ЭВОЛЮЦИЯ АРХИТЕКТУРНЫХ ПАТТЕРНОВ: КОМПОЗИЦИОННОЕ ПРОЕКТИРОВАНИЕ В МИКРОСЕРВИСНЫХ И БЕССЕРВЕРНЫХ СРЕДАХ

Ульданов А.М., бакалавр

Научный руководитель **Шарафутдинов А.Г.,** к.э.н., доцент

БашГАУ, Уфа, Россия

Аннотация. В статье рассматриваются принципы проектирования информационных систем в условиях перехода от монолитной архитектуры к распределенным вычислениям. Проведен анализ методов декомпозиции предметной области на основе Domain
-
DrivenDesign
. Предложен подход к обеспечению отказоустойчивости с использованием паттерна Saga
и событийно-ориентированного взаимодействия. Описаны требования к наблюдаемости в бессерверных средах.

Ключевые слова: микросервисная архитектура, Domain-DrivenDesign, событийно-ориентированное взаимодействие,
FaaS
, распределенная трассировка, контрактное программирование, проектирование ИС, отказоустойчивость.

Автор: Ульданов А.М.
05.06.2026 10:57 -

Современный этап развития индустрии программирования характеризуется фундаментальным сдвигом в представлении о единице развертывания программного обеспечения. Если на протяжении десятилетий стандартом являлось монолитное приложение, инкапсулирующее всю бизнес-логику в рамках одного процесса, то сегодня доминирующей парадигмой становится композиционная сборка слабосвязанных сервисов.

Главным стимулом отказа от монолитов является не технологическая мода, а объективная потребность бизнеса в независимом масштабировании функциональных модулей и сокращении времени вывода новых версий продукта на рынок. Однако механическое разделение системы на отдельные компоненты без должного методологического обоснования приводит к росту энтропии связей, проблемам с согласованностью данных и усложнению эксплуатации.

Ключевой методологической основой грамотной декомпозиции выступает предметно-ориентированное проектирование (Domain-DrivenDesign, DDD). Задача архитектора заключается не в техническом разрезании базы данных, а в выделении ограниченных контекстов (
BoundedContexts
), границы которых определяются языком и правилами конкретной бизнес-функции. Критерием качества декомпозиции служит минимизация числа синхронных вызовов между контекстами.

Автор: Ульданов А.М.
05.06.2026 10:57 -

При переходе к распределенной архитектуре радикально меняется модель взаимодействия компонентов. Прямые синхронные вызовы по протоколу HTTP/REST создают жесткую связанность (
coupling
), при которой простой одного сервиса вызывает каскадные отказы всей системы. Альтернативой является событийно-ориентированная архитектура (
Event
-
DrivenArchitecture
), где взаимодействие осуществляется через асинхронную публикацию событий в брокере сообщений (
ApacheKafka
,
RabbitMQ
). Сервисы-подписчики реагируют на изменения состояния независимо, что обеспечивает пространственную и временную развязку.

Особое внимание при проектировании следует уделять обработке транзакций в распределенной среде. Поскольку классическая модель ACID в контексте нескольких независимых баз данных недостижима, приходится применять паттерн
Saga
. Он подразумевает разбиение глобальной бизнес-транзакции на последовательность локальных шагов, для каждого из которых разработчик обязан предусмотреть компенсирующее действие на случай сбоя. Это требует принципиально иного мышления: архитектор проектирует не только успешный сценарий, но и сложную логику отката с гарантией идемпотентности операций.

Дальнейшим витком эволюции проектирования выступает бессерверная парадигма (Fun ctionasaService
,
FaaS
). Такие платформы, как

AWSLambda
или
YandexCloudFunctions
, абстрагируют от разработчика управление средой выполнения и пулом потоков. Единицей дизайна здесь выступает атомарная функция, время жизни которой измеряется миллисекундами. Проектирование в
FaaS
требует максимальной чистоты и отсутствия состояния (
statelessness
), что накладывает жесткие ограничения на выбор паттернов хранения данных и ведет к популяризации внешних
state
-машин для оркестрации.

Неотъемлемой частью современного проекта информационной системы становится наблюдаемость (Observability). В условиях, когда один пользовательский запрос обслуживается десятками микросервисов и функций, классический дебаггинг становится невозможен. Необходимо на этапе проектирования закладывать стандарты распределенной трассировки (DistributedTracing). Каждый входящий запрос должен снабжаться уникальным идентификатором (
TraceID
), который передается по всей цепочке вызовов. Без внедрения единого формата логирования и сбора метрик (
Prometheus
,
Grafana
) эксплуатация микросервисной системы становится хаотичной.

Влияние технологий искусственного интеллекта на проектирование усиливается. Системы-ассистенты программиста (GitHubCopilot, GPT-подобные модели) берут на себя генерацию шаблонного кода, смещая фокус архитектора с реализации на спецификацию интерфейсов. Растет значимость декларативных языков и схем (

OpenAPI
,
AsyncAPI
,
Protobuf
), которые становятся не просто документацией, а исходным артефактом, управляющим кодогенерацией.

Таким образом, проектирование современных информационных систем трансформируется в инженерную дисциплину управления сложностью через формальные контракты. Мастерство архитектора заключается не в знании синтаксиса конкретного языка, а в умении разбивать систему на устойчивые к изменениям компоненты, обеспечивать асинхронное взаимодействие и предотвращать деградацию качества данных в распределенной среде.

Литература

1. Афанасьев, А. Н. Методология разработки распределенных интеллектуальных систем проектной деятельности / А. Н. Афанасьев, Н. Н. Войт // Труды Международного конгресса по интеллектуальным системам и информационным технологиям «IS&IT'12». – М. : Физматлит, 2012. – Т. 1. – С. 341–347.

2. Шитько, А. М. Проектирование микросервисной архитектуры программного обеспечения / А. М. Шитько // Труды БГТУ. Серия 3: Физико-математические науки и информатика. – 2017. – № 2 (200). – С. 98–101.

Автор: Ульданов А.М.
05.06.2026 10:57 -