DOI 10.37882/2223-2966.2025.12.43

КОМПЛЕКСНАЯ МЕТОДИКА МНОГОКРИТЕРИАЛЬНОЙ ОЦЕНКИ АРХИТЕКТУРНЫХ СТИЛЕЙ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

A COMPREHENSIVE METHODOLOGY FOR MULTI-CRITERIA ASSESSMENT OF SOFTWARE ARCHITECTURAL STYLES

I. Chernikov
A. Bolotov
S. Bronnikov

Summary. The aim of the research is to address the gap in the field of justified choice of architectural paradigms by creating a comprehensive methodology for comparative analysis. The scientific novelty lies in proposing a holistic algorithm that integrates three groups of criteria: quantitative, qualitative, and economic. To aggregate heterogeneous data, normalization procedures and an adaptive weighting mechanism based on Thomas Saaty's Analytic Hierarchy Process (AHP), validated through the consistency index calculation, are introduced. The article details the stages of the methodology — from formulating project requirements and selecting metrics to conducting an integrated assessment and interpreting the results. The main result is a formalized, reproducible procedure that allows transferring the architectural choice from the realm of purely subjective preferences to the sphere of quantitatively justified decision-making. The article is methodological in nature and serves as a basis for subsequent experimental approbation.

Keywords: software architecture, comparative analysis, multi-criteria assessment, Analytic Hierarchy Process (AHP), cost analysis, prototyping, microservices, monolith.

Черников Илья Сергеевич
Уфимский университет науки и технологий
chernikov-i@inbox.ru
Болотнов Анатолий Миронович
доктор физико-математических наук, профессор,
Уфимский университет науки и технологий
bolotovAM@mail.ru
Бронников Сергей Анатольевич
кандидат педагогических наук, доцент, Бирский филиал,
Уфимский университет науки и технологий, г. Бирск
bronbir@rambler.ru

Аннотация. Целью исследования является устранение дефицита в области обоснованного выбора архитектурных парадигм за счёт создания комплексной методики сравнительного анализа. Научная новизна заключается в предложении целостного алгоритма, объединяющего три группы критериев: количественные, качественные и экономические. Для агрегации разнородных данных вводятся процедуры нормализации и механизм адаптивного взвешивания на основе метода анализа иерархий Т. Саати, валидируемый через расчёт индекса согласованности. В статье детально излагаются этапы методики — от формулирования требований проекта и выбора метрик до проведения интегральной оценки и интерпретации результатов. Основным результатом является формализованная, воспроизводимая процедура, позволяющая перевести выбор архитектуры из области только субъективных предпочтений в сферу количественно обоснованного решения. Статья носит методологический характер и служит основой для последующей экспериментальной апробации.

Ключевые слова: архитектура программного обеспечения, сравнительный анализ, многокритериальная оценка, метод анализа иерархий, анализ затрат, прототипирование, микросервисы, монолит.

Актуальность проблемы

В современных условиях стремительной цифровизации к программному обеспечению предъявляются все более высокие требования по производительности, масштабируемости, надёжности и скорости внесения изменений. Выбор базовой архитектурной парадигмы — монолитной, микросервисной или гибридной — является одним из наиболее судьбоносных решений на этапе проектирования информационной системы, определяющим ее технические и экономические характеристики на всём жизненном цикле. Однако данный выбор зачастую осуществляется исключительно на основе субъективных критериев, экспертных оценок или сиюминутных трендов, без наличия объективного количественного обоснования. Отсутствие унифицированного подхода к сравнению архитектур приводит к рискам реализации неоптимальных решений, росту стоимости обслуживания и проблемам с производительностью.

Степень изученности проблемы

Вопросы преимуществ и недостатков различных архитектурных стилей широко освещаются в профессиональной литературе, научных статьях и технoблогах [1]. Существует множество работ, посвященных теоретическому описанию структуры архитектуры, паттернам проектирования и примерам их реализации. Однако большинство существующих исследований носят качественный, обзорный характер или фокусируются на узком наборе технических критериев [2, 3]. При этом широко известные методики архитектурной оценки зачастую избыточно сложны для применения в проектах

среднего масштаба и не предлагают детальных, формализованных процедур для учёта операционных издержек и человеческих факторов. Таким образом, наблюдается дефицит комплексных, практико-ориентированных методик, предлагающих формализованный, измеримый и воспроизводимый алгоритм сравнительной оценки архитектур ПО, интегрирующий как объективные количественные метрики, так и структурированные субъективные. Именно этот дефицит и формирует научную проблему данного исследования [4].

Цель данной работы заключается в разработке комплексной методики сравнительного анализа архитектурных стилей программного обеспечения. Методика представляет собой формализованный алгоритм, обеспечивающий многоаспектную оценку с учётом объективных количественных и субъективных качественных данных. Конечным результатом применения методики является количественно обоснованное решение для выбора оптимальной архитектурной парадигмы в условиях конкретного проекта.

Для достижения поставленной цели были решены следующие *задачи*:

1. Проанализированы и выбраны подходы и их методы для сравнения архитектур программного обеспечения.
2. Разработан алгоритм проведения сравнительного анализа, включающий этапы подготовки, тестирования, анализа и интерпретации данных.
3. Систематизированы и формализованы ключевые качественные критерии (атрибуты качества) сравнения архитектурных стилей.
4. Определена система количественных метрик для экономических и операционных показателей, также выбран инструментарий для их измерения.
5. Разработана модель взвешивания критериев для формирования интегральной оценки, основанной на приоритетах проекта.
6. Проведена демонстрация работы методики на абстрактных примерах.

Следует отметить, что в рамках настоящего исследования приводятся модельные примеры, достаточные для демонстрации принципов работы и иллюстрирующие применение разработанной методики. В перспективе планируется публикация результатов апробации методики на практическом кейсе проектирования информационной системы для сферы дополнительного образования, где будет проведено комплексное сравнение монолитной, микросервисной и гибридной модульной монолитной архитектур, что позволит оценить практическую эффективность предложенного подхода в условиях реального проектирования [5, 6].

1. Подходы и методы сравнения архитектур

В современной практике сравнения архитектур программного обеспечения существует множество подходов, каждый из которых решает специфические задачи и имеет свои особенности применения. В рамках данной статьи будет рассмотрено как наиболее распространенные подходы, которые хорошо зарекомендовали себя в практике архитектурного тестирования, так и малоизвестные методы позволяющие взглянуть на патовые ситуации выбора с других ракурсов.

Архитектурный обзор или ревью это структурированная экспертная оценка проектных решений, проводимая командой менеджера, архитекторов и разработчиков. Основной фокус направлен на соответствии архитектуры бизнес-требованиям, стандартам и долгосрочным целям проекта [7].

Прототипирование, то есть создание прототипов — позволяет проверить жизнеспособность архитектурных решений на практике. В рамках этого подхода разрабатываются упрощенные версии ключевых компонентов системы для валидации основных технических решений и практической проверки *критических сценариев* использования. Реализация наиболее критичных или спорных модулей системы в нескольких вариантах разных архитектурных решений — это дорого и долго, но даёт самые точные результаты [8].

Сценарные тесты — метод, который фокусируется на проверке архитектуры через призму конкретных пользовательских сценариев. Тестируется способность архитектуры поддерживать различные варианты использования системы, включая нестандартные ситуации и граничные случаи (кросс-модульные операции).

Бенчмаркинг и метрики производительности — этот метод продолжает концепцию подхода прототипирования, расширяя и углубляя тестирование системы. Суть этого метода в нагрузочном тестировании прототипов с идентичным функционалом, но созданных на основе различных архитектурных решений [9-11]. Ключевым метриками измерения производительности могут быть:

- **Пропускная способность,** показывающая количество операций в секунду (RPS — Requests Per Second).
- **Задержка**: P50, P95, P99 (перцентили).

Перцентиль — это показатель, который определяет значение, ниже которого находится определенный процент наблюдений в распределении. Например, **P50** (50-й перцентиль) — медиана распределения, **P95** (95-й перцентиль) — значение, ниже которого находятся 95 % данных, **P99** (99-й перцентиль) — значение, ниже которого находятся 99 % данных.

В анализе производительности систем P50 показывает среднее время отклика системы, P95 помогает оценить работу системы для большинства пользователей, а P99 выявляет проблемы с производительностью для самых «медленных» запросов.

- **Потребление ресурсов**, показывающее потребление/использование процессора (CPU), оперативной памяти (RAM), объём данных, передаваемых через сетевую подсистему за определённый период времени (Network I/O).

Анализ затрат — это комплексный метод оценки всех ресурсов, необходимых для создания и поддержки программного обеспечения на протяжении всего его жизненного цикла. Метод сравнения работает не только по финансовым затратам, но и по затратам времени и человеческих ресурсов [12].

Финансовыми затратами считаются прямые расходы (зарплаты, оборудование), косвенные расходы (аренда, коммунальные платежи), операционные расходы, капитальные вложения. К человеческим ресурсам можно отнести количество задействованных специалистов, квалификацию персонала, распределение нагрузки, обучение и развитие. Временными затратами можно учитывать длительность этапов разработки, сроки реализации задач, время на тестирование, период поддержки [13].

Метод анкетирования — это метод сбора оценок качественных характеристик, который дополняет количественные метрики и помогает оценить «человеческий фактор» при работе с разными архитектурами [14].

После завершения разработки прототипа по каждой архитектуре команда разработчиков и тестировщиков (потенциальных пользователей) заполняет анонимную анкету, оценивая свои впечатления и сложности работы по различным аспектам. Целями анкетирования могут быть:

- Оценка субъективного восприятия архитектуры разработчиками и тестировщиками.
- Выявление скрытых проблем, не проявляющиеся в количественных метриках или других качественных логических методах.
- Измерение когнитивной нагрузки и удобства работы.
- Оценка влияния на скорость разработки с точки зрения команды.

Этот метод дает ценную информацию о том, насколько архитектура удобна для живых разработчиков, что критически важно для долгосрочного успеха проекта [14].

2. Алгоритм проведения комплексного сравнительного анализа

После анализа и выбора сравнительных методик сформируем последовательный алгоритм по проведению сравнительного анализа:

Этап 1: Подготовительный

Определение контекста и ограничений
- Формулировка бизнес-требований к системе дополнительного образования.
- Определение технических ограничений (бюджет, сроки, команда).
- Выбор сравниваемых архитектур.

Разработка функциональной спецификации
- Создание единого набора пользовательских функциональных возможностей для всех архитектур.

Подготовка инструментария
- Подготовка стенда для тестирования (одинаковые hardware/software характеристики).

Разработка анкет
- Создание пула вопросов: общих, детальных по критериям, с открытым ответом и установка шкалы оценивания для анкетирования.

Этап 2: Разработка прототипов

Создание функционально эквивалентных прототипов
- Архитектура 1, архитектура 2, …

Реализация базового функционала
- Функция 1, функция 2, …
- Операция 1, операция 2, …

Этап 3: Тестирование и замеры

Проведение анкетирования
- Анкетирование по этапам: во время проектирования ИС, разработки, после реализации (тестирование). Это нужно для отслеживания сложности работы на разных этапах развития ПО, а также для уменьшения эффекта новизны, когда первая архитектура может оцениваться строже. Анкетирование проводится анонимно для честных субъективных данных.

Проведение архитектурного обзора (качественная оценка)
- Экспертная оценка по чек-листу атрибутов качества.
- Анализ соответствия принципам проектирования.

- Выполнение сценарных тестов.
- Тестирование типичных пользовательских сценариев.
- Оценка удобства разработки и внесения изменений.

Проведение бенчмаркинга (количественные замеры)
- Нагрузочное тестирование для каждого прототипа.
- Замер метрик производительности на идентичной нагрузке.
- Анализ потребления ресурсов.

Анализ затрат
- Оценка экономических, временных затрат и затрат человеческих ресурсов.

Этап 4: Анализ и апробация

Сбор и обработка результатов
- Нормализация полученных данных.
- Расчет интегральных показателей для каждой архитектуры.

Формулировка выводов и рекомендаций
- Сравнительный анализ комплексных оценок.
- Формулировка рекомендаций по выбору архитектуры.

3. Определение ключевых качественных критериев оценки

После составления алгоритма для сравнительного анализа, нужно определить ключевые качественные атрибуты оценки и сформировать систему количественных метрик, которые будут учитываться в сравнительном анализе.

В **группу технических атрибутов качества** входит ***масштабируемость***, которая делится на горизонтальное и вертикальное масштабирование.

Горизонтальное масштабирование позволяет увеличивать производительность системы путем добавления новых элементов, в то время как *вертикальное масштабирование* улучшает характеристики существующих.

Избирательное масштабирование предоставляет возможность масштабировать только те компоненты системы, которые испытывают проблемы, что позволяет оптимизировать ресурсы и минимизировать затраты.

Отказоустойчивость системы включает изоляцию сбоев и деградацию функциональности. *Изоляция сбоев* локализует последствия отказа одного компонента, предотвращая распространение проблемы на всю систему. *Деградация функциональности* позволяет системе продолжать работать при частичных отказах, обеспечивая минимальное снижение производительности при максимально доступном функционале.

Сложность системы охватывает когнитивную нагрузку, операционную сложность и сложность отладки. *Когнитивная нагрузка* отражает простоту понимания архитектуры для новых разработчиков, *операционная сложность* — трудоемкость развертывания и поддержки системы, а *сложность отладки* — скорость поиска и исправления возникающих ошибок.

Гибкость системы включает внесение изменений, технологическую независимость и адаптивность к изменениям требований. *Внесение изменений* должно быть быстрым и безопасным, чтобы минимизировать время простоя системы. *Технологическая независимость* позволяет использовать различные технологии в компонентах, что повышает гибкость и адаптивность системы. *Адаптивность к изменениям требований* означает способность архитектуры эволюционировать и адаптироваться к новым условиям и требованиям.

Поддерживаемость системы охватывает модульность, тестируемость и документированность. *Модульность* обеспечивает чёткость границ между компонентами, что упрощает разработку и поддержку системы. *Тестируемость* отражает простоту написания модульных и интеграционных тестов, что повышает качество и надёжность системы. *Документированность*, в свою очередь, оценивает качество и актуальность документации, что облегчает работу новых разработчиков и поддержку системы.

4. Определение ключевых количественных метрик

Наряду с качественными атрибутами формируется и **система количественных метрик производительности (бенчмаркинг)** [9, 10]**.**

Пропускная способность определяется метрикой Requests Per Second (RPS). Она измеряет количество ключевых операций, выполненных за секунду. Например, одновременные записи 100 пользователей позволяет оценить, насколько эффективно система справляется с нагрузкой.

Задержка измеряется с помощью метрик P50, P95 и P99 по времени ответа в миллисекундах. Эти метрики показывают время, необходимое для выполнения операций. Например, получение отчёта со сложными расчётами помогает выявить, как долго система обрабатывает запросы.

Потребление ресурсов включает в себя несколько аспектов: CPU Usage, Memory Usage и Network I/O. CPU Usage отслеживает среднюю и пиковую загрузку процессора в процентах, что позволяет оценить эффективность его использования. Memory Usage показывает объём занимаемой оперативной памяти в мегабайтах, что важно для определения, достаточно ли памяти для выполнения задач. Network I/O измеряет объём передаваемых данных в мегабайтах в секунду, что помогает оценить эффективность работы с сетью.

5. Определение метрик анализа затрат

Для крупных и масштабных проектов и тех малых проектов, которые требуют строгого контроля расходов, в систему сравнения добавляются **метрики анализа затрат**. Эти метрики делятся на три группы: финансовые затраты, временные затраты и затраты на человеческие ресурсы [12, 13].

Первая группа — ***финансовые затраты*** включает инфраструктурные затраты, затраты на разработку и эксплуатационные затраты. В инфраструктурные затраты могут входить расходы на серверы, базы данных, сетевые ресурсы, лицензии программного обеспечения и мониторинг с аналитикой. Затраты на разработку могут включать оплату труда разработчиков и обучение команды. Эксплуатационные затраты охватывают расходы на DevOps/SRE команду, круглосуточную поддержку, резервное копирование и аварийное восстановление.

Вторая группа — ***временные затраты***, которые включают общее время от идеи до продакшена, время разработки функции, исправления бага, развертывания и масштабирования.

Третья группа — ***затраты на человеческие ресурсы***. В неё можно внести сложность командной работы, которая включает размер необходимой команды, время ввода нового разработчика, частоту переключения контекста и время код-ревью.

6. Нормализация результатов метрик

После определения качественных и количественных метрик с их характеристиками можно разработать модель взвешивания критериев для формирования интегральной оценки на основе приоритетов под конкретный проект.

Перед тем как объединить оценки для получения итогового результата и вывода следует привести все данные к единой шкале, то есть нормализовать.

Для **нормализации количественных метрик** воспользуемся двумя формулами:

Формула для группы метрик «***чем больше — тем лучше***», к таким метрикам относятся, например, производительность и пропускная способность:

$$N_{ij} = \frac{X_{ij} - \min(X_j)}{\max(X_j) - \min(X_j)}, \quad (1)$$

где N_{ij} — нормализованное значение j-го критерия для i-й архитектуры;

X_{ij} — исходное значение замера по j-му критерию для i-й архитектуры;

$\max(X_j), \min(X_j)$ — максимальное и минимальное значения замеров по j-му критерию среди всех архитектур.

Формула для группы метрик «***чем меньше — тем лучше***», к таким метрикам относятся, например, задержка, стоимость и потребление ресурсов:

$$N_{ij} = \frac{\max(X_j) - X_{ij}}{\max(X_j) - \min(X_j)}, \quad (2)$$

где N_{ij} — нормализованное значение j-го критерия для i-й архитектуры;

X_{ij} — исходное значение замера по j-му критерию для i-й архитектуры;

$\max(X_j), \min(X_j)$ — максимальное и минимальное значения замеров по j-му критерию среди всех архитектур.

Для **нормализации качественных метрик** т.е. экспертных оценок — воспользуемся формулой составной оценки качественной метрики, составные части которой были разобраны ранее:

$$Q_{ij} = \sum_{k=1}^{n} c_{jk} * q_{ijk}, \quad (3)$$

где Q_{ij} — составное значение оценки j-го критерия для i-й архитектуры;

q_{ijk} — замер значения k-го подпункта по j-му критерию для i-й архитектуры (например, оценка от 1 до 10);

c_{jk} — коэффициент важности k-го подпункта для j-го критерия (значения от 0 до 1);

n — количество подпунктов критерия Q_{ij}.

Коэффициенты c_{jk} формируются на основе анкетирования ключевых участников команды разработки и агре-

гации их ответов. Это необходимо, так как для каждой команды и проекта критерии из подпунктов имеют свои приоритеты.

Поскольку коэффициенты определяют значимость подпунктов в общей оценке, их сумма должна равняться единице:

$$\sum_{k=1}^{n} c_k = 1, \quad (4)$$

где n — количество подпунктов критерия; c_k — коэффициент важности для k-го подпункта критерия.

Например, определим формулы составных оценок четырёх критериев для условной команды разработчиков проекта, где сумма коэффициентов подпунктов соответствует условию (4):

Масштабируемость = 0.4*Горизонтальное М. + 0.4*Вертикальное М. + 0.2 * Избирательное М.;

Отказоустойчивость = 0.25*Изоляция сбоев + 0.25*Восстановление + 0.2*Избыточность + 0.15*Деградация + 0.15*Мониторинг;

Сложность = 0.3*Когнитивная нагрузка + 0.25*Эксплуатация + 0.25*Разработка+ 0.15*Тестирование + 0.05*Отладка;

Гибкость = 0.25*Лёгкие изменения + 0.1*Независимость технологий+ 0.2*Адаптивность изменений + 0.25*Расширяемость функционала + 0.1*Гибкость конфигурации + 0.1*Простота интеграции.

Теперь следуя формулам (1)–(2), можно получить нормализованные значения количественных критериев, исходные значения которых представляют собой численные цифровые замеры бенчмарками, а для нормализованных значений качественных критериев формула (3), где исходные значения будут шкалированные оценки из анкет, например от 1 до 10.

7. Определение весов результирующих оценок

Однако для объединения результатов нормализации качественных и количественных оценок метрик следует определить веса приоритетов критериев выбора для этих двух разных групп. Для этого воспользуемся ***подходом анализа иерархий*** разработанным американским математиком Томасом Саати для задач связанных с принятием решений, а точнее используем метод парных сравнений со шкалой Саати [15, 16].

Шкала позволяет количественно оценивать суждения эксперта путем присвоения числовых значений от «1» до «9» (и их обратных величин). Значения от «1» до «9» показывают превосходство одного элемента над другим, чем больше число, тем сильнее превосходство элемента. Обратные значения («1/9» до «1/1») отражают превосходство второго элемента. Значение «1» означает равную важность элементов (см. *Таблица 1*).

Таблица 1.

Таблица значений шкалы Саати

Числовое значение	Словесное определение
9	Абсолютное превосходство
8	Очень сильное превосходство
7	Сильное превосходство
6	Умеренно-сильное превосходство
5	Умеренное превосходство
4	Слабое превосходство
3	Некоторое превосходство
2	Едва заметное превосходство
1	Равная важность
1/2	Едва заметное превосходство другого элемента
1/3	Некоторое превосходство другого элемента
1/4	Слабое превосходство другого элемента
1/5	Умеренное превосходство другого элемента
1/6	Умеренно-сильное превосходство другого элемента
1/7	Сильное превосходство другого элемента
1/8	Очень сильное превосходство другого элемента
1/9	Абсолютное превосходство другого элемента

На основе анкетирования ключевых участников команды разработки и агрегации их ответов с использованием шкалы Саати формируется матрица парных сравнений размером $n*n$, где n — это количество приоритетно сравниваемых критериев [16]. Если a_{ij} элемент матрицы, то

$$a_{ij} \in \left\{\frac{1}{9}, \frac{1}{8}, \ldots, 1, 2, \ldots, 9\right\} (\text{шкала Саати})\, a_{ij} = \frac{1}{a_{ji}}; a_{ii} = 1.$$

После формирования матрицы рассчитывается вес критерия — геометрическое среднее для каждой строки-критерия:

$$p_i = \sqrt[n]{\prod_{j=1}^{n} a_{ij}}, \quad (5)$$

где p_i — ненормированный вес i-го критерия; a_{ij} — элемент матрицы парных сравнений в i-й строке, j-м столбце;

n — количество критериев;

$\prod_{j=1}^{n} a_{ij}$ — произведение элементов i-й строки.

После расчёта среднего геометрического значения веса по каждому критерию высчитывается его нормированный вес:

$$W_i = \frac{p_i}{\sum_{j=1}^{n} p_j}, \quad (6)$$

где W_i — итоговый нормированный вес i-го критерия;

p_i — ненормированный вес i-го критерия;

$\sum_{j=1}^{n} p_j$ — сумма ненормированных весов всех критериев.

Например, для условной команды разработчиков проекта составим матрицу парных сравнений приоритетов четырёх критериев и найдём их весы (5)–(6) (см. *Таблица 2*).

Таблица 2.

Пример матрицы парных сравнений

	Производительность (колич. метрика)	Масштабируемость (кач. метрика)	Стоимость (колич. метрика)	Скорость разработки (колич. метрика)
Производительность vs другие	1	3	5	2
Масштабируемость vs другие	1/3	1	4	1/2
Стоимость vs другие	1/5	1/4	1	1/3
Скорость разработки vs другие	1/2	2	3	1

Для данной матрицы нормированные веса критериев будут [0.42, 0.25, 0.1, 0.23] для Производительности, Масштабируемости, Стоимости и Скорости разработки соответственно. То есть для данной условной команды проекта важнее всего производительность системы, а наименее всего важна стоимость.

8. Проверка подлинности весовых коэффициентов

После получения нормализованных количественных оценок, нормализованных качественных составных оценок, а также приоритетных весов всех этих критериев — можно составить интегральную оценку по каждой сравниваемой архитектуре.

Однако перед расчётами необходимо пройти валидацию модели сравнения построенной командой разработчиков. Для этого необходимо рассчитать индекс согласованности (CI).

Индекс согласованности (CI) показывает, насколько логично и непротиворечиво эксперт расставил приоритеты в матрице попарных сравнений. Так если:

- CI = 0, показывает идеальную согласованность (эксперт абсолютно последователен и логичен),
- CI <0.1, показывает хорошую согласованность (приемлемо для практического использования),
- CI> 0.1, показывает плохую согласованность (эксперт противоречит сам себе).

Плохая согласованность указывает на ошибочное, противоречивое построение матрицы парного сравнения, чтобы это проверить воспользуемся расчётными формулами:

$$(Aw)_i = \sum_{j=1}^{n} (a_{ij} * w_j), \quad (7)$$

где $(Aw)_i$ — i-й элемент вектора приоритета матрицы;

a_{ij} — элемент матрицы парных сравнений в i-й строке, j-м столбце;

w_j — нормированный вес j-го критерия;

n — количество критериев.

$$\lambda_{max} = \frac{1}{n} \sum_{i=1}^{n} \frac{(Aw)_i}{w_i}, \quad (8)$$

где λ_{max} — максимальное собственное значение матрицы,

$(Aw)_i$ — i-й элемент вектора приоритета матрицы (7);

w_i — нормированный вес i-го критерия;

n — количество критериев;

$$CI = \frac{\lambda_{max} - n}{n - 1}, \quad (9)$$

где CI — индекс согласованности;

λ_{max} — максимальное собственное значение матрицы (8);

n — размер матрицы т.е. количество критериев.

В то же время *CI* тоже можно нормализовать относительно размера матрицы для более точного оценивания адекватности значений матрицы. Нужно разделить *CI*

на «ожидаемый» уровень несогласованности для случайной матрицы такого же размера — *RI*.

$$CR = \frac{CI}{RI}, \quad (10)$$

где *CR* — нормализованный индекс согласованности;

CI — индекс согласованности;

RI — среднее значение индекс согласованности для матриц размера n.

Значение RI берутся из эталонной таблицы случайных индексов, разработанной Томасом Саати (см. *Таблица 3*).

Таблица 3.

Эталонная таблица случайных индексов (Саати, 1980)

Размер матрицы (n)	1	2	3	4	5	6	7	8	9	10	…
Случайный индекс (RI)	0	0	0.58	0.90	1.12	1.24	1.32	1.41	1.45	1.49	…

Используя пример матрицы парных сравнений, вычислим λ_{max}, CI, CR по формулам (7)–(10):

$$\lambda_{max} = \frac{4.048+4.179+4.151+4.090}{4} \approx 4.117$$

$$CI = \frac{(\lambda_{max} - n)}{n-1} = \frac{(4.117-4)}{4-1} = \frac{0.117}{3} \approx 0.039$$

$$\text{Для } n=4, RI=0.90 \text{ (из эталонной таблицы)}$$

$$CR = \frac{CI}{RI} = \frac{0.039}{0.90} \approx 0.0433 (4.33\%)$$

По расчётам CR = 4.33 % значительно ниже порога в 10 %. Это означает, что экспертные оценки логически непротиворечивы и результатам можно доверять.

9. Формирование итоговых интегральных оценок архитектур

Исходя из вывода, что веса приоритетов для критериев непротиворечивы можно рассчитать интегральную оценку для каждого типа архитектуры, которые мы сравниваем:

$$S_i = \sum_{j=1}^{n} W_j * N_{ij}, \quad (11)$$

где S_i — интегральная итоговая оценка *i*-й архитектуры;

W_j — нормализованный вес *j*-го критерия для проекта;

N_{ij} — нормализованная оценка по *j*-му критерию для *i*-й архитектуры.

Таким образом, после расчётов (11) мы получаем комплексные оценки выбора по каждой архитектуре. Сопоставляя значения интегральных оценок архитектур, рекомендуется следующий подход к принятию решения:

- **Разница оценок меньше 0.05** обозначает, что архитектуры практически эквивалентны для вашего проекта — решение должно основываться на дополнительных факторах (навыки команды, стратегические соображения и т.п.).
- **Разница оценок от 0.05 до 0.15** показывает существенные различия — выбирайте архитектуру с наивысшим баллом, но проанализируйте её слабые стороны, они могут быть критическими для вашей команды.
- **Разница оценок больше 0.15** указывает на явного лидера — рекомендуется выбирать архитектуры с максимальной оценкой.

Тем не менее, в процессе комплексного сравнения были использованы методы анкетирования, предназначенные для выявления субъективного ощущения команды проекта при работе с различными архитектурами. Такой подход позволяет учесть не только формальные требования проекта, но и субъективные предпочтения команды, что особенно важно при оценке качественных критериев. Интеграция анкетирования в процесс взвешивания критериев обеспечивает более обоснованный и принимаемый командой результат выбора архитектуры. Именно на основе агрегированных оценок опросов мы можем определить более подходящую архитектуру, если критериальные оценки оказываются близкими для конкретного выбора или имеют критический недостаток.

Заключение

В результате исследования разработана и формально описана комплексная методика сравнительного анализа архитектурных стилей программного обеспечения. Ключевым научным вкладом работы является синтез в рамках одного алгоритма объективных количественных измерений, структурированной экспертной оценки качественных характеристик и анализа операционных издержек.

Предложенная методика преодолевает ограничения существующих подходов, которые носят чисто качественный обзорный характер либо фокусируются на узком спектре технических показателей. Она предоставляет проектировщикам систематизированный, формализованный и воспроизводимый инструмент, адаптируемый под приоритеты конкретного проекта с помощью метода анализа иерархий.

Перспективой дальнейших исследований является экспериментальная верификация методики на практи-

ке, её программная реализация в виде инструмента поддержки принятия решений, а также уточнение и валидация весовых коэффициентов для различных предметных областей. Последующая апробация и развитие данного подхода создают основу для формирования отраслевого стандарта в области обоснованного выбора архитектурных решений.

ЛИТЕРАТУРА

1. Черников И.С. Эволюция архитектурных парадигм: от спагетти-монолитов до облачных микросервисов / И.С. Черников // Актуальные вопросы инноваций и современные научные открытия: Сборник научных статей по материалам VIII Международной научно-практической конференции, Уфа, 12 сентября 2025 года. — Уфа: Общество с ограниченной ответственностью «Научно-издательский центр «Вестник науки», 2025. — С. 229–235.
2. Корниенко Д.В. Анализ методов оценки архитектуры программного обеспечения / Д.В. Корниенко, А.В. Никулин // Технологии и техника: пути инновационного развития: Сборник научных статей 3-й Международной научно-технической конференции, Воронеж, 17 июня 2025 года. — Воронеж: ЗАО «Университетская книга», 2025. — С. 146–153.
3. Терешкова А.А. Методика построения целевой архитектуры предприятия на основе анализа бизнес-потребностей / А.А. Терешкова, Л.К. Бобров // Молодежный научный форум: технические и математические науки. — 2017. — № 5(45). — С. 91–99.
4. Урсегова Н.А. Технология проектирования и разработки методических рекомендаций / Н.А. Урсегова // Педагогика: традиции и инновации: Материалы IX Международной научной конференции, Казань, 20–23 января 2018 года. — Казань: Бук, 2018. — С. 96–98.
5. Черников И.С. Разработка функциональных и технических требований для информационной системы дополнительного образования: обзор перспективных архитектурных решений / И.С. Черников // Большая студенческая конференция: сборник статей XI Международной научно-практической конференции : в 3 ч., Пенза, 15 декабря 2024 года. — Пенза: Наука и Просвещение (ИП Гуляев Г.Ю.), 2024. — С. 12–15.
6. Черников И.С. Цифровизация дополнительного образования: роль информационных технологий и проектирование новых решений / И.С. Черников, С.А. Бронников // Современные тенденции развития теории и практики социально-педагогической, психолого-педагогической деятельности и социальной работы: Материалы VI Всероссийской научно-практической конференции, Бирск, 20 декабря 2024 года. — Бирск: Уфимский университет науки и технологий, 2024. — С. 400–405.
7. Ричардс Марк, Форд Нил Фундаментальный подход к программной архитектуре: паттерны, свойства, проверенные методы. — СПб.: Питер, 2023. — 448 с.: ил. — (Серия «Для профессионалов»).
8. Капралова Н.Н. Прототипирование как этап разработки программного обеспечения / Н.Н. Капралова, Ю.К. Мухин // Вестник ФКУ НИИИТ ФСИН России: выпуск 7. — Тверь: Федеральное казенное учреждение «Научно-исследовательский институт информационных технологий Федеральной службы исполнения наказаний», 2024. — С. 39–44.
9. Maneva Nelly, Daneva Maya, Petrova Valia Benchmarking in Software Development // Benchmarking — Theory and Practice. 1995. pp.166-175.
10. Шакирзянова А.А. Аспекты измерения производительности.NET приложений / А.А. Шакирзянова // Актуальные проблемы общества, экономики и права в контексте глобальных вызовов (шифр — МНКАВ 32): Сборник материалов XXXII Международной научно-практической конференции, Москва, 28 октября 2024 года. — Москва: ООО «Издательство Академическая среда», 2024. — С. 112–123.
11. Болотнов А.М., Гарифуллина С.Р., Коробчинская О.Г., Нурисламова Э.А. Эффективность исполняемого кода в некоторых языках программирования // Наука и инновации в XXI веке: актуальные вопросы, открытия и достижения: сборник статей XV Международной научно-практической конференции, Пенза, 23 октября 2019 года. — Пенза: «Наука и Просвещение» (ИП Гуляев Г.Ю.), 2019. — С. 12–17.
12. Зубкова Т.М. Расчет затрат при проектировании программного обеспечения / Т.М. Зубкова, С.А. Купырев // Символ науки: международный научный журнал. — 2024. — Т. 3, № 4–1. — С. 12–18.
13. Стефанова Н.А. Оценка стоимости разработки программного обеспечения / Н.А. Стефанова, Д. Курбангелдыев // Актуальные вопросы современной экономики. — 2020. — № 1. — С. 67–72.
14. Ковалев Д.А. Проведение экспериментального прикладного исследования на начальных этапах ИТ-разработки: фокус-группы, личное интервью, анкетирование, получение экспертного мнения / Д.А. Ковалев, А.В. Дроздов // Международный журнал гуманитарных и естественных наук. — 2017. — № 10. — С. 143–146.
15. Вотинцева В.О. Применение метода анализа иерархии для выбора методик проектирования системной архитектуры предприятия / В.О. Вотинцева, М.С. Сахипова, А.И. Дерябин // Математика и междисциплинарные исследования — 2016 : Сборник докладов всероссийской научно-практической конференции молодых ученых с международным участием, Пермь, 16–19 мая 2016 года / гл. ред. Ю.А. Шарапов; Пермский государственный национальный исследовательский университет. — Пермь: Пермский государственный национальный исследовательский университет, 2016. — С. 201–207.
16. Саати Т.Л. Об измерении неосязаемого. Подход к относительным измерениям на основе главного собственного вектора матрицы парных сравнений / Т.Л. Саати // Cloud of Science. — 2015. — Т. 2, № 1. — С. 5–39.

© Черников Илья Сергеевич (chernikov-i@inbox.ru); Болотнов Анатолий Миронович (bolotovAM@mail.ru);
Бронников Сергей Анатольевич (bronbir@rambler.ru)

Журнал «Современная наука: актуальные проблемы теории и практики»