Архитектура как код: первые 10 лет

Автор: Максим Смирнов
Архитектура ИТ-решений
Telegram: @it_arch

Введение

Архитектура как код (Architecture as Code, AaC) — это подход, при котором архитектура представлена в виде декларативного машиночитаемого описания, развиваемого, обновляемого и отслеживаемого с использованием инструментов непрерывной интеграции и систем контроля версий. Эта практика обеспечивает согласованность между проектированием и реализацией, непрерывную эволюцию архитектуры, создание актуальных архитектурных артефактов, активное участие разработчиков и заинтересованных сторон.

Спустя более чем тридцать пять лет исследований и развития в области архитектуры программного обеспечения, несколько фундаментальных проблем остаются нерешенными. Несмотря на важность наличия четко определенного архитектурного описания, согласованного с системой, несоответствия и неправильное выравнивание все еще распространены. Хотя существует множество языков для описания архитектур, ни один не получил широкого распространения или признания в качестве де-факто стандарта. Особый интерес к AaC вызван развитием LLM (Large Language Models).

История возникновения концепции

Первое официальное упоминание

Первое официальное использование термина "Архитектура как код" можно найти на конференции **SATURN 2015** года, проводимой Институтом инженерии программного обеспечения (SEI) Университета Карнеги-Меллона. Саймон Браун, всем известный автор C4 модели, выступил с докладом "Software Architecture as Code" и, собственно, это было первое широкое упоминание термина архитектура как код.

На конференции Браун не дал каких-то четких определений концепции, сосредотачиваясь в основном на рассказе о C4 модели, которую он в течение многих лет представляет на представительных конференциях. Это было примечательно, потому что даже тогда C4 модели существовали уже 10 лет, и Браун был известным спикером, пропагандирующим эту идею.

Отсутствие четких определений

Удивительно, что в течение всех этих десяти лет после первого упоминания на SATURN 2015 были очень мало четких определений этого термина. Более того, практически не было никаких обзоров или документов, которые объясняли бы, что такое архитектура как код понятным языком. Первый серьезный материал появился только в **2025 году** — это был обзор литературы от группы итальянских и шведских исследователей.

Эти исследователи впервые систематически попытались разобраться, что такое архитектура как код, какие бывают инструменты и сервисы, и дать определения этой концепции. Они собрали пять различных определений и попытались найти в них общие моменты, результатом чего стало следующее консенсусное определение.

Определение архитектуры как код

Согласно исследованию 2025 года, архитектура как код — это подход, при котором архитектура представлена в виде декларативного машиночитаемого описания, развиваемого, обновляемого и отслеживаемого с использованием инструментов непрерывной интеграции и систем версионного контроля.

Это определение важно, так как позволяет нам правильно понимать границы концепции и сравнивать её с другими концепциями "как код".

Что не так с описанием архитектуры

Основные вызовы

Исследование выявило три основных вызова, которые архитектура как код пытается решить:

1. Несоответствие архитектуры и реальной системы

Первая проблема — это несоответствие описания архитектуры и реальной архитектуры системы. Система и её архитектурное описание часто не соответствуют друг другу. Это знакомая проблема для большинства архитекторов, и она встречается постоянно. Несоответствия и неправильное выравнивание остаются распространенными, и над 70% проблем несоответствия между описаниями архитектуры и исходным кодом возникают из-за недостатков в документации.

2. Отсутствие стандартного языка описания

Вторым вызовом является то, что ни один из языков описания архитектуры (Architecture Description Language, ADL) не получил широкого распространения, несмотря на то, что таких языков существует достаточно много. За время существования этой концепции было создано множество языков, но почему-то они не приживаются в практике.

3. Невозможность выявления архитектурно значимых изменений

Третий вызов, пожалуй, самый удивительный. Мы не можем четко понять, какое изменение какого аспекта является архитектурно значимым, а какое нет. Нет универсально принятых методологий для выявления значимых архитектурных аспектов. Это большая проблема для выявления архитектурных изменений и различения между архитектурными и не архитектурными вещами.

Сравнение с Infrastructure as Code

Определение IaC

Для лучшего понимания архитектуры как код полезно сравнить её с **Инфраструктурой как код** (Infrastructure as Code, IaC).

Инфраструктура как код — это процесс управления (managing and provisioning) ресурсами центров обработки данных с помощью машиночитаемых файлов-определений, а не конфигурированием оборудования посредством интерактивных инструментов или непосредственно.

Файлы-определения могут находиться в системе контроля версий, а не поддерживаться вручную. Для управления конфигурациями могут использоваться скрипты или декларативные определения, но IaC чаще использует декларативные подходы.

Ключевое различие

Главное отличие между IaC и AaC заключается в том, что инфраструктура как код предоставляет ценность через управление оборудованием из приложений посредством API. Архитектура же как код не предоставляет таких возможностей управления из приложений. Нет "коробочки архитектуры", из которой торчали бы API, позволяющие вызывая эти API загружать конфигурации и изменять архитектуру системы.

Архитектура как код просто означает формализованное описание архитектуры системы, что является принципиально другой вещью.

Ожидания от Architecture as Code

От концепции архитектуры как код существует множество ожиданий:

1. Интеграция проектирования в разработку

Первое ожидание состоит в том, что архитектура как код позволит нам включить процесс проектирования в стандартный процесс разработки и развертывания, в наши пайплайны CI/CD. Однако это не совсем правильная история, так как архитектура обязательно нужна, когда соответствие между описанием системы и её реальным состоянием актуально.

2. Автоматическое создание архитектурных диаграмм

Второе ожидание — это автоматизированное создание архитектурных диаграмм. Есть два варианта реализации этого подхода: анализ кода для автоматического создания архитектурного описания или использование комментариев в коде (документирование в коде) для создания диаграмм при сборке системы. Это может включать либо явное описание архитектуры в исходном коде (Diagrams as Code, DaC), либо автоматический анализ структуры кода.

3. Контроль соответствия

Третье ожидание — контроль соответствия кода архитектуре. Это вполне понятное и важное ожидание, хотя вопрос о том, как это соответствие будет автоматически обнаружено, остается открытым.

4. Валидация архитектуры

Четвертое ожидание состоит в том, что если мы описываем архитектуру в виде декларативных машиночитаемых описаний, то мы сможем автоматизировать процесс проверки и валидации архитектуры, а также все связанные с архитектурой операции.

5. Автоматическое создание представлений

Пятое ожидание — это так называемый "Killer Application". Это когда софт из архитектурных моделей, преимущественно из архитектурного репозитория, может автоматически рисовать те или иные диаграммы. Это не очень сложная задача, и она практически решена в инструментах диаграмм как код.

6. AI-ассистенты для архитектуры

Шестое ожидание связано с применением искусственного интеллекта. Если мы хотим использовать AI для помощи архитекторам, нам нужно формализовать, что такое архитектура, хотя это может получиться не очень хорошо.

История развития подходов

До появления концепции архитектуры как код существовало множество подходов и практик:

Формальные языки описания архитектур (ADL)

Существовали формальные языки описания архитектур (Architecture Description Language), которые существуют достаточно давно. Главная проблема этих языков в том, что они не получили широкого распространения, несмотря на их технические возможности.

Системная инженерия на основе моделей (MBSE)

Существовала и существует концепция системной инженерии, базирующейся на моделях (MBSE). Это отдельное направление, которое нормально развивается и предлагает множество полезных подходов для формализации инженерных процессов.

Записи архитектурных решений (ADR)

Где-то в районе 2010-2011 годов появились записи архитектурных решений (Architecture Decision Records, ADR) после того, как Майкл Нейг написал соответствующий блог-пост. Однако архитектурные решения существовали и раньше, они просто находились в менее формализованном виде. ADR предоставляют структурированный способ документирования важных решений.

Стандарты обмена

Существовали также стандарты форматов файлов для обмена архитектурными описаниями, например, ArchiMate Model Exchange File Format. Это позволяет различным инструментам обмениваться архитектурной информацией. Соответствующие стандарты описаны в международных документах, таких как ГОСТ Р 57100-2016/ISO/IEC/IEEE 42010:2011 "Система и программная инженерия. Описание архитектуры".

Ограничения и проблемы архитектуры как код

Существуют серьезные ограничения и проблемы концепции архитектуры как код:

1. Разнообразие архитектуры

Главная проблема в том, что архитектура описывает разные аспекты системы, с разных точек зрения, в разные моменты времени и с разными целями. Используется понятие "архитектурный ландшафт" — совокупность всех возможных описаний архитектуры, сделанных разными командами, в разное время, с разными целями, используя разные нотации.

Важно различать дескриптивную архитектуру (описание реального положения дел в данный момент времени) и перспективную архитектуру (ожидаемое будущее состояние). Дескриптивные архитектуры описывать проще, а вот с перспективными куда сложнее.

2. Отсутствие управляющих API

До появления IaC инфраструктура предоставляла API для конфигурирования оборудования. У приложений и архитектур нет таких API, что ограничивает возможности автоматизации. Распространению Infrastructure as Code предшествовало появление облачных API для управления ресурсами.

3. Синхронизация кода и архитектуры

Синхронизация кода приложений и архитектурных описаний остается непростой задачей. Архитектура описывает не только код, но и данные, окружение и другие аспекты системы. Поддержание актуальности архитектурной документации — это постоянный вызов.

4. Проблемы масштабирования

Инструменты архитектуры как код очень плохо работают с большими моделями. Даже простые инструменты типа IcePanel начинают разваливаться на полутора десятках элементов. Масштабируемость остается открытой проблемой для большинства доступных решений.

5. Мотивационная часть архитектуры

Очень сложно вписать в архитектуру как код мотивационную часть — ответы на вопрос "Зачем?" Как правильно описать в AaC записи архитектурных решений (ADR) или элементы мотивационных моделей остается неясным. Инструменты часто сосредотачиваются только на структурных аспектах.

6. Ориентация на изменения

Современная архитектура предприятия ориентирована на выявление и планирование изменений, что требует моделей реализации и миграции. Архитектура как код немного полезна для этих целей. Процессы трансформации требуют дополнительных описаний и метаданных.

7. Совместная работа

Почти все инструменты архитектуры как код очень плохо приспособлены для совместной работы. Они не поддерживают распределение описания на отдельные файлы с перекрестными ссылками, как это происходит в системах контроля версий для кода. Отсутствие поддержки merge-операций для моделей — значительное ограничение.

Диаграммы как код — решение, которое уже существует

Определение и концепция

Пока все мечтали про полноценную архитектуру как код, концепция **диаграмм как код** (Diagram as Code, DaC) уже реализована и работает.

Диаграммы как код — это методология, при которой диаграммы программных систем, архитектур или инфраструктурных компонентов генерируются программами, с использованием декларативного или императивного кода, а не создаются вручную в графических редакторах.

Преимущества диаграмм как код

Диаграммы как код успешно решает большинство задач, которые архитектура как код пытается решить, и использует мощь версионирования, CI/CD инструментов и других подходов разработки программного обеспечения:

- **Использование репозиториев ПО и инструментов CI/CD** — диаграммы хранятся в системах контроля версий наравне с кодом
- **Версионирование и автоматическое обновление при изменениях** — история всех изменений сохраняется, возможен откат
- **Формирование диаграмм по запросу, с требуемым уровнем детализации** — можно генерировать различные представления

- **Возможность использования разных нотаций и инструментов визуализации** — гибкость в выборе формата
- **Автоматическая проверка ограничений и валидация диаграмм** — можно добавить проверки в CI/CD
- **Поддержка сложных и динамических диаграмм** — возможность описания динамического поведения

Современные инструменты для диаграмм как код

Существует множество современных инструментов для работы с диаграммами как код:

LIKEC4

LIKEC4 — это современный инструмент, который позволяет описывать C4 модели в декларативной форме и анимировать последовательности событий. Он предоставляет playground для экспериментов с архитектурными моделями.

```
model {
  customer = actor 'Customer' {
    description 'Customer of Cloud System'
  }

  cloud = system 'Cloud System' {
    backend = component 'Backend' {
      description 'Backend services and API'
      auth = component 'Authentication' {
        description 'Self-Hosted Authentication'
      }
      api = component 'Backend API' {
        technology 'java/spring'
        description 'Java Spring Web-service'
      }
    }
  }
}
```

Structurizr

Structurizr — инструмент для создания архитектурных диаграмм, предоставляющий облачное хранилище и поддержку различных типов диаграмм.

IcePanel

IcePanel — еще один вариант инструмента для создания архитектурных диаграмм с поддержкой совместной работы. Он предоставляет интерфейс для совместного редактирования архитектурных моделей несколькими пользователями.

Excalidraw

Excalidraw — это простой веб-инструмент для создания диаграмм с поддержкой совместного редактирования и экспорта в различные форматы.

Eraser

Eraser предоставляет AI-поддержку для создания диаграмм на основе текстовых описаний, с возможностью автоматического создания диаграмм различных типов (последовательности, сущности, облачные архитектуры).

Отношение между архитектурой и диаграммами

Понимание диаграмм как проекций

Важно понимать, что любая архитектурная диаграмма — это не целое, не полное понимание, это не архитектура, это некая проекция архитектуры на ту или иную плоскость:

- **Плоскость данных** — описание потоков данных и хранилищ
- **Плоскость активной структуры** (структура компонентов) — статическая структура системы
- **Плоскость поведения** — динамическое взаимодействие компонентов
- **Плоскость развертывания и другие** — как система развертывается и работает в окружении

Каждая диаграмма — это частный взгляд на систему. Существуют другие альтернативные, может быть, даже противоречивые представления того же самого, и их тоже иногда полезно принимать во внимание. Главное — понимать, что этот частный взгляд действительно частный.

Результаты опросов о целях Architecture as Code

Исследование предпочтений архитекторов показало разнообразные мнения о том, как лучше всего представлять архитектуру:

По типам диаграмм

- C4 Container: 36% (574)
- Context: 18% (291)
- BPMN: 18% (294)
- Entity Relationship: 11% (186)
- Flow: 13% (216)
- Deployment: 10% (158)

По формам представления

- Свободный формат: 37% (594)
- Таблицы/текст: 17% (282)

Эти результаты показывают, что нет единого мнения о "правильном" способе представления архитектуры, что подтверждает сложность стандартизации в этой области.

Все остальное как код

Архитектура как код — это лишь один из примеров более общей парадигмы "Everything as Code". Существует целая экосистема различных подходов:

Инфраструктура и конфигурация

- Infrastructure as Code
- Configuration as Code
- Network as Code
- Storage as Code
- Orchestration as Code
- Platform as Code

- Environments as Code

Приложение и разработка

- Architecture as Code
- Diagrams as Code
- Docs as Code
- Data as Code
- Database as Code
- Pipeline as Code
- Project as Code

Мониторинг и анализ

- Monitoring as Code
- Dashboards as Code
- Analytics as Code
- Detection as Code

Безопасность и соответствие

- Security as Code
- Policy as Code
- IAM as Code
- Compliance as Code
- SLO as Code
- Privacy as Code

Эта таксономия показывает, как подход "как код" применяется ко всем аспектам современной ИТ-инфраструктуры и разработки.

Проблемы внедрения архитектуры как код в предприятии

Локальный подход к внедрению

Если говорить об внедрении архитектуры как код на уровне предприятия, нужно помнить несколько важных моментов:

Не надо прививать архитектуру на уровень всего предприятия. Нужно поделить предприятие на команды. Если мы работаем с десятками команд, совсем необязательно, что все команды занимаются архитектурой. Пусть будут хорошие команды, которые занимаются архитектурой, и команды, которые не занимаются ею. Всегда будет возможность выбора, кому ту или иную задачку отдать.

То же самое касается внедрения подходов Enterprise Architecture. Нужно выбирать стейкхолдеров, которые могут обещать бизнес-результаты в случае архитектурных изменений, и с ними работать.

Контроль соответствия и обратная связь

Для того чтобы архитектура как код действительно работала, нужны контроль и точки обратной связи. Если бы существовал Control Loop (контрольный цикл), который по архитектуре как код проверял бы реальное положение дела, это было бы очень полезно. То же самое касается feedback points — точек обратной связи. Пока такие механизмы не реализованы в полном объеме.

Использование AI для архитектуры

На горизонте 3-5 лет нас ждет развитие архитектурного AI — решения в стиле "поговори со своим архитектором", точно так же как существуют решения "поговори со своим юристом", "поговори со своим бизнес-аналитиком".

Архитекторам нужно будет проектировать эти решения и использовать их в промышленных масштабах, потому что очередь желающих из разных функциональных подразделений очень длинная.

Современные подходы используют GraphRAG и другие продвинутые методики поиска и обработки информации:

- **Similarity search** на векторных базах данных, содержащих текст
- **Graph search** на графах знаний
- **Retrieval Augmented Generation (RAG)** для получения контекстной информации
- **Fine-tuning** моделей для специфических задач

Версионирование архитектурных моделей при командной работе

При командной разработке вопрос версионирования архитектурной модели встает особенно остро. Ответ на вопрос "как версионировать" не является главным. Главный вопрос — "как поделить архитектурное описание на кусочки".

Нужно поделить архитектурное описание на кусочки не в соответствии с архитектурными доменами и не в соответствии с 4+1 представлениями, а каким-то другим способом. Основание для деления должно быть в том, чтобы архитектурное описание изменялось вместе с исходниками системы. Нужно сблизить гранулярность архитектурного описания и исходников.

Отрицательные примеры использования

Самая большая ошибка

Самый распространенный отрицательный кейс использования архитектуры как код состоит в том, что архитектор вместо того, чтобы проектировать решение, начинает программировать инструменты для архитектуры как код. Вместо того, чтобы придумывать конкретные реализации бизнес-инициатив, архитектор сидит и создает какой-нибудь новый инструмент для архитектуры. Это была большая проблема на протяжении всей карьеры архитекторов, начиная еще со времен работы в телекоме, где люди придумывали языки описания приложений и архитектур.

Важно помнить, что инструменты должны служить целям архитектора, а не наоборот.

Заключение

За десять лет развития концепции архитектуры как код мы увидели множество ожиданий, попыток внедрения и интересных решений. Однако полноценная реализация архитектуры как код все еще остается в стадии развития.

Вместе с тем, концепция диаграмм как код уже практически реализована и приносит реальную ценность. Это позволяет использовать мощь системы контроля версий, CI/CD инструментов и других практик разработки для управления архитектурными диаграммами.

Архитектура как код — это важная концепция, которая пытается формализовать и автоматизировать процессы описания архитектуры. Однако нужно понимать, что это не панацея, и её внедрение требует правильного подхода, выбора правильных инструментов и понимания ограничений этого подхода.

Главное, что нужно помнить — архитектура как код должна служить целям организации, а не становиться самоцелью. Архитектор должен проектировать реальные решения, а не тратить время на создание новых инструментов, пока это в разумных пределах и служит достижению бизнес-целей.

Вопросы и дальнейшее развитие темы остаются открытыми, и предстоящие годы принесут новые инсайты и решения в области архитектуры как код.
[1] [2]

⁂

1. Architecture-as-Code-Article.pdf
2. Architecture-as-Code-5-noiabria-2025.pdf