ARCHITECTURE DECISION RECORDS КАК ИНСТРУМЕНТ УПРАВЛЕНИЯ АРХИТЕКТУРНЫМИ ЗНАНИЯМИ

Гурин Алексей Вячеславович,
старший инженер-разработчик мобильных приложений, TUTU, мобильная разработка для платформы iOS, г. Самара

E-mail: gurin.fear@gmail.com

Михайлов Богдан Александрович,
Lead Software Engineer, Компания VK, г. Москва

E-mail: bogdan.mihailov@internet.ru

Мозолевский Дмитрий Иванович,
инженер-программист полного цикла, США, г. сан Франциско

E-mail: dmytro.mozolevskyi@gmail.com

Аннотация. Данная статья посвящена исследованию использования Architecture Decision Records (ADR) как инструмента управления архитектурными знаниями в процессе разработки программного обеспечения. В статье рассматриваются основные преимущества внедрения ADR, а также его влияние на гибкость архитектуры, сокращение времени принятия решений и улучшение документации. Результаты исследования подтверждают, что использование ADR ускоряет процесс принятия решений и повышает гибкость архитектуры, что делает этот инструмент важным элементом успешных Agile-практик. В статье также представлены практические рекомендации для внедрения ADR в команды разработки и обозначены возможности для дальнейших исследований.

Ключевые слова: Architecture Decision Records, управление архитектурными знаниями, Agile, принятие решений, гибкость архитектуры, документация, разработка программного обеспечения, технический долг, управление проектами.

Актуальность исследования

Современные методы разработки программного обеспечения требуют высокой гибкости и скорости принятия решений, особенно когда речь идет о

проектировании архитектуры систем. В условиях быстро меняющихся требований и ограничений, командная работа и эффективное управление архитектурными знаниями становятся ключевыми для достижения успеха. Одним из инструментов, который может значительно улучшить этот процесс, является Architecture Decision Records (ADR) – методика, направленная на документирование архитектурных решений и их контекста.

Внедрение ADR помогает командам избежать дублирования ошибок и недоразумений, снижает зависимость от устных соглашений и способствует лучшему пониманию и поддержанию принятых решений в долгосрочной перспективе. Несмотря на все обещания ADR, до сих пор существует недостаточно эмпирических данных, которые бы подтверждали влияние этой практики на скорость и качество принятия архитектурных решений в реальных условиях разработки программного обеспечения.

Актуальность исследования заключается в необходимости выявления и анализа конкретных эффектов использования ADR в реальных проектах разработки ПО, а также оценки того, как этот инструмент способствует улучшению качества и скорости принятия решений.

Цель исследования

Целью данного исследования является эмпирическая оценка влияния Architecture Decision Records на скорость и качество принятия архитектурных решений в процессе разработки программного обеспечения.

Материалы и методы исследования

Материалы исследования: документация по использованию ADR в командах разработки, научные статьи, книги, касающиеся архитектурных решений, использования ADR и Agile-практик в разработке программного обеспечения.

Методы исследования: теоретический анализ, статистический анализ, сравнительный анализ.

Результаты исследования

Architecture Decision Records (ADR) – это методика, предлагающая структурированный подход к документированию архитектурных решений, включая описание контекста, альтернатив и причин выбора той или иной стратегии. ADR помогает зафиксировать ключевые решения и их обоснования, что способствует лучшему пониманию и поддержке архитектуры системы в долгосрочной перспективе.

Ключевые компоненты ADR:

– Название решения – короткое, но информативное название, которое позволяет легко идентифицировать суть решения.

– Дата – дата, когда решение было принято.

– Статус – текущий статус решения (например, «принято», «отменено», «неактуально»).

– Контекст – описание ситуации, которая привела к необходимости принятия решения.

– Решение – описание самого принятого решения.

– Мотивировка – обоснование, почему было принято именно это решение, а не альтернативы.

– Альтернативы – краткое описание альтернативных решений, которые были рассмотрены, и причины, по которым они были отклонены.

– Последствия – потенциальные последствия принятия данного решения для проекта и команды.

Данный подход к документированию позволяет команде не только фиксировать решение, но и анализировать его последствия в будущем, обеспечивая таким образом поддержание архитектурных знаний на протяжении всего жизненного цикла проекта.

Архитектурные знания – это совокупность всех концепций, принципов, паттернов и решений, которые используются в процессе проектирования системы. В условиях современной разработки ПО управление архитектурными знаниями становится важнейшей составляющей успешной реализации проекта, поскольку архитектурные решения оказывают влияние на устойчивость, масштабируемость и сопровождаемость системы на протяжении ее жизненного цикла.

ADR является эффективным инструментом управления архитектурными знаниями, поскольку он позволяет:

1) Систематизировать архитектурные решения и предоставить их в удобной для анализа и повторного использования форме.

2) Обеспечить преемственность знаний, что критически важно в условиях динамичных команд и частых изменений.

3) Упростить процесс передачи знаний между различными группами разработчиков, особенно в больших распределенных командах.

4) Позволяет снижать риски из-за утраты знаний или забытых решений, что особенно актуально в крупных и долгосрочных проектах.

Несмотря на очевидные преимущества, внедрение ADR может сталкиваться с рядом проблем (таблица 1).

Таблица 1

Проблемы внедрения ADR и пути их решения

Проблема	Описание	Решение
Сопротивление изменениям	Негативное восприятие нововведений в командах	Обучение и демонстрация реальных выгод от использования ADR
Поддержание актуальности	Трудности с регулярным обновлением документации	Внедрение процессов, обеспечивающих регулярное обновление ADR
Перегрузка документацией	Избыточные записи могут затруднить поиск информации	Установление стандартов и практик написания кратких и лаконичных ADR

ADR помогает интегрировать процесс принятия решений в общий рабочий процесс команды. Это достигается через систематическое документирование решений и их контекста, что позволяет команде возвращаться к предыдущим записям, если необходимо, и на основе этого принимать новые решения. Такой подход позволяет избегать дублирования усилий и уменьшает риски, связанные с принятием решений, основанных на ограниченной информации или неясных условиях.

Запись принятых решений с анализом альтернатив позволяет отслеживать эволюцию архитектурных решений и лучше понять, как и почему были сделаны те или иные выборы. Это помогает команде осознавать, что определенные архитектурные решения могут быть устаревшими и требуют пересмотра, а также выявлять удачные практики, которые могут быть повторно использованы в других частях проекта.

Влияние интеграции ADR на скорость принятия архитектурных решений показано на рисунке 1.

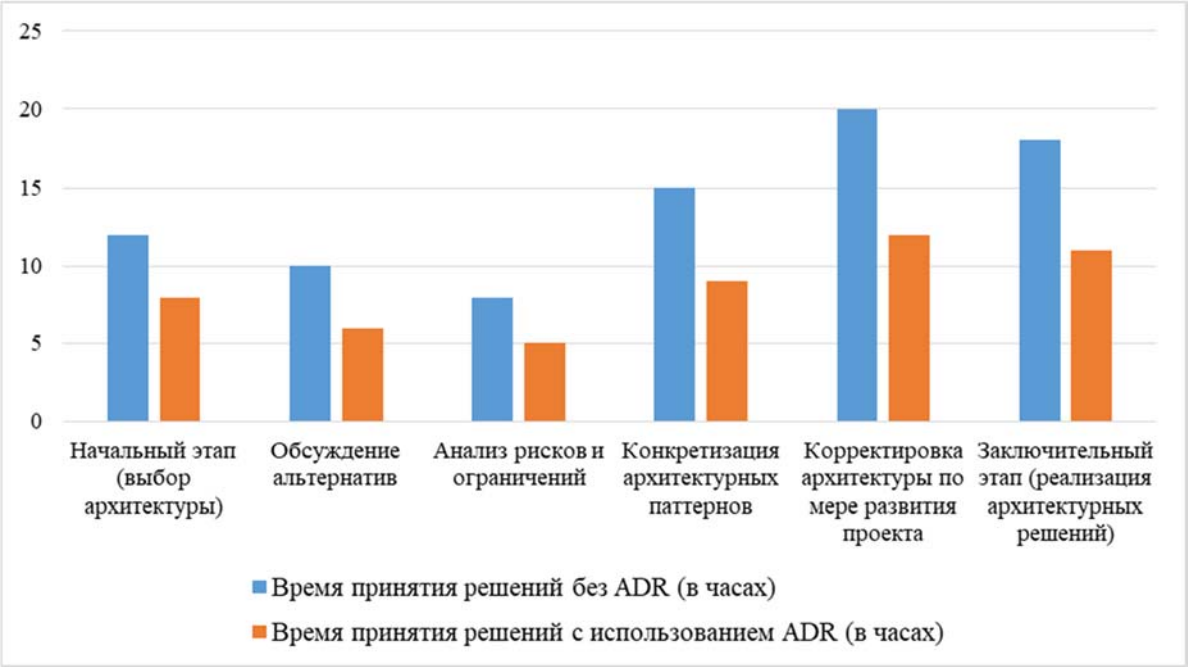

Рис. 1 Влияние интеграции ADR на скорость принятия архитектурных решений

Технический долг – это концепция, обозначающая необоснованные упрощения и решения, принятые для ускорения разработки, которые могут привести к проблемам в будущем. В контексте архитектуры программного обеспечения технический долг часто возникает из-за непродуманных архитектурных решений, которые, хотя и решают текущие задачи, создают сложности для дальнейшей разработки и масштабирования системы.

С помощью ADR можно фиксировать не только окончательное архитектурное решение, но и осознание того, что принятое решение является компромиссом или временной мерой. Например, в процессе проектирования системы могут быть выбраны менее оптимальные решения из-за нехватки времени или ресурсов. В этом случае ADR фиксирует, что выбранное решение является временным, и оно подлежит пересмотру в будущем [4, с. 7].

Документирование таких решений создает базу для более осознанного подхода к улучшению архитектуры системы в будущем. Это важно, поскольку, без этой документации, команды могут забыть о техническом долге или не осознавать его накопление.

Снижение архитектурного долга – это не только задача архитекторов, но и всех участников разработки, включая тестировщиков, менеджеров и аналитиков. Использование ADR помогает команде всегда иметь представление о том, где и почему могут возникнуть проблемы в архитектуре, а также снижает вероятность повторения тех же ошибок в будущем. При этом, если решение не работает в процессе разработки, оно может быть пересмотрено и заменено более эффективным.

ADR также тесно интегрируется с методологиями Agile и DevOps, которые нацелены на гибкость, скорость и итеративность разработки. В этих методологиях архитектурные решения часто принимаются в процессе разработки, а не на старте проекта, и могут меняться по мере изменения требований [3, с. 87].

Методология Agile фокусируется на частых поставках и изменениях в краткосрочной перспективе, что делает архитектурные решения динамичными. В таком контексте ADR становится инструментом для документирования и анализа этих изменений. Применение ADR в Agile позволяет сохранять гибкость, не теряя осведомленности о принятых решениях, их контексте и возможных последствиях. В командах, работающих по принципам Agile, ADR помогает поддерживать баланс между скоростью разработки и необходимостью обеспечения качества архитектурных решений [2, с. 841].

Влияние ADR на гибкость архитектуры в Agile представлено на рисунке 2.

В контексте DevOps ADR помогает систематизировать принятие архитектурных решений, которые могут напрямую влиять на процессы непрерывной интеграции и доставки. Одной из ключевых особенностей DevOps является необходимость быстрой адаптации системы к изменяющимся условиям эксплуатации, а также высокая степень автоматизации разработки и тестирования.

ADR помогает командам разработчиков, тестировщиков и специалистов по DevOps быть на одной волне, так как все архитектурные изменения фиксируются и становятся доступными для всех участников процесса. Это позволяет не только ускорить процессы интеграции, но и избежать ошибок, связанных с непониманием контекста архитектурных решений [1, с. 25].

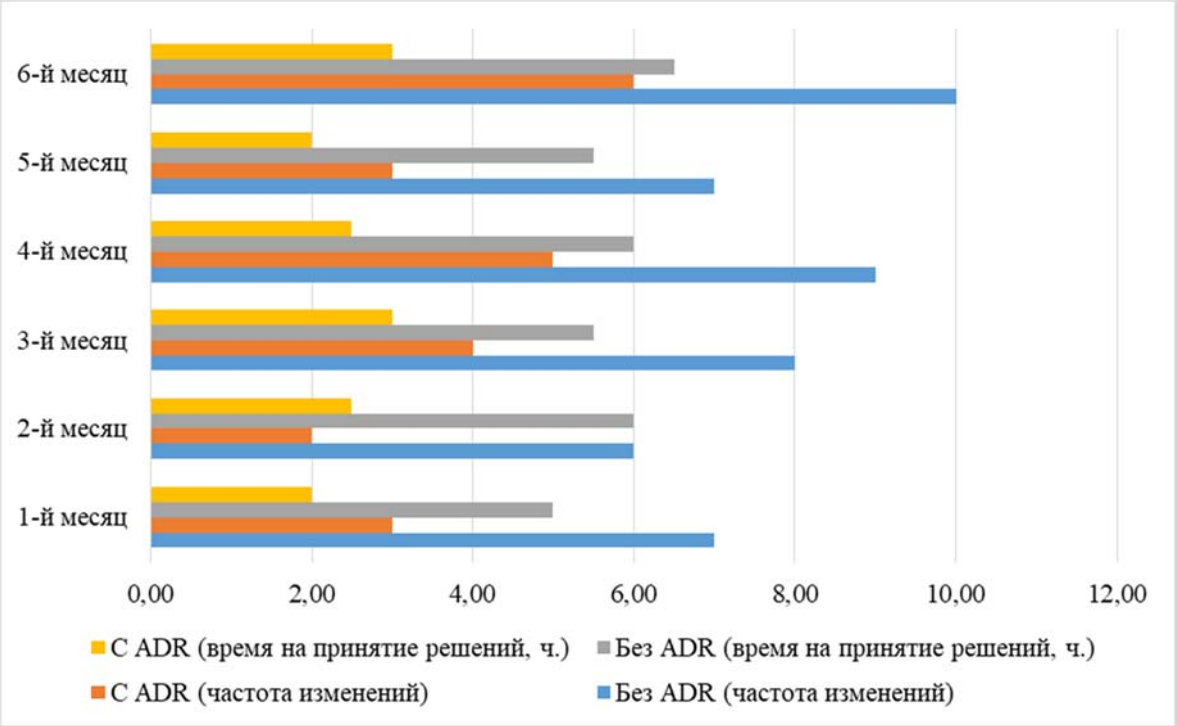

Рис. 2 Влияние ADR на гибкость архитектуры в Agile

Использование ADR также имеет психологический эффект на команду, поскольку повышает степень вовлеченности и уверенности в принятии решений (таблица 2).

Таблица 2

Психологические преимущества внедрения ADR

Преимущество	Описание
Повышение ответственности	Участники команды становятся более ответственными за архитектурные решения
Улучшение командной работы	Прозрачность решений способствует лучшему взаимопониманию и сотрудничеству
Снижение неопределенности	Документирование решений снижает уровень неопределенности и стресса среди участников проекта

Для внедрения Architecture Decision Records в команды разработки, важно предоставить практические рекомендации, которые помогут эффективно интегрировать этот инструмент в рабочий процесс. Ниже приведена таблица 3 с практическими рекомендациями, основанными на лучших практиках и опыте использования ADR в индустрии.

Ограничения текущего исследования предоставляют ясное понимание вызовов, с которыми сталкиваются команды при внедрении ADR.

Ограничения исследования:

1. Ограниченная выборка. Исследование основывается на данных, полученных из нескольких конкретных команд разработки, что может ограничить универсальность выводов.

2. Ограниченность в области применения. Рассмотрены только проекты, использующие методологии Agile, что исключает возможность обобщения результатов на другие подходы к разработке.

3. Фокус на технической стороне. Изучены преимущественно технические аспекты внедрения ADR, в то время как социальные и организационные аспекты внедрения могут потребовать отдельного исследования.

4. Краткосрочные наблюдения. Данные, собранные в ходе исследования, ограничены временными рамками, что не позволяет полноценно оценить долгосрочные эффекты от внедрения ADR.

Для будущих исследований существует несколько направлений, которые могут расширить понимание влияния ADR на процесс разработки. Во-первых, можно провести более широкие исследования, включая разнообразные типы организаций и команд, чтобы определить, насколько универсальными являются полученные результаты и в каких контекстах ADR оказывают наибольшее влияние. Во-вторых, полезным будет проведение долгосрочных исследований, которые позволят анализировать долговременные эффекты внедрения ADR, включая их влияние на снижение технического долга, улучшение качества кода и другие важные метрики разработки.

В будущем можно исследовать влияние ADR на конкретные аспекты архитектуры, такие как управление отказами, безопасность и масштабируемость, а также оценить, насколько эффективно ADR помогают в принятии решений, связанных с инновационными и сложными технологиями, такими как микросервисная архитектура или serverless-программирование.

Наконец, необходимо также рассмотреть возможность интеграции ADR с другими практиками и инструментами управления проектами, такими как DevOps, CI/CD и автоматизация тестирования. Это позволит более полно оценить, как ADR может способствовать улучшению всего жизненного цикла разработки и повлиять на эффективность и качество программного обеспечения в целом.

Таблица 3

Практические рекомендации для внедрения ADR в команды разработки

Рекомендация	Описание	Пример из практики
Стандартизация формата ADR	Важно создать и использовать единый формат для всех ADR, чтобы каждый документ был понятен и структурирован	Использование стандартных шаблонов ADR, таких как markdown шаблон
Обучение команды	Обучение участников команды тому, как правильно создавать и использовать ADR для документирования решений	Проведение внутренних семинаров по использованию ADR и обзору лучших практик
Принятие ADR на уровне всей команды	Убедитесь, что ADR является частью общего рабочего процесса команды, а не изолированным инструментом	Регулярные обзоры ADR на командных митингах или ретроспективах
Интеграция с системой контроля версий	Хранение ADR в системе контроля версий (например, Git) позволяет отслеживать изменения и поддерживать историю решений	Хранение всех ADR в репозитории Git, использование Pull Request для утверждения решений
Регулярные ревизии ADR	Периодическая проверка и обновление существующих ADR для обеспечения актуальности и соответствия текущим условиям	Внедрение практики ревизий ADR раз в квартал или при значительных изменениях в проекте
Оценка и документирование альтернатив	В каждом ADR должны быть указаны альтернативные решения и причины их отклонения, чтобы обеспечить понимание контекста	Включение раздела «Альтернативы и их анализ» в каждый документ ADR
Применение ADR на всех уровнях	Не ограничивайтесь только высокоуровневыми архитектурными решениями – используйте ADR для всех значимых решений, включая решения по технологиям, библиотекам, фреймворкам и т.д.	Введение практики использования ADR для всех решений, даже для мелких изменений, таких как выбор библиотек.
Интеграция с процессом код-ревью	Включение проверки ADR в процесс код-ревью для улучшения качества решений и обеспечения согласованности с архитектурой	Включение проверки ADR как обязательной части процесса код-ревью, особенно для крупных изменений
Использование ADR для управления техническим долгом	Включайте ADR в процесс управления техническим долгом для фиксации решений, которые могут быть изменены в будущем	Документирование временных решений с пометкой «подлежит пересмотру» в будущем для минимизации технического долга
Использование инструментов для автоматизации	Применение специализированных инструментов для генерации и хранения ADR (например, Structurizr, Docusaurus)	Использование автоматизированных инструментов для создания и отображения ADR

Выводы

Таким образом, внедрение Architecture Decision Records влияет на эффективность принятия архитектурных решений в условиях Agile-разработки. Основными положительными эффектами являются сокращение времени на принятие решений, повышение гибкости архитектуры и улучшение документации. ADR способствуют улучшению коммуникации внутри команды и повышению прозрачности процесса принятия решений. Несмотря на определенные ограничения, такие как субъективность данных и ограниченность времени наблюдения, результаты исследования подтверждают значимость ADR как инструмента для повышения качества разработки программного обеспечения. Внедрение ADR помогает командам быстрее адаптироваться к изменениям, снизить риски и эффективно управлять архитектурными знаниями. Будущие исследования могут расширить это исследование, анализируя долгосрочные эффекты применения ADR в различных контекстах и для других аспектов разработки программного обеспечения.

Литература:

1. Байдыбеков А.А., Гильванов Р.Г., Молодкин И.А. Современные фреймворки для разработки web-приложений // Интеллектуальные технологии на транспорте. – 2020. – № 4(24). – С. 23-29.

2. Локтионов Д.А., Масловский В.П. Критерии применения Agile-методологии для управления проектом // Креативная экономика. – 2018. – Т. 12, № 6. – С. 839-854.

3. Мухамадеева Р.М., Брюхова Е.М. Современное состояние и проблемы разработки программного обеспечения с использованием гибких методологий (реферат) // Устойчивое инновационное развитие: проектирование и управление. – 2022. – Т. 18, № 1(54). – С. 86-102.

4. Рогачева Ю.В. Гибкие методологии разработки программного обеспечения // Молодой ученый. – 2021. – № 38(380). – С. 5-8.