированной методологии многопрофильного обзора литературы (Multivocal Literature Review, MLR).

В нашем MLR объединяются академические и производственные знания, обеспечивая расширенное представление о паттернах от практиков в области MSA. Обсуждение

паттерном с точек зрения отрасли и академии обеспечит более четкое понимание того, для чего используются шаблоны, а также роли атрибутов качества и метрик, используемых вместе с такими шаблонами.

Статья организована следующим образом. В разд. 2 излагается методология, используемая для многопрофильного обзора литературы и включающая вопросы, которыми мы задаемся при анализе публикаций, ограничения и критерии выбора источников. В разд. 3 представлены результаты, полученные путем применения методологии и включающие ответы на наши вопросы, а также нашу интерпретацию этих данных. В разд. 4 приводится сводка наиболее важных результатов нашей работы и описываются направления будущих исследований.

2. *Регламент работы над обзором*

В нашей работе используется подход MLR для использования данных серой литературы с целью предоставления полезной информации исследователям из производственного сообщества. Мы использовали рекомендации [21] для использования серой литературы и создания многопрофильных обзоров литературы по программной инженерии. Предлагаемые ниже этапы основаны на работе [22], где содержатся основные принципы систематических обзоров литературы по программной инженерии. Первым этапом является планирование обзора с целью выявления потребности в обзоре и уточнения вопросов, на которые требуется получить ответы. Второй этап работы над обзором включает в себя выработку стратегии поиска и критериев выбора источников, выбор источника, оценку качества исследований, представленных серой литературой, извлечение данных и синтез данных. Наконец, на третьем этапе описываются результаты в виде ответов на основные вопросы обзора.

2.1 Планирование обзора

Основной мотивацией нашей работы является количественное и качественное уточнение информации о паттернах, представленной в нашем предыдущем исследовании, на основе отраслевых знаний, углубленное изучение взаимосвязей между паттернами и атрибутами качества и исследование метрик, связанных с паттернами и используемых в индустрии. Обзор основан на трех исследовательских вопросах (Research Question, RQ) о паттернах проектирования в микросервисной архитектуре:

RQ1. Какие паттерны выявляются в микросервисных системах?

RQ2. Какие атрибуты качества связаны с паттернами, используемыми при разработке микросервисных систем?

RQ3. Какие метрики используются для количественной оценки атрибутов качества, связанных с паттернами микросервисов?

Вопрос **RQ1** направлен на непосредственное выявление паттернов микросервисов. **RQ2** является в нашем исследовании основным вопросом, поскольку ответы на этот вопрос позволяют получить детальную информацию об атрибутах качества на основании уникальных характеристик каждого паттерна, что помогает объяснить включение шаблонов для конкретных задач. Вопрос **RQ3** дает возможность исследовать метрики для количественного измерения уровня достижимости атрибутов качества, помогая лучше понять связи между паттернами MSA и атрибутами качества.

3. *Ведение процесса*

Процесс подготовки обзора включает ручной и автоматический поиск, изучение оценок качества серой литературы, извлечение данных, а также синтез данных с использованием метаэтнографического метода взаимной транслокации [23], который улучшает интерпретацию качественной информации.

3.1 Стратегия поиска и критерии выбора публикаций

Для сбора соответствующей информации в белой литературе использовался автоматический поиск в следующих службах индексирования: Scopus, Science Direct, Springer Link, IEEE Xplore и ACM. В то же время, для поиска в серой литературе использовалась поисковая система Google.

Для ограничения результатов применялись следующие критерии:

- критерии включения:
  - публикации с 2014 по 2019 гг.;
  - язык написания английский;
  - тип публикации – в серую литературу включались блоги для повышения качества источников и упрощения оценки;
- критерий исключения:
  - книга или глава из книги.

В дополнение к автоматическому поиску в качестве последнего шага применялся ручной поиск с использованием обратного метода снежного кома [24].

3.2 Выбор из источников

Процесс отбора проходил в пять этапов:

1) исключение по названию;
2) включение по обратному методу снежного кома для белой литературы;
3) исключение по контрольному списку для белой литературы / исключение по оценке качества для серой литературы;
4) исключение по ответу на исследовательские вопросы;
5) включение по обратному методу снежного кома для серой литературы.

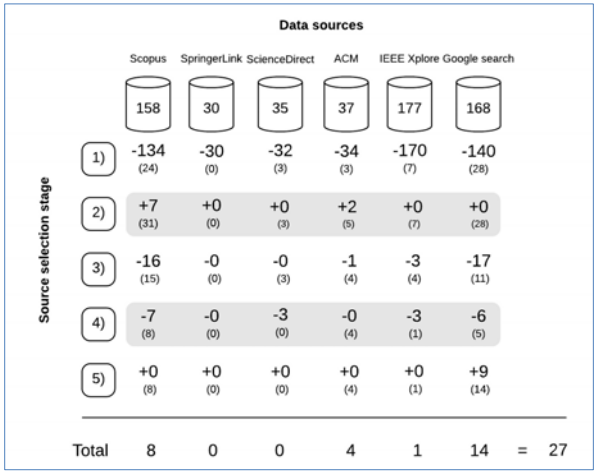

Рис. 1. Число работ для каждого источника на каждом этапе выбора источников. Каждое число показывает, насколько увеличилось или уменьшилось число статей, происходящих из данного источника представляет на данном этапе, а число в скобках показывает общее число статей из данного источника в конце данного этапа

Fig. 1. Data sources frequency against source selection stages. Each intersection represents the result between source selection stage and data source, and the numbers enclosed in parentheses highlight the remaining studies after each selection stage

Мы скорректировали процесс отбора, чтобы следовать рекомендациям по использованию серой литературы [21]. Оценка качества производилась для уменьшения предвзятости и валидации источника для серой литературы.

Был выполнен автоматический поиск по пяти источникам информации и в поисковой системе, в результате чего были получены 605 результатов, удовлетворяющих фильтрам стратегии поиска, как показано на рис. 1. После этого исключение на этапе 1 отбросило все документы с названиями, не связанными с темой исследования, что уменьшило количество публикаций до 65. Затем, на втором этапе в белой литературе мы нашли дополнительные девять публикаций. На этапе 3 после исключения по контрольному списку и оценки качества общее количество исследований было сокращено до 37. Наконец, на четвертом этапе было проведено полное чтение работ, и ответ хотя бы на один исследовательский вопрос был проверялся по другому контрольному списку, что сократило число работ до 18. С другой стороны, обратный метод снежного кома для серой литературы включил девять публикаций, что дало в общей сложности 27 источников для получения информации.

3.3 Оценка качества

В серой литературе отсутствует контроль качества загружаемых материалов и отсутствуют учреждения, проверяющие достоверность и объективность содержания. Следовательно, исследователи должны гарантировать качество включенных данных. Для этого мы применили контрольный список, предложенный в [21], к каждому из источников из серой литературы.

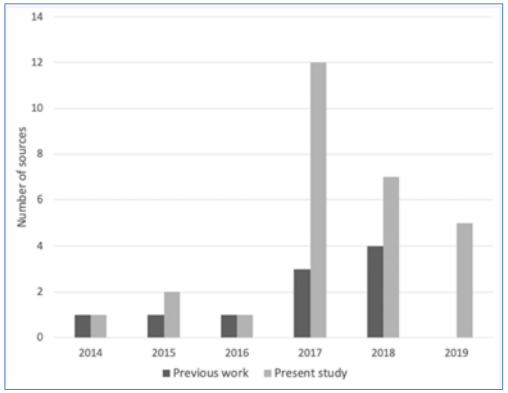

Рис. 2. Число работ, посвященных паттернам микросервисов; показаны также результататы нашего предыдущего исследования [20], которое не охватывает 2019 год и не использует серую литературу

Fig. 2. Microservice patterns trend, showing also results from our previous study [20], which does not cover 2019 and does not include grey literature.

4. Результаты и обсуждение

После завершения исследования мы можем составить представление об интересе к паттернам микросервисов в индустрии и академии. Число выбранных источников по годам приведено на рис. 2. Интересно, что в нашей предыдущей работе [20] с учетом только белой литературы больше всего работ было в 2018 году, но с учетом серой литературы пик интереса отмечается в 2017 году. Кроме того, как показано на рис. 3, включение серой литературы обеспечивает почти половину от общего числа публикаций, хотя учитывались только блоги. Между тем, в белой литературе наиболее частыми форматами были журнальные статьи и материалы конференций.

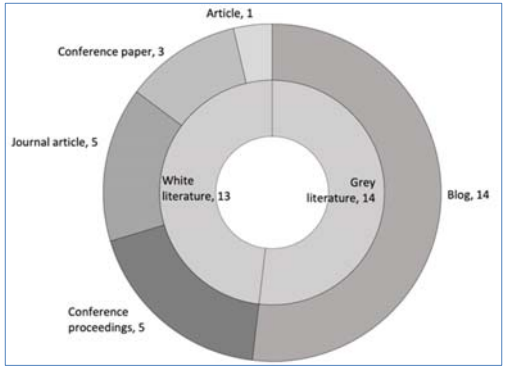

Fig. 3. Типы проанализированных публикаций
Fig. 3. Literature type identified in the sources analyzed..

В окончательно отобранных работах было выявлено 54 паттерна. Это довольно большое число, но следует отметить, что некоторые из паттернов представляют собой вариации или специализированные реализации других паттернов. Что касается атрибутов качества, мы использовали в качестве ориентира модель качества программного продукта, определенную в стандарте ISO/IEC 25010 [50]. Эта модель включает восемь характеристик качества. Однако мы обнаружили только шесть характеристик, имеющих отношение к паттернам микросервисов. Как видно на рис. 4, качественными характеристиками в порядке убывания числа упоминаний являлись удобство обслуживания, надежность, безопасность, эффективность, совместимость и переносимость. Никакой связи паттернов с атрибутами функциональной пригодности и удобства использования нами не обнаружено. Некоторые паттерны связаны с более чем одним атрибутом.

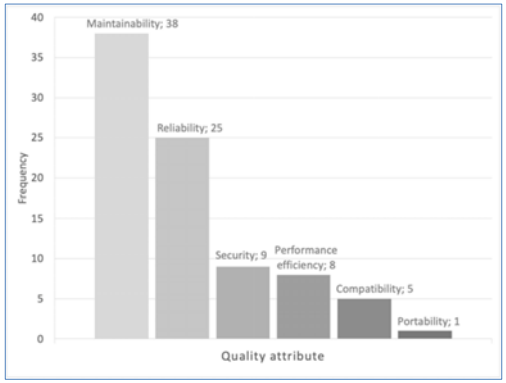

Рис. 4. Число упоминаний атрибутов качества, относящихся к выявленным паттернам
Fig. 4. Quality attributes frequency related to the identified patterns.

Табл. 1. Паттерны микросервисов, сгруппированные по основным преимуществам, и связанные с ними атрибуты качества
Table 1. Microservice patterns grouped by benefits and related quality attributes

Название группы	***Атрибут качества***	***Источник***
Постоянное хранение данных / Data Persistency		
Прокси локальной базы данных / Local Database Proxy	Удобство обслуживания и надежность	[25], [26]

Маршрутизатор на основе локального разделения данных / Local Sharding-based Router	Удобство обслуживания и эффективность	[25], [26]
Разделение ответственности за команды и запросы / Command and Query Responsibility Segregation	Удобство обслуживания, эффективность, надежность и безопасность	[27]–[29]
Порождение событий / Event Sourcing	Безопасность* и эффективность *	[27], [28], [30]
Масштабируемое хранилище / Scalable Store	Удобство обслуживания и надежность	[19]
Коммуникации		
Безопасный канал Secure Channel	Безопасность	[31]
Изменение зависимости уровня кода на зависимость уровня сервиса / Change Code Dependency to Service Call	Совместимость	[18], [32]
Реестр сервисов / Service Registry	Переносимость, удобство обслуживания и надежность	[18], [19], [30], [32]–[34]
Обнаружение сервисов / Service Discovery	Безопасность и надежность	[19], [35]
Локатор сервисов / Service Locator	Безопасность и удобство обслуживания *	[36]
Уведомление о событии / Event Notification	Надежность	[36]
Клиент реестра сервисов / Service Registry Client	Удобство обслуживания и надежность	[33]
Интеграция на основе REST / REST Integration	Удобство обслуживания и совместимость*	[18]
Конкурирующие потребители / Competing Customers	Удобство обслуживания	[25]
Каналы и фильтры / Pipes and Filters	Надежность и удобство обслуживания	[25]
Асинхронный обмен сообщениями / Asynchronous Messaging	Надежность, удобство обслуживания и эффективность	[19], [27], [28], [30], [34], [37]
Запрос-реакция / Request-Reaction	Удобство обслуживания*	[30]
Точка входа		
Шлюз API / API Gateway	Удобство обслуживания, надежность, эффективность и безопасность	[19], [27], [28], [30], [35], [38]–[41]
Отдельная серверная часть для каждого интерфейса / Backend for Frontend	Совместимость и удобство обслуживания	[18], [19], [30], [32], [35], [42]
Сервис аутентификации / Auth-Service	Надежность	[39]
Привратник / Gatekeeper	Безопасность	[25]
Распространение		
Душитель / Strangler	Совместимость	[28], [30], [43]
Уровень защиты от повреждений / Anti-Corruption Layer	Совместимость	[44]
Энтернализованная конфигурация /	Безопасность	[35]

Externalized Configuration		
Самодостаточность сервисов / Self-Containment of Services	Удобство обслуживания*	[18]
Контейнер / Container	Удобство обслуживания и надежность	[18], [19]
Развертывание кластера и оркестровка контейнеров / Deploy Cluster and Orchestrate Containers	Надежность и удобство обслуживания	[18]
DevOps микросервисов / Microservices DevOps	Удобство обслуживания *	[18], [19]
База данных является сервисом / Database is the Service	Удобство обслуживания и надежность	[19]
Разрешение непрерывной интеграции / Enable Continuous Integration	Удобство обслуживания	[19]
Самодостаточные системы / Self-Contained Systems	Удобство обслуживания и надежность	[30]
Отказоустойчивость		
Отсек / Bulkhead	Надежность	[45]
Автоматический выключатель / Circuit Breaker	Удобство обслуживания и надежность	[19], [28], [34]
Дополнения		
Очередь с приоритетами / Priority Queue	Удобство обслуживания	[25], [26]
Кеш результатов / Results Cache	Эффективность	[19], [32]
Кеш страниц / Page Cache	Эффективность	[19], [32]
Хранилище «ключ-значение» / Key Value Store	Безопасность и надежность	[19], [32]
Идентификатор корреляции / Correlation Id	Удобство обслуживания	[32]
Агрегатор журналов / Log Aggregator	Удобство обслуживания и эффективность работы	[19], [32]
Балансировщик нагрузки/Балансировка нагрузки / Load Balancer/Load-Balancing	Надежность и удобство обслуживания	[19], [33], [35], [41]
Посредник / Ambasador	Надежность и удобство обслуживания	[46]
Прицеп / Sidecar	Удобство обслуживания	[47]
Агрегатор шлюзов / Gateway Aggregator	Удобство обслуживания	[48]
Разгрузка шлюзов / Gateway Offloading	Удобство обслуживания и надежность	[49]
Асинхронный запрос / Asynchronous Query	Надежность	[36]
Маркер асинхронного завершения / Asynchronous Completion Token	Надежность	[36]
Пограничный сервер / Edge Server	Удобство обслуживания и надежность	[33]
Внутренний балансировщик нагрузки / Internal Load Balancer	Удобство обслуживания и надежность	[33]
Внешний балансировщик нагрузки / External Load Balancer	Удобство обслуживания и надежность	[33]

Проверка работоспособности / Health Check	Удобство обслуживания	[19]
Монитор / Monitor	Удобство обслуживания	[19]
Контракты, ориентированные на потребителя / Consumer-Driven Contracts	Удобство обслуживания *	[30]
Толерантный читатель / Tolerant reader	Удобство обслуживания *	[30]
Агрегатор / Aggregator	Удобство обслуживания	[28]
* В публикации явно не говорится, но следует из текста		

4.1 Анализ RQ1

Что касается паттернов, выявляемых в системах микросервисов, в табл. 1 представлены паттерны, сгруппированные по свойственным им преимуществам; паттерны указываются в хронологическом порядке по возрастанию времени. Большинство паттернов было выявлено в литературе, посвященной микросервисам, однако некоторые паттерны связаны также с SOA, которую можно считать более общим видом архитектуры распределенных служб [8]. К таким паттернам относятся, например, *проверка работоспособности*, *асинхронный обмен сообщениями*, *обнаружение сервисов*, автоматический выключатель и *монитор*. Это неудивительно, поскольку микросервисы можно понимать, как эволюцию SOA. SOA – это нецентрализованный архитектурный стиль, основанный на коммуникации сервисов для обеспечения требуемых функциональных возможностей и допускающий использование сервисов сторонних производителей [51]. Однако у SOA имеются проблемы с коммуникациями, промежуточным программным обеспечением и уровнем модульности; для смягчения этих проблем в качестве альтернативы были разработаны микросервисы [8].

Наиболее часто выявляемые паттерны связаны с решением коммуникационных задач и обеспечением координации между сервисами – например, *асинхронный обмен сообщениями*, *реестр сервисов* и *запрос-реакция*. В литературе чаще всего обнаруживаются паттерны *шлюз API*, *отдельная серверная часть для каждого интерфейса* и *реестр сервисов*. Поскольку термины «паттерн проектирования» и «архитектурный паттерн» у авторов четко не различаются, некоторые паттерны были у них отнесены к обеим категориям, а в некоторых случаях проходили под общим названием «паттерн». Например, в категории просто паттернов были обнаружены *разрешение непрерывной интеграции* [19] и *контейнер* [19], [25]. *Непрерывная интеграция* – это прием, связанный с гибкой (agile) разработкой программного обеспечения. Этот паттерн позволяет смягчить проблему так называемой «интеграции большого взрыва» (big-bang integration) с использованием инструментов для интеграции кода, запуска наборов тестов и создания артефактов развертывания [52]. Следование этому паттерну подразумевает деятельность по разработке, не связанную напрямую с системными структурами. Уровень абстракции *контейнера* может быть ограничен применяемой технологией, поскольку речь идет о специализированной реализации архитектурного стиля виртуализации [51].

В ходе обобщения данных нами были выявлены пять основных преимуществ использования паттернов. Эти преимущества позволили нам отнести каждый шаблон к одной из следующих групп: постоянное хранение данных, коммуникации, точка входа, распространение, отказоустойчивость и дополнительные компоненты. Каждая группа демонстрирует общие преимущества паттернов и облегчает интерпретацию результатов, но эти группы не предназначены для классификации паттернов.

4.2 Анализ RQ2

Как отмечалось выше, вопрос RQ2 касается атрибутов качества, связанных с паттернами, используемыми в микросервисах. Атрибуты качества, выявленные в литературе, заведомо связаны с MSA. В табл. 1 представлены атрибуты качества, свойственные каждому выявленному паттерну. Для сопоставления паттернов и атрибутов качества мы непосредственно в публикациях определяли атрибуты, связанные с паттернами. Однако не все найденные паттерны имели документированную связь с атрибутами качества; в этих случаях мы анализировали характеристики паттернов и консультировались с дополнительным исследователем, имеющим опыт работы с микросервисами.

На рис. 4 первые два наиболее часто встречающихся атрибута качества составляют почти 75% от общего количества, а остальные четыре атрибута – всего 25% совместно. Высокая частота упоминаний удобства обслуживания и надежности может рассматриваться как принципа быстрого и простого развертывания микросервисных систем, а также внимания к обеспечению функциональных возможностей, удовлетворяющих требованиям качества. Однако не все аспекты MSA оказались связанными с паттернами. Например, характеристика удобства обслуживания была наиболее часто встречающимся атрибутом качества, но ни в одном описании этого паттерна не упоминалась подхарактеристика тестируемости, несмотря на то, что тестирование является одним из важнейших аспектов MSA. Кроме того, некоторые паттерны связаны с тремя или четырьмя атрибутами качества; это может поставить под сомнение возможность полного удовлетворения каждого атрибута качества с учетом требуемых компромиссов.

4.3 Анализ RQ3

Наконец, вопрос RQ3 касается метрик, используемых для количественной оценки атрибутов качества, связанных с паттернами микросервисов. В литературе удалось выявить следующие метрики, связанные с оценкой атрибутов качества в паттернах:

- время (эффективность);
- процент принятых запросов (интероперабельность);
- число файлов, которые нужно изменить (удобство обслуживания);
- энергопотребление (эффективность).

Были выявлены три особых аспекта, касающихся времени: время ответа для выполнения определенного количества запросов в секунду, время взаимодействия в миллисекундах (время до того, как пользователь сможет взаимодействовать со своим графическим интерфейсом) и время ответа в миллисекундах. Что касается времени, в 2014 г. в [26] были проанализированы паттерны *прокси локальной базы данных*, *маршрутизатора на основе локального разделения данных* и *очереди с приоритетами*. Эти три шаблона положительно влияют на скорость выполнения запросов на чтение и запись данных. Первый шаблон больше подходит для запросов на чтение, второй – для записи, а последний оказывает умеренно положительное влияние на оба вида запросов. Разумно комбинировать последний паттерн с одним из первых двух. Кроме того, в этих трех паттернах выбор алгоритмов не дает большой разницы в скорости, поскольку ключевым моментом является принятие правильного решения в соответствии с паттерном.

В 2017 г. в [39] паттерны *самодостаточности* и *шлюза API* были оценены на предмет удобства обслуживания. Первый паттерн лучше второго с точки зрения числа файлов, которые нужно изменить, и процентного отношения приемлемых модификаций. У шлюза API имеются недостатки при наличии высоких нагрузок. Что касается числа миллисекунд, при небольшом количестве запросов оба шаблона показывали одно и то же время, но при большом числе запросов *шлюз API* требует немного больше времени.

В 2018 г. в [25] паттерны *очереди с приоритетами*, *прокси локальной базы данных*, *маршрутизатора на основе локального разделения данных*, *каналов и фильтров*, *конкурирующих потребителей* и *привратника* оценивались по потреблению энергии. Энергопотребление измерялось на уровне процессов с использованием Power-API, интерфейса прикладного программирования, написанного на Java API для отслеживания потребляемой энергии. В целом, все пять паттернов ведут к сокращению потребления энергии, измеряемой в килоджоулях, и времени, измеряемого в миллисекундах, повышая эффективность работы микросервисов по сравнению с монолитными системами. Паттерн *каналы и фильтры* сокращает время работы и снижают потребление энергии. К удивлению, этот паттерн не дает тех же преимуществ в монолитных системах. Точно так же эти преимущества могут быть получены с использованием паттернов *привратника* и *конкурирующих потребителей*, но отдельная реализация привратника может отрицательно повлиять на время отклика и потребление энергии. Комбинация *прокси локальной базы данных* и *конкурирующих потребителей* отрицательно влияет на время отклика, как и комбинация *конкурирующих потребителей*, *конкурирующих потребителей* и *каналов и фильтров*.

К сожалению, анализ публикаций 2019 года и учет отраслевой информации не позволили добавить какие-либо новые метрики к тому, что мы изложили в [20]. В соответствии с данными серой литературы, при выборе паттерна могут вообще не применяться количественные измерения, хотя можно усомниться в том, что в этом деле можно обойтись только опытом. Как следствие, можно сделать вывод о низком уровне зрелости метрик MSA в серой литературе.

5. Заключение

Проведение MLR помогло расширить список паттернов, представленных в нашей предыдущей работе [20]. При включении в обзор публикаций серой литературы было обнаружено 20 новых паттернов. Кроме того, в некоторых источниках серой литературы представлена дополнительная информация о паттернах, поскольку авторы не ограничены количеством страниц или слов. Например, Microsoft подробно описывает следующие аспекты каждого паттерна: контекст, проблему, решение, спорные моменты, соображения и, в некоторых случаях, примеры с кодом.

Изучение «серой» литературы требует дополнительной работы, например, привлечения контрольного списка, предложенного в [21] для оценки качества. Эту оценку необходимо производить для каждой выбранной публикации серой литературы. Некоторые вопросы в контрольном списке являются субъективными, а другие требуют дополнительного поиска. Еще один недостаток – огромное количество результатов поиска в серой литературе. На первом этапе поиска было найдено 2 640 000 результатов, хотя использовалась поисковая строка. Чтобы проанализировать наиболее релевантные для исследования результаты, мы использовали алгоритм ранжирования Google. Однако нерелевантный материал все же предоставлялся, например, ссылки на покупку книг, курсов или семинаров.

В белой литературе пояснения по поводу паттернов были краткими и не прямыми, авторы обычно не объясняют контекст и метод, использованный для получения информации. Напротив, в серой литературе авторы были более прямыми: если обсуждаемый вопрос требовал какого-либо пояснения контекста, авторы приводили ссылки на другие ресурсы, предоставляющие необходимую информацию. Другой момент заключается в том, что в анализируемой литературе уровень абстракции термина «паттерн» может быть неясен. Некоторые из описываемых паттернов не являются, строго говоря, паттернами, например, *разрешение непрерывной интеграции* [19] и *контейнер* [19], [25].

Таким образом, включение в обзор серой литературы принесло такие преимущества, как добавление в каталог новых паттернов, но также добавилась и дополнительная работа для

обеспечения качества. Несмотря на увеличение количества паттернов, представленный список не следует считать окончательным. В будущей работе ожидается включение большего числа типов публикаций, например, книг, материалов конференций и фирменных описаний. Кроме того, мы изучаем различные методы уменьшения количества нерелевантных публикаций при использовании серой литературы.

Список литературы / References

[1] Д.А. Грушин, Н.Н. Кузюрин. О задаче эффективного управления вычислительной инфраструктурой. Труды ИСП РАН, том 30, вып. 6, 2018 г., стр. 123-142. DOI: 10.15514/ISPRAS-2018-30(6)-7 / D.A. Grushin and N.N. Kuzyurin. On Effective Scheduling in Computing Clusters. Programming and Computer Software, vol. 45, no. 7, 2019, pp. 398–404.

[2] Б.М. Шабанов, О.И. Самоваров. Принципы построения межведомственного центра коллективного пользования общего назначения в модели программно-определяемого ЦОД. Труды ИСП РАН, том 30, вып. 6, 2018 г., стр. 7-24. DOI: 10.15514/ISPRAS-2018-30(6)-1 / B.M. Shabanov and O.I. Samovarov. Building the Software-Defined Data Center. Programming and Computer Software, vol. 45, no. 8, 2019, pp. 458–466.

[3] Вл.В. Воеводин, Н. Н. Попова. Инфраструктура суперкомпьютерных технологий. Программирование, том 45, no 3, 2019, стр. 6–13 / V.V Voevodin and N.N. Popova. Infrastructure of Supercomputing Technologies. Programming and Computer Software, vol. 45, no. 3, 2019, pp. 89–95.

[4] А.П. Крюков, А.П. Демичев. Децентрализованные хранилища данных: технологии построения. Программирование, том 44, no. 5, 2018, стр. 12–30. / A.P. Kryukov and A.P. Demichev. Decentralized Data Storages: Technologies of Construction. Programming and Computer Software, vol. 44, no. 5, 2018, pp. 303–315.

[5] T. Hoff. Amazon Architecture. High Scalability, 2007. [Online]. Available: http://highscalability.com/blog/2007/9/18/amazon-architecture.html. [Accessed: 07-Nov-2018].

[6] Netflix Technology Blog. Netflix Conductor: A microservices orchestrator. Netflix Techblog, 2016. [Online]. Available: https://medium.com/netflix-techblog/netflix-conductor-a-microservices-orchestrator-2e8d4771bf40. [Accessed: 07-Nov-2018].

[7] P. Calçado. Building Products at SoundCloud – Part I: Dealing with the Monolith. Developers Soundcloud, 2014. [Online]. Available: https://developers.soundcloud.com/blog/building-products-at-soundcloud-part-1-dealing-with-the-monolith. [Accessed: 03-Nov-2018].

[8] S. Newman, Building Microservices: Designing Fine-Grained Systems, 1st ed. O'Reilly Media, 2015, 280 p.

[9] J. Lewis and M. Fowler. Microservices – A definition of this new architectural term. Martinfowler.Com, 2014. [Online]. Available: http://martinfowler.com/articles/microservices.html.

[10] P. Clements, R. Kazman, and M. Klein. Evaluating Software Architectures: Methods and Case Studies. Addison-Wesley, 2001, 368 p.

[11] L. Bass, P. Clements, and R. Kazman. Software Architecture in Practice, 3rd ed. Addison-Wesley Professional, 2012, 624 p.

[12] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Longman Publishing Co., Inc., 1995, 416 p.

[13] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal. Pattern-Oriented Software Architecture – Volume 1: A System of Patterns. Wiley Publishing, 1996, 476 p.

[14] D.C. Schmidt, M. Stal, H. Rohnert, and F. Buschmann. Pattern-Oriented Software Architecture, Volume 2: Patterns for Concurrent and Networked Objects. Wiley, 2000, 666 p.

[15] K.H. Möller and D.J. Paulish. Software metrics: a practitioner's guide to improved product development. IEEE Computer Society Press, 1993, 257 p.

[16] P. Clements, F. Bachmann et al. Documenting Software Architectures: Views and Beyond, 2nd ed. Addison-Wesley Professional, 2010, 592 p.

[17] D. Taibi, V. Lenarduzzi, and C. Pahl. Architectural Patterns for Microservices: A Systematic Mapping Study. In Proc. of the 8th International Conference on Cloud Computing and Services Science, vol. 1, 2018, pp. 221–232.

[18] F. Osses, G. Márquez, and H. Astudillo. Exploration of academic and industrial evidence about architectural tactics and patterns in microservices. In Proc. of the 40th International Conference on Software Engineering, 2018, pp. 256–257.

[19] G. Marquez and H. Astudillo. Actual Use of Architectural Patterns in Microservices-Based Open Source Projects. In Proc. of the 25th Asia-Pacific Software Engineering Conference (APSEC), 2018, pp. 31–40.

[20] J.A. Valdivia, X. Limón, and K. Cortes-Verdin. Quality attributes in patterns related to microservice architecture: A Systematic Literature Review. In Proc. of the 7th International Conference in Software Engineering Research and Innovation (CONISOFT), 2019, pp. 181–190.

[21] V. Garousi, M. Felderer, and M. V. Mäntylä. Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. Information and Software Technology, vol. 106, 2019, pp. 101–121.

[22] B. Kitchenham and S. Charters. Guidelines for performing Systematic Literature Reviews in Software Engineering. Technical Report EBSE-2007-01, Keele University, University of Durham, 2007.

[23] G.W. Noblit and R.D. Hare. Meta-ethnography: Synthesizing qualitative studies. Counterpoints, vol. 44, 1988, pp. 93-123.

[24] C. Wohlin. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In Proc. of the 18th International Conference on Evaluation and Assessment in Software Engineering, 2014, article no. 38.

[25] F. Khomh and S.A. Abtahizadeh. Understanding the impact of cloud patterns on performance and energy consumption. Journal of Systems and Software, vol. 141, 2018, pp. 151–170.

[26] G. Hecht, B. Jose-Scheidt, C. De Figueiredo, N. Moha, and F. Khomh. An Empirical Study of the Impact of Cloud Patterns on Quality of Service (QoS). In Proc. of the IEEE 6th International Conference on Cloud Computing Technology and Science, 2014, pp. 278–283.

[27] K. Aram. A Microservices implementation journey – Part 1. Medium Koukia blog, 2018. [Online]. Available: https://koukia.ca/a-microservices-implementation-journey-part-1-9f6471fe917. [Accessed: 12-Jan-2020].

[28] K. Sahiti. Everything You Need to Know About Microservices Design Patterns. Edureka blog, 2019. [Online]. Available: https://www.edureka.co/blog/microservices-design-patterns. [Accessed: 12-Jan-2020].

[29] Microsoft. Responsibility Segregation (CQRS) pattern. Azure Architecture Center, 2019. [Online]. Available: https://docs.microsoft.com/en-us/azure/architecture/patterns/cqrs. [Accessed: 13-Jan-2020].

[30] J. Bogner, J. Fritzsch, S. Wagner, and A. Zimmermann. Assuring the Evolvability of Microservices: Insights into Industry Practices and Challenges. In Proc. of the IEEE International Conference on Software Maintenance and Evolution (ICSME), 2019, pp. 546-556.

[31] A.K. Dwivedi and S.K. Rath. Incorporating Security Features in Service-Oriented Architecture using Security Patterns. ACM SIGSOFT Software Engineering Notes, vol. 40, no. 1, 2015, pp. 1-6.

[32] K. Brown and B. Woolf. Implementation patterns for microservices architectures. In Proc. of the 23th International Conference on Pattern Languages of Programs, 2016, 35 p.

[33] A. Balalaie, A. Heydarnoori, P. Jamshidi, D. A. Tamburri, and T. Lynn. Microservices migration patterns. Software Practice and Experience, vol. 48, no. 11, 2018, pp. 2019-2042.

[34] G. Márquez and H. Astudillo. Identifying availability tactics to support security architectural design of microservice-based systems. In Proc. of the 13th European Conference on Software Architecture, vol. 2, 2019, pp. 123-129.

[35] K.A. Torkura, M.I.H. Sukmana, F. Cheng, and C. Meinel. Leveraging Cloud Native Design Patterns for Security-as-a-Service Applications. In Proc. of the 2nd IEEE International Conference on Smart Cloud, 2017, pp. 90–97.

[36] G. Rodríguez, J. A. Díaz-Pace, and Á. Soria. A case-based reasoning approach to reuse quality-driven designs in service-oriented architectures. Information Systems, vol. 77, 2018, pp. 167–189.

[37] Microsoft. Designing a microservice-oriented application. .NET Microservices: Architecture for Containerized .NET Applications, 2018. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/architecture/microservices/multi-container-microservice-net-applications/microservice-application-design. [Accessed: 12-Jan-2020].

[38] C. Richardson. Building Microservices: Using an API Gateway. Nginx blog, 2015. [Online]. Available: https://www.nginx.com/blog/building-microservices-using-an-api-gateway/. [Accessed: 12-Jan-2020].

[39] H. Harms, C. Rogowski, and L. Lo Iacono. Guidelines for adopting frontend architectures and patterns in microservices-based systems. In Proc. of the 11th Joint Meeting on Foundations of Software Engineering, 2017, pp. 902-907.

[40] Microsoft. Gateway Routing pattern. Azure Architecture Center, 2017. [Online]. Available: https://docs.microsoft.com/en-us/azure/architecture/patterns/gateway-routing. [Accessed: 13-Jan-2020].

[41] D. Müssig, R. Stricker, J. Lässig, and J. Heider. Highly Scalable Microservice-based Enterprise Architecture for Smart Ecosystems in Hybrid Cloud Environments. Proc. In Proc. of the International Conference on Enterprise Information Systems, vol. 1, 2017, pp. 454-459.
[42] Microsoft. Backends for Frontends pattern. Azure Architecture Center, 2017. [Online]. Available: https://docs.microsoft.com/en-us/azure/architecture/patterns/backends-for-frontends. [Accessed: 12-Jan-2020].
[43] Microsoft. Strangler pattern. Azure Architecture Center, 2017. [Online]. Available: https://docs.microsoft.com/en-us/azure/architecture/patterns/strangler. [Accessed: 13-Jan-2020].
[44] Microsoft. Anti-Corruption Layer pattern. Azure Architecture Center, 2017. [Online]. Available: https://docs.microsoft.com/en-us/azure/architecture/patterns/anti-corruption-layer. [Accessed: 12-Jan-2020].
[45] Microsoft. Bulkhead pattern. Azure Architecture Center, 2017. [Online]. Available: https://docs.microsoft.com/en-us/azure/architecture/patterns/bulkhead. [Accessed: 13-Jan-2020].
[46] Microsoft. Ambassador pattern. Azure Architecture Center, 2017. [Online]. Available: https://docs.microsoft.com/en-us/azure/architecture/patterns/ambassador. [Accessed: 12-Jan-2020].
[47] Microsoft. Sidecar pattern. Azure Architecture Center, 2017. [Online]. Available: https://docs.microsoft.com/en-us/azure/architecture/patterns/sidecar. [Accessed: 13-Jan-2020].
[48] Microsoft. Gateway Aggregation pattern. Azure Architecture Center, 2017. [Online]. Available: https://docs.microsoft.com/en-us/azure/architecture/patterns/gateway-aggregation. [Accessed: 13-Jan-2020].
[49] Microsoft. Gateway Offloading pattern. Azure Architecture Center, 2017. [Online]. Available: https://docs.microsoft.com/en-us/azure/architecture/patterns/gateway-offloading. [Accessed: 13-Jan-2020].
[50] ISO/IEC 25010: 2011 Systems and Software Engineering-Systems and Software Quality Requirements and Evaluation (SQuaRE) – System and Software Quality Models. ISO, 2011.
[51] R.N. Taylor, N. Medvidovic, and E. M. Dashofy. Software Architecture: Foundations, Theory, and Practice. Wiley Publishing, 2009, 750 p.
[52] C. Carneiro and T. Schmelmer. Microservices from Day One: Build Robust and Scalable Software from the Start. Apress, 2016, 268 p.

Информация об авторах / Information about authors

Хосе Али ВАЛЬДИВИЯ, бакалавр в области программной инженерии. Область научных интересов: архитектура программного обеспечения, распределенные системы, паттерны проектирования.

José Ali VALDIVIA, Bachelor of Software Engineering. Research interests: Software architecture, distributed systems, design patterns.

Алонсо ЛОРА-ГОНСАЛЕС, бакалавр в области программной инженерии, профессиональный разработчик программного обеспечения. Область научных интересов: программные архитектуры, паттерны проектирования и распределенные системы.

Alonso LORA-GONZALEZ, Bachelor degree in Software Engineering. Professional Software developer. Research interests: Software Architectures, Design patterns and Distributed Systems.

Ксавье ЛИМОН, кандидат наук в области искусственного интеллекта, доцент факультета статистики и информатики. Область научных интересов: распределенные системы, архитектуры программного обеспечения, многоагентные системы, машинное обучение.

Xavier LIMÓN, Doctor of Artificial Intelligence, Associate Professor of the Statistics and Informatics Faculty. Research interests: Distributed Systems, Software Architectures, Multi-agent systems, Machine Learning.

Карен КОРТЕС-ВЕРДИН, кандидат компьютерных наук, профессор факультета статистики и информатики. Область научных интересов: программная инженерия, качество программного обеспечения, семейства программных продуктов.

Karen CORTES-VERDIN, Doctor in Computing Sciences, Professor at the School of Statistics and Informatics. Research interests: software engineering, software quality, software product lines.

Хорхе Октавио ОЧАРАН-ЭРНАНДЕС, кандидат компьютерных наук, профессор факультета статистики и информатики. Область научных интересов: разработка программного обеспечения, архитектура программного обеспечения, разработка требований, разработка API.

Jorge Octavio OCHARÁN-HERNÁNDEZ, Doctor in Computing Sciences, Professor at the School of Statistics and Informatics. Research interests: software engineering, software architecture, requirements engineering, API design.