ВЕСТНИК ПЕРМСКОГО УНИВЕРСИТЕТА
2021 Математика. Механика. Информатика Вып. 2(53)

УДК 519.688

# Стратегия миграции программного кода из монолитной архитектуры в микросервисы

**Д. К. Радостев, Е. Ю. Никитина**
Пермский государственный национальный исследовательский университет
Россия, 614990, г. Пермь, ул. Букирева, 15
radostevd3@gmail.com; +79194741092

Целью данной работы является описание стратегии, которая сможет помочь предприятиям с переходом от монолитной архитектуры программного кода приложений к архитектуре микросервисов. Используя эту стратегию миграции, новая система получит ряд преимуществ, предлагаемых архитектурой микросервисов, таких как масштабируемость и ремонтопригодность. Компании смогут перенести свои старые системы в более гибкие, увеличивая при этом производительность своего программного обеспечения.

**Ключевые слова:** *программный код; архитектура приложения; монолитная архитектура; микросервисы.*

DOI: 10.17072/1993-0550-2021-2-65-68

## Введение

В последние годы набирает популярность микросервисная архитектура программного обеспечения. Микросервисы можно рассматривать как метод разработки программных приложений, которые, наследуя принципы и концепции стиля сервис-ориентированной архитектуры (SOA), позволяют структурировать сервис-ориентированное приложение как набор очень небольших слабо связанных программных сервисов. Архитектуру микросервисов можно рассматривать как новую парадигму программирования приложений с помощью композиции небольших сервисов, каждый из которых выполняет свои собственные процессы и взаимодействует с помощью легких механизмов. Ключевыми особенностями микросервисной архитектуры являются ограниченный контекст, гибкость и модульность. В этом отношении микросервисы позволяют проектировать и разрабатывать приложения, которые являются легко ремонтируемыми и масштабируемыми.

В монолитной архитектуре функции инкапсулируются в одно приложение [1]. Монолитные приложения до момента существенного увеличения в объеме имеют свои сильные стороны, например легкость разработки, тестирования и развертывания. Однако когда приложение имеет тенденцию расширяться, структура монолита увеличивается в размерах, превращаясь в большую, трудноуправляемую и плохо масштабируемую часть программного обеспечения.

Проблема состоит в том, что на сегодняшний день многие компании все еще используют свое программное обеспечение в виде монолита, со всеми проблемами, связанными с этой архитектурой. Большинство из этих программ трудно поддаются управлению и развитию, что вынуждает компании покупать новое программное обеспечение вместо разработки новых функциональных возможностей.

Миграция монолитной архитектуры в микросервисы может решить эти проблемы, вернуть приложению возможность развития, а также получить остальные преимущества, связан-ные с микросервисной архитектурой.

В настоящей работе обсуждается стратегия миграции, которая позволит перейти от монолитной системы к микросервисной.

Монолитные архитектуры в основном состоят из ряда функций, а последовательность вызовов функций может быть определена как бизнес-функциональность. Описываемая стратегия будет сосредоточена на бизнес-функциях. Методология состоит из пяти этапов. Она начинается с анализа функций монолита, проходит через анализ бизнес-функций и заканчивается созданием микро-сервисной архитектуры.

## 1. Преимущества перехода от монолитной архитектуры к микросервисам

Монолитная архитектура на данный момент является традиционным способом разработки программного обеспечения, которым в недавнем прошлом пользовались крупные компании.

В монолитной архитектуре функции инкапсулируются в одном приложении. Когда монолитное приложение небольшое по размеру и имеет всего несколько функций, он обладает своими сильными сторонами, например, простота разработки, тестирования, развертывания и масштабирования. Если, например, необходимо больше вычислительных возможностей, достаточно продублировать весь монолит, и при небольшом размере системы будут лишь небольшие накладные расходы.

Слабые стороны этой системы проявляются, когда монолит начинает обновляться и развиваться, увеличиваясь в размере и количестве функций. Как только это произойдет, недостатки монолитной архитектуры перевесят его преимущества.

Например, если требуется больше вычислительных возможностей, масштабирование большого монолита приведет к большим накладным расходам по сравнению с небольшим монолитом. Еще один недостаток, вызванный большим размером монолита, заключается в том, что разработка может замедлиться и, как следствие, стать препятствием для непрерывного развертывания из-за более длительного времени запуска.

У микросервисов есть множество преимуществ по сравнению с монолитной архитектурой, что приводит к миграции существующих монолитных архитектур в архитектуры на основе микросервисов.

Ключевыми преимуществами микросервисов являются отказоустойчивость и ремонтопригодность. Также разделение системы на независимые и самостоятельно развертываемые службы помогает командам разработчиков тестировать свои службы и вносить изменения независимо от других разработчиков, что упрощает распределенную разработку. Небольшой размер микросервисов способствует тому, что код становится более понятным для разработчиков [2].

Еще одно ключевое преимущество микросервисов – масштабируемость. В микросервисной архитектуре каждый микросервис может быть развернут на разных машинах, каждый с разным уровнем производительности, и может быть написан на более подходящем языке программирования. Если, например, существует узкое место в конкретном микросервисе, его можно поместить в контейнер и запустить на нескольких параллельно работающих хостах, без необходимости развертывания системы на новой мощной машине.

Следующее существенное преимущество микросервисов – возможность их замены. Поскольку отдельные микросервисы имеют небольшой размер, их легко заменить на более совершенную реализацию или даже удалить. Команды разработчиков, использующие подходы с использованием микросервисов, не сталкиваются с большими трудностями с полным переписыванием микросервисов, когда это необходимо.

Важным преимуществом микросервисов является отказоустойчивость. Отказ микросервиса обычно не влияет на всю систему. Тем не менее, неисправные микросервисы также могут быть быстро перезапущены.

Еще одно ключевое преимущество микросервисов – надежность, так как данная технология разработки программного обеспечения подразумевает создание больших систем из простых и небольших компонентов с чистыми интерфейсами (микросервисами).

## 2. Стратегия миграции от монолита к микросервисам

В монолитной архитектуре система имеет определенное количество функций, которые используются для совершения определенных действий. Функции могут быть вызваны в упорядоченных и конечных последовательностях, эти последовательности вызо-

вов функций называются *бизнес-функциями*. Монолитная система может иметь ряд бизнес-функций в зависимости от ее характеристик и конструкции [3]. И именно бизнес-функции играют важную роль в такого рода миграциях.

Предлагаемая стратегия миграции монолитной архитектуры в архитектуру микросервисов сосредоточена вокруг концепции бизнес-функций и состоит из пяти фаз, которые показаны на рис. 1.


Рис. 1. Ф*азы перехода от монолита к микросервисам*

Поясним вышеуказанные фазы миграции.

1. Функциональный анализ. На первом этапе рассматриваемой стратегии необходимо выделить и проанализировать все функции, составляющие монолит. Функции анализируются для извлечения таких данных, как частота использования и размер (с точки зрения строк кода). Затем функции также могут быть изменены: если функция большая по размеру, ее можно разделить на несколько подфункций; если две или более функции очень похожи и невелики по своему размеру, их можно объединить в одну и т.д.
2. Определение бизнес-функций. На данном этапе рассматриваемой стратегии определяются бизнес-функции. Каждый монолит может иметь определенное количество бизнес-функций в зависимости от его характеристик и структуры. Идентификация всех бизнес-функций в монолите – сложная задача. Самый простой способ определить их – задать вопросы владельцу или разработчикам монолита. Другой способ – изучение документации рассматриваемой монолитной системы и сопоставление ее функционала с описанными бизнес-процессами.
3. Анализ бизнес-функций. Бизнес-функции, определенные на предыдущем шаге, анализируются с тем, чтобы извлечь такие данные, как их уровень использования и другую статистическую информацию с точки зрения приложения. Полученные данные впоследствии дополнительно анализируются.
4. Назначение бизнес-функций. На этом этапе бизнес-функции теоретически назначаются микросервисам. Благодаря статистической информации, полученной на предыдущем этапе, имеется реальная возможность улучшить качество, размер и степень детализации микросервисов. Следуя этой стратегии, если бизнес-функции используются часто по отношению к другим, то они назначаются одному микросервису; если вместо этого две или более функциональных возможности используются более или менее одинаковое количество раз и их объем аналогичен, все они встраиваются в один микросервис. Назначение бизнес-функций микросервисам – сложная задача, и в этом разделе описывается только возможная стратегия назначения.
5. Создание микросервисов. После теоретического присвоения бизнес-функций микросервисам, начинается их разработка и внедрение. Создаются микросервисы, внутри которых разрабатываются функции, связанные с бизнес-процессами, или же

бизнес-процессы целиком, если они имеют небольшой размер.

После окончания разработки и развертывания микросервисов с конкретным функционалом, необходимо в обязательном порядке подключить их к еще работающему монолиту, тем самым передав часть функционала вновь созданным микросервисам.

Таким образом, монолит будет какое-то время частично играть роль шины данных, постепенно передавая все функции создаваемым микросервисам, пока не сделает это полностью.

## Выводы

Благодаря вышеописанной стратегии, новая система может получить преимущества, связанные с микросервисной архитектурой. Компании, которым необходимо развить свои системы или добавлять к ним новые функции, смогут перенести свои существующие системы и реализовать в них новые функции (вместо того чтобы разрабатывать новую систему), тем самым экономя ресурсы.

Стратегия, описанная в этой статье, дает лишь общее представление о том, как реализуется миграция программного кода приложения из монолитной архитектуры в микросервисную архитектуру.

## Список литературы

1. *Никитин И.В., Гриценко Т.Ю.* Сравнение подходов монолитной архитектуры и микросервисной архитектуры при реализации серверной части веб-приложения // Дневник науки. 2020. № 3.
2. *Мартин Р.* Чистая архитектура. Искусство разработки программного обеспечения. СПБ.: Питер, 2018.
3. *Mazlami G., Cito J., Leitner P*. Extraction of microservices from monolithic software architectures // IEEE International Conference on Web Services (ICWS). 2017. P. 524–531.

# Software code migration strategy from monolithic architecture to microservices

**D. K. Radostev, E. Yu. Nikitina**
Perm State University; 15, Bukireva st., Perm, 614990, Russia
radostevd3@gmail.com; +79194741092

The purpose of this work is to describe a strategy that can help enterprises transition from a monolithic application code architecture to a microservice architecture. Using this migration strategy, the new system will receive a number of pre-assets offered by the microservice architecture, such as scalability and repairability. Companies will be able to migrate their older systems to more flexible systems, while increasing the performance of their software.

**Keywords:** *software code; application architecture; monolithic architecture; microservices*.