УДК 004.273

ПРОБЛЕМЫ ПЕРЕХОДА ОТ МОНОЛИТНОЙ К МИКРОСЕРВИСНОЙ АРХИТЕКТУРЕ

И.Ф.Киринович
доцент кафедры инженерной психологии и эргономики БГУИР, кандидат физико-математических наук, доцент

А.В.Аксенчик
профессор кафедры вычислительных методов и программирования БГУИР, доктор физико-математических наук, профессор

Белорусский государственный университет информатики и радиоэлектроники».
E-mail: Kirinovich.irina@yandex.ru.

И. Ф. Киринович
Окончила Белорусский государственный университет имени В. И. Ленина по специальности «математика». Работает на кафедре инженерной психологии и эргономики БГУИР. Отличник образования Республики Беларусь.

А. В. Аксенчик
По окончании Минского радиотехнического института и по настоящее время работает преподавателем в БГУИР.

Аннотация. Термин «Микросервисная Архитектура» появился относительно недавно, и описывает подход к разработке архитектуры приложения, представляющего собой отдельно разворачиваемые модули [9]. Сама идея предполагает гетерогенность и слабосвязанность компонентов архитектуры, основанная на принципах разделения обязанностей. В работе определены проблемы, с которыми сталкиваются организации при переходе с монолитной на микросервисную архитектуру, приведены пути ранней идентификации и предотвращения данных проблем.

Ключевые слова: монолитная архитектура, микросервисы, сервис-ориентированная архитектура.

Введение.

В состав архитектуры любой корпоративной системы входят взаимодействующие модули или отдельные подсистемы. При значительном количестве модулей интеграция между ними напрямую требует поддержки множества внешних интерфейсов. В результате появляются интеграционные платформы, которые закладываются в основу инфраструктуры корпоративных приложений. Первые интеграционные платформы создавались на основе центрального брокера сообщений. Позже их сменили продукты класса ESB [3], Некоторые из таких систем заменили центральный брокер сообщений на более гибкие структуры, развивая идеи сервис-ориентированного подхода [7], при котором несколько сервисов работают совместно для предоставления некоего конечного набора возможностей [2]. Под сервисом здесь понимается полностью отдельный процесс операционной системы. Связь между такими сервисами осуществляется через сетевые вызовы, а не через вызовы методов в границах процесса.

Системы, созданные на микросервисной архитектуре, легко масштабируются, используют разнородные источники данных, что повышает их эффективность. Микросервисы можно заменять, адаптировать под новые требования, что делает их гибкими.

Анализ проблем.

Проблемы, с которыми сталкиваются организации при построении архитектуры микросервисов, зачастую не проявляются на ранних этапах разработки системы.

В настоящее время многие аспекты работы сети, построенной по данным принципам, ещё

недостаточно исследованы. Не существует эталонных примеров микросервисной архитектуры.

К проблемам, с которыми сталкиваются организации, относятся:

1) Сложность координации между сервисами. В корпоративном приложении необходимо координировать запросы между сервисами, поддерживать сложную высокоуровневую логику [8]. В таких случаях предусматривают механизм координации микросервисов. Используются либо готовые платформы развёртывания бизнес-процессов, либо собственные разработки. В любом случае типичными шаблонами выступают композит и медиатор [4].

2) Сложность администрирования микросервисов. С течением времени количество микросервисов возрастает, что диктуется потребностями бизнеса. Также возможны зависимости на конкретные версии подключаемых модулей. Возрастает сложность управления развёртываем и конфигурацией как новых микросервисов, так и поддержки существующих. Поэтому в проекты вводятся практики непрерывной интеграции, которые предусматривают автоматизацию процессов, а также использования систем непрерывного и автоматизированного развёртывания [1], анализа качества кода и покрытия тестами.

3) Вопросы связывания данных [8]. Для связывания нескольких сущностей в монолитном приложении, как правило, используется понятие контекста исполнения, в рамках которого производятся транзакции, сохраняются или изменяются данные. В архитектуре микросервисов состояние исполнения единого бизнес процесса, как правило, не хранится в какой-либо единой сущности, передаваемой по сети. Для этого используются данные о корелляции, позволяющие определить, какие данные или события необходимо обработать. Продуманная событийная модель и механизм корелляции позволят избавиться от проблем, связанных с обработкой и фильтрацией большого количества лишней информации.

4) Проблемы поддержки конечного продукта. При использовании монолитной архитектуры весь исходный код основного приложения, как правило, расположен в едином репозитории, запускается в единой среде. При использовании микросервисов, каждый отдельный модуль может иметь свои особенности запуска и исполнения, архитектура отдельно взятых сервисов может сильно отличаться, что усложняет задачу анализа и дальнейшей работы над сервисом. Поэтому важно иметь полноценную документацию по всем компонентам, а также их связям и взаимодействиям. Для достижения единого стиля, архитекторы разрабатывают общие требования к микросервисам, которых должны придерживаться все разработчики.

Выводы.

Анализ архитектуры микросервисов, а также проблем при их использовании, с которыми в настоящее время сталкиваются организации, не позволяет однозначно выделить этот подход как базовый. Слабая поддержка результирующей инфраструктуры, недостаточная квалификация разработчиков может привести к получению сложной и ненадёжной системы. В настоящее время существует подход, когда команды прибегают к использованию монолитного подхода с дальнейшим выделением компонентов в отдельные микросервисы [6]. Таким образом, можно оценить разницу и планировать дальнейшее развитие системы.

Список литературы

[1] Paul M. Duvall Continious Integration. Improving Software Quality and Reducing Risk. – Boston: Pearson Education, 2007. – p. 227 - 272

[2] Newman S. Designing Fine-Grained Systems. – Sebastopol: O'Reilly, 2015. – p. 33-35

[3] David A. Chappell Enterprise Service Bus. – USA: O'Reilly, 2004. – 276 p.

[4] Freeman E. Head First Design Patterns. – USA: O'Reilly, 2004. – p. 320 – 377, 622 -624

[5] Fowler M. Continuous Integration [электронный ресурс] // MARTINFOWLER.COM : Software Development Articles. 2006 г. – Электрон. данные. URL: https://martinfowler.com/articles/continuousIntegration.html (дата обращения 20.01.2017 г.). – Заглавие с экрана.

[6] Fowler M. MonolithFirst [электронный ресурс] // MARTINFOWLER.COM : Software Development Articles. 2015 г. – Электрон. данные. URL: https://martinfowler.com/bliki/MonolithFirst.html (дата обращения

05.02.2017 г.). – Заглавие с экрана.

[7] Oracle Service Bus 12.1.3 [электронный ресурс] // DOCS.ORACLE.COM : Product documentation. 2015 г. – Электрон. данные. URL: https://docs.oracle.com/middleware/1213/osb/docs.htm (дата обращения 20.01.2017 г.). – Заглавие с экрана.

[8] Wähner K. Do Good Microservices Architectures Spell the Death of the Enterprise Service Bus? [электронный ресурс] // VOXXED.COM : Developer Works. 2016 г. – Электрон. данные. URL: https://www.voxxed.com/blog/2015/01/good-microservices-architectures-death-enterprise-service-bus-part-one/ (дата обращения 05.02.2017 г.). – Заглавие с экрана.

[9] Fowler M. Microservices – a definition of this new architectural term [электронный ресурс] // MARTINFOWLER. COM : Software Development Articles. 2014 г. – Электрон. данные. URL: http://www.martinfowler.com/articles/microservices.html (дата обращения 04.02.2017 г.). – Заглавие с экрана.

PROBLEMS OF TRANSITION FROM MONOLITHIC TO MICROSERVICE ARCHITECTURE

I.F KIRINOVICH.
Ph.D., assistant professor, associate professor of engineering psychology and ergonomics department of Belarusian state university of informatics and radioelectronics

A.V. AKSENCHIK
D.Sci., professor, professor of Computational Methods and Programming department of Belarusian state university of informatics and radioelectronics

Belarussian State University of Informatics and Radioelectronics
E-mail: Kirinovich.irina@yandex.ru

Abstract. The term "Microservice Architecture" has appeared relatively recently, and describes an approach to developing an application architecture that consists of separately deployable modules [11]. The idea itself assumes heterogeneity and loosely coupled components of the architecture, based on the principles of separation of responsibilities. The paper identifies the problems that organizations face when switching from a monolithic to a micro-service architecture, and provides ways to identify and prevent these problems early.

Keywords: monolithic architecture, microservices, service-oriented architecture.