

Научно-практическая статья  
УДК 338.225  
JEL classification: I21, I28  
EDN: VJPOGN

## СРАВНИТЕЛЬНЫЙ АНАЛИЗ СПОСОБОВ ИНЖЕНЕРИИ ТРЕБОВАНИЙ К ПРОГРАММНОМУ ОБЕСПЕЧЕНИЮ

**Аникин Н.К.,**

обучающийся магистратуры кафедры информационных технологий,  
Донецкий филиал РАНХиГС,  
г. Донецк, Донецкая Народная Республика, Российская Федерация.  
E-mail: smart.rebook@gmail.com

**Аннотация. Цель.** Разработать понятную и проверяемую методику выбора способов фиксации требований к программному обеспечению и определить минимально достаточные связки артефактов для разных задач, с отдельным акцентом на интеграционные сценарии.

**Материалы и методы.** В качестве материалов рассмотрены основные артефакты инженерии требований: SRS, пользовательские истории и критерии приёмки (в т.ч. Gherkin), BPMN, UML (диаграммы последовательностей и состояний), DFD/контекстная диаграмма, ER-модель и словарь данных, а также контракты API (OpenAPI, AsyncAPI). Сравнение выполнено по критериям понятности, проверяемости, полноты/непротиворечивости, трудоёмкости сопровождения, трассируемости к NFR и пригодности к интеграциям, с балльной оценкой по шкале 0–5 и весами критериев; применена экспертная оценка и сведение результатов в матрицу «задача/артефакт» и итоговую таблицу баллов.

**Результаты.** Получена матрица соответствия задач и артефактов и подтверждающая её количественная оценка. Показано, что для интеграций наиболее результативна связка «контекст + DFD + контракты API (OpenAPI/AsyncAPI) + UML sequence», а при координации шагов между системами уместно добавление фрагмента BPMN. Для описания процессов лидирует BPMN, для жизненного цикла сущностей – UML state, для данных и отчётности – ER-модель и словарь данных; SRS демонстрирует стабильно высокую ценность за счёт полноты и трассируемости к NFR, но не заменяет поведенческие и интеграционные модели.

**Заключение.** Предложенная методика помогает выбирать артефакты по задаче и избегать избыточной документации, удерживая принцип «минимально достаточного набора». Это снижает риск несогласованности требований и дефектов на стыках систем, повышая проверяемость, трассируемость к NFR/SLA и управляемость изменений в проектах разработки и интеграции.

**Ключевые слова:** инженерия требований, SRS, пользовательские истории, BPMN, UML, DFD, ER, API (OpenAPI, AsyncAPI), NFR, SLA, интеграции, матрица выбора артефактов.

**Для цитирования:** Аникин Н.К., Сравнительный анализ способов инженерии требований к программному обеспечению. «Феноменус». 2026. №1(33). С. 114–122. EDN: VJPOGN



Scientific and practical article  
UDC 338.225  
JEL classification: I21, I28  
EDN: VJPOGN

## COMPARATIVE ANALYSIS OF SOFTWARE REQUIREMENTS ENGINEERING METHODS

**Nikita K. Anikin,**

Master's degree student of the Department of Information Technology,  
Donetsk Branch of RANEPa,  
Donetsk, Donetsk People's Republic, Russian Federation.  
E-mail: smart.rebook@gmail.com

**Annotation. Objective.** To develop a clear and verifiable methodology for selecting methods for capturing software requirements and to determine the minimum sufficient set of artifacts for different tasks, with a separate focus on integration scenarios.

**Materials and methods.** The main artifacts of requirements engineering are considered as materials: SRS, user stories and acceptance criteria (including Gherkin), BPMN, UML (sequence and state diagrams), DFD/context diagram, ER model and data dictionary, as well as API contracts (OpenAPI, AsyncAPI). The comparison was made based on the criteria of clarity, verifiability, completeness/consistency, maintenance complexity, traceability to NFR, and suitability for integration, with a score on a scale of 0–5 and criterion weights; Expert assessment was applied and the results were compiled into a “task/artifact” matrix and a final score table.

**Results.** A matrix of task and artifact correspondence and a quantitative assessment confirming it were obtained. It was shown that for integrations, the most effective combination is “context + DFD + API contracts (OpenAPI/AsyncAPI) + UML sequence,” and when coordinating steps between systems, it is appropriate to add a BPMN fragment. BPMN is the leader for describing processes, UML state for the entity lifecycle, and the ER model and data dictionary for data and reporting. SRS demonstrates consistently high value due to its completeness and traceability to NFR, but does not replace behavioral and integration models.

**Conclusion.** The proposed methodology helps to select artifacts for the task and avoid excessive documentation, adhering to the principle of a “minimally sufficient set.” This reduces the risk of inconsistencies in requirements and defects at system interfaces, increasing verifiability, traceability to NFR/SLA, and manageability of changes in development and integration projects.

**Keywords:** requirements engineering, SRS, user stories, BPMN, UML, DFD, ER, API (OpenAPI, AsyncAPI), NFR, SLA, integrations, artifact selection matrix.

**For citation:** Anikin, N.K., (2026). Comparative analysis of software requirements engineering methods. Phenomenus, 1(33), 114-122. EDN: NKCYZQ

### Постановка проблемы в общем виде

В проектах по созданию программного обеспечения команды используют разные способы фиксации требований: текстовые документы (например, спецификация требований – SRS), модели процессов и поведения (BPMN, UML), модели потоков и данных (DFD, ER-модель), а также контракты интерфейсов для интеграций (API на основе OpenAPI или AsyncAPI). На практике эти артефакты создаются неравномерно: где-то ограничиваются пользовательскими историями, где-то рисуют только диаграммы, а

где-то подробно описывают контракты, но не показывают контекст процессов. В результате повышаются риски несогласованности, растут сроки согласования и стоимость сопровождения [1].

При этом под «фиксацией требований» понимаются не только формальные документы, но и любые согласованные представления о том, что именно должна делать система, в каких условиях, с какими ограничениями и как будет проверяться результат. В современных командах требования нередко существуют одновременно в нескольких формах: в виде описания бизнес-целей, набора



пользовательских сценариев, схем данных, диаграмм взаимодействий и технических контрактов. Однако, чем больше распределена команда и чем больше участников у процесса (аналитики, разработчики, тестировщики, представители бизнеса, внешние подрядчики), тем важнее становится единый и понятный способ описания, который снижает риск разночтений и уменьшает «шум» при обсуждениях.

Корневая проблема – отсутствие понятных правил выбора «какой артефакт нужен под какую задачу». Одни задачи требуют представить цели пользователей и границы системы, другие – описать маршрут бизнес-процесса, третьи – зафиксировать обмен данными между системами и его ошибки. Если вместо целевого артефакта использовать неподходящий, теряется проверяемость требований, ухудшается трассируемость к нефункциональным требованиям (NFR), возрастает вероятность конфликтов в данных и дефектов на этапе валидации.

На практике выбор артефактов часто происходит стихийно: команда использует то, что привычнее, быстрее или «принято в компании», а не то, что лучше подходит под конкретную задачу. Например, пользовательские истории удобны для планирования разработки, но плохо раскрывают структуру данных и интеграционные зависимости; диаграммы процессов помогают понять последовательность шагов, но не фиксируют формат сообщений и правила ошибок; спецификация требований хорошо покрывает полноту и ограничения, но без моделей поведения может оставлять неоднозначности в сценариях. В итоге один и тот же вопрос («как система должна реагировать?») обсуждается по нескольким кругам, а разрозненные описания начинают противоречить друг другу.

Сложнее всего обстоит дело с интеграциями проектов нескольких систем. Большинство инцидентов возникает не из-за алгоритмов, а на стыках систем на уровне интерфейсов обмена: неочевидные зависимости, неформализованные политики повторов, отсутствие идемпотентности (устойчивости к повторной обработке одного и того же запроса без изменения результата), разночтения в версиях схем. Когда контекст и потоки данных не показаны явно, контракты API не согласованы с диаграммами последовательностей, а целевые уровни сервиса (SLA) не привязаны к шагам взаимодействия, команда теряет управляемость. Это приводит к росту числа дефектов,

ручным обходным операциям и «бумажным» согласованиям, которые не помогают при эксплуатации [2].

Интеграционные контуры обычно включают множество «невидимых» требований, которые редко формулируются в начале проекта, но критичны для стабильной работы: ограничения по времени ответа, лимиты запросов, правила повторной отправки сообщений, обработка частичных отказов, логирование и трассировка, требования к мониторингу и уведомлениям, требования к безопасности и разграничению доступа. Отдельное место занимают вопросы согласования версий: изменение формата сообщения или поля в схеме может нарушить работу внешнего потребителя, даже если сама логика внутри системы не менялась. Поэтому интеграции требуют не просто описания функций, а согласования контекстной картины, потоков данных и поведения при ошибках, чтобы эксплуатация не превращалась в постоянный ручной «разбор полётов».

Дополнительная проблема – избыточная документация. Попытка «описать всё сразу» создаёт тяжёлый набор артефактов, которые сложно поддерживать в актуальном состоянии. Через несколько итераций часть материалов устаревает, и разработчики перестают им доверять [3]. Нужен принцип минимально достаточного набора: столько и таких артефактов, сколько требуется для согласования решений, проектирования, тестирования и сопровождения, без лишнего.

Таким образом, существует практическая потребность в системном сравнении способов инженерии требований с опорой на признанные стандарты и руководства (например, ISO/IEC/IEEE 29148 и 12207, SWEBOK, BABOK, спецификации BPMN, UML, OpenAPI и AsyncAPI) [4]. Требуется показать, какой артефакт решает какую задачу, как связать их между собой и как учесть интеграционные аспекты: контекст, потоки, контракты, последовательности, NFR и SLA.

Цель исследования – разработать понятную и проверяемую методику выбора способов фиксации требований к программному обеспечению, которая связывает типовые задачи анализа и проектирования с подходящими артефактами (спецификация требований – Software Requirements Specification, SRS; Business Process Model and Notation, BPMN; Unified Modeling Language, UML; Data Flow Diagram, DFD; ER-модель; контракты Application Programming Interface, API на основе OpenAPI и AsyncAPI) и задаёт

минимально достаточные связи для интеграций.

Данная методика опирается на признанные стандарты и руководства (SWEBOOK, BABOK) и учитывает критерии понятности, проверяемости, трассируемости к нефункциональным требованиям, трудоёмкости сопровождения и соответствия целям интеграции.

Практическая значимость методики заключается в том, что она помогает командам быстрее достигать согласования между участниками проекта, снижать число правок на этапе тестирования и уменьшать риск интеграционных дефектов при вводе в эксплуатацию. Теоретическая значимость связана с формированием структурированного подхода к сопоставлению артефактов инженерии требований и уточнением их роли в обеспечении качества требований, особенно в условиях интеграции нескольких информационных систем. В перспективе предложенный подход может быть расширен за счёт учёта отраслевых требований, специфики регуляторных ограничений и применения автоматизированных инструментов трассируемости.

#### **Изложение основного материала исследования**

В основе сравнения лежит простая идея: каждый способ фиксации требований решает свою задачу и даёт ценность при определённых условиях. Чтобы выбор был осознанным, сначала различим четыре класса артефактов, затем определим критерии оценки и покажем, как эти артефакты работают вместе в интеграционных сценариях. Опора – международные руководства по инженерии требований и жизненному циклу программного обеспечения, а также отраслевые спецификации моделирования и описания интерфейсов.

Текстовая спецификация требований (Software Requirements Specification, SRS) [5] фиксирует границы системы, словарь терминов и проверяемые функциональные и нефункциональные требования. Этот документ даёт единое поле смыслов для команды и внешних участников. В нём удобно поддерживать трассируемость к целям и метрикам качества, так как структура стандартизована.

На практике SRS удобна тем, что позволяет фиксировать требования в «проверяемой» форме: каждое требование можно связать с источником (стейкхолдером), бизнес-целью, риском или метрикой качества, а также с тестом или приёмочным критерием. Кроме того,

именно текстовая спецификация чаще всего используется как основа для формального согласования между подразделениями и внешними контрагентами, поскольку она легче воспринимается участниками, которые не работают с диаграммами. Однако, при высокой динамике изменений SRS требует дисциплины сопровождения: если документ не обновляется синхронно с изменениями, он перестаёт быть «единым источником истины» и теряет управленческую ценность.

Сценарии использования (use case) [6] описывают цели акторов, основные и альтернативные сценарии, условия начала и конца. Они хорошо подходят для согласования ожиданий и для связи с приёмочными испытаниями. Там, где важны параллельные ветви и регламенты, одних прецедентов мало, их дополняют моделями процессов. Шаблоны Алистера Коберна и Карла Вигерса учитывают в явном виде параллельные ветви [7].

В данной статье под use case понимается структурированный сценарий взаимодействия актора с системой, оформленный по шаблону, где явно фиксируются: цель актора, основной поток событий, альтернативные потоки (включая исключения), пред- и постусловия, а также бизнес-правила и ограничения. Такой формат позволяет отделять «желательное поведение» от «допустимого поведения» и сразу задавать точки контроля для тестирования. При необходимости use case может быть дополнен диаграммой последовательности для уточнения сообщений между компонентами или диаграммой процесса BPMN – если требуется показать регламент и участие нескольких ролей/систем. Таким образом, use case выступает мостом между «пользовательской целью» и дальнейшей формализацией поведения.

Модели бизнес-процессов в нотации Business Process Model and Notation (BPMN) отражают роли, маршруты, ветвления и события. Нотация помогает согласовать регламенты, формализовать таймауты, компенсации и эскалации. Это особенно полезно, когда одна бизнес-цель достигается несколькими системами.

Диаграммы потоков данных (Data Flow Diagram, DFD) и контекст-диаграмма отвечают на вопрос, где возникают и куда уходят данные, через какие каналы и в каком направлении. Эти артефакты быстро выявляют недостающие источники и потребителей данных. Они не

показывают хронологию шагов, поэтому требуют дополнения поведенческими моделями.

Диаграммы последовательностей Unified Modeling Language (UML) фиксируют взаимодействия участников во времени. На таких диаграммах удобно согласовывать протоколы каскада вызовов, поведение при сбоях, повторы и идемпотентность. Они естественным образом связываются с контрактами интерфейсов.

В интеграционных сценариях диаграммы последовательностей часто становятся ключевым инструментом согласования «поведения при проблемах»: что происходит при таймауте, как выглядит повторная отправка, кто инициирует компенсацию, какие ответы считаются временными, а какие – финальными. Они также позволяют фиксировать ожидания по порядку вызовов и зависимостям (например, нельзя подтверждать операцию до получения определённого события). В результате команда получает не только описание интерфейса, но и согласованную модель взаимодействия, которая помогает тестированию (в том числе интеграционному и нагрузочному).

Диаграммы состояний UML описывают жизненный цикл ключевой сущности. Этот инструмент задаёт допустимые и запрещённые переходы и тем самым снижает количество ошибок бизнес-логики, связанных с некорректными статусами.

На практике диаграммы состояний особенно важны для сущностей, которые «живут долго» и участвуют в интеграции: заявки, платежи, заказы, обращения, талоны, результаты обследований и т.п. Если состояния не определены, то разные участники начинают трактовать статусы по-разному, что приводит к несогласованным действиям и конфликтам. Формализация жизненного цикла позволяет связывать каждое состояние с допустимыми операциями, правами доступа, условиями перехода и событиями интеграции (например, какие статусы отправляются наружу и какие – остаются внутренними).

Модель данных уровня сущностей и связей (ER-модель) и словарь данных создают основу для проверок целостности, отчётности и обмена. Эти артефакты фиксируют термины и форматы, поэтому их удобно связывать с требованиями к качеству данных.

Помимо структуры, данные имеют «качество»: полноту, корректность, уникальность, актуальность и согласован-

ность. Эти характеристики особенно критичны при обмене между системами, где разные источники могут по-разному называть одни и те же сущности или хранить разные форматы. Словарь данных выполняет роль общего языка: он фиксирует определения, допустимые значения, формат, единицы измерения и правила преобразования. Это снижает риск «тихих» ошибок, когда интеграция формально работает, но данные оказываются непригодными для отчётности или принятия решений.

Контракты интерфейсов Application Programming Interface (API) на основе OpenAPI или AsyncAPI задают структуру сообщений, правила версионирования, обязательные поля и коды ошибок. Они служат источником для генерации стабов и тестов и обеспечивают проверяемость интеграций.

Важно отметить, что контракт API – это не только описание полей, но и соглашение о поведении: семантика кодов ошибок, правила обработки частичных отказов, требования к идемпотентности, ограничения по частоте запросов, а также политика совместимости версий. Для событийных интеграций (AsyncAPI) дополнительно значимы вопросы гарантии доставки, порядок событий, дедупликация и обработка повторов. Таким образом, контракты выступают основой для технической «гарантии совместимости» между системами и позволяют автоматизировать значительную часть проверок за счёт контрактного тестирования.

Руководство SWEBOK [6] раскрывает область «Software Requirements» практическими разделами: обзоры требований, прототипирование, валидация моделей, приёмочные испытания. Отдельные параграфы посвящены атрибутам требований, трассировке и измерениям качества требований; даются ориентиры по инструментам. Это источник прикладных приёмов «как проверять требования на практике» и чем подкреплять требования тестами и метриками.

В рамках статьи SWEBOK используется как опорный источник для того, чтобы связать обсуждение артефактов с реальными практиками разработки: выявление требований, анализ и согласование, документирование, валидация и управление изменениями. Это важно, потому что требования – не статичный результат, а процесс: они уточняются по мере появления новых ограничений, обратной связи и интеграционных рисков. Следовательно, эффективность артефакта определяется тем, насколько он

поддерживает жизненный цикл требований и снижает издержки изменений.

Руководство BABOK [7] формализует анализ потребностей и артефакты бизнес-анализа: от выявления заинтересованных сторон и постановки целей до критериев приёмки. Материал задаёт терминологию и структуру работ, которые предшествуют и сопутствуют SRS: как формировать пользовательские истории, как фиксировать допущения и ограничения, как согласовывать ожидаемые результаты с бизнесом. Для нашей данной темы BABOK полезен тем, что объясняет, где уместны Use case и как их привязать к приёмочным сценариям.

Кроме того, BABOK делает акцент на управлении ожиданиями: фиксации допущений, ограничений и критериев успеха. Эти элементы часто воспринимаются как второстепенные, но именно они становятся причиной конфликтов, когда проект выходит на стадию внедрения или интеграции. В рамках выбранной темы это позволяет подчеркнуть: качественная инженерия требований включает не только «перечень функций», но и согласование того, что считается корректным результатом, кто и как его принимает, и какие показатели качества считаются достаточными.

Спецификация BPMN 2.0.2 [4] определяет нотацию процессов: роли, события, шлюзы, границы, а также расширения для ошибок, таймаутов и компенсаций. Документ подчёркивает двуединую цель: диаграммы должны быть понятны участникам процесса и при этом достаточно формальны, чтобы поддерживать автоматизацию. В контексте статьи BPMN – это средство описания маршрута процесса и исключительных ситуаций, особенно когда целевая бизнес-цель достигается через несколько систем.

Методика сравнительной оценки. Методика применяется для воспроизводимого сравнения способов фиксации требований и не зависит от предпочтений участников проекта. В оценку включаются спецификация требований (SRS), прецеденты, диаграммы BPMN, диаграммы последовательностей и состояний UML, DFD и контекстная диаграмма-диаграмма, ER-модель и словарь данных, контракты API по OpenAPI или AsyncAPI, а также пользовательские истории с критериями приёмки. Сравнение проводится для типовых задач: описание процесса, формулирование пользовательских целей, интеграция систем, данные и отчётность, жизненный цикл сущности. Квалификационный выбор по

задачам зафиксирован в матрице соответствий артефактам, представленной в таблице 1; именно она служит входом для количественной части.

При формировании матрицы соответствия учитывается, что один артефакт редко закрывает задачу полностью: чаще требуется связка, где разные представления дополняют друг друга. Поэтому матрица не рассматривает артефакты как взаимоисключающие, а показывает их «основную» и «вспомогательную» роль. Например, use case может задавать цель и сценарии, BPMN – регламент процесса, UML sequence – технический протокол взаимодействия, а контракт API – формат и правила ошибок. Такой взгляд позволяет не просто выбрать «лучший артефакт», а собрать минимально достаточную конфигурацию для конкретного типа проекта.

Количественная оценка опирается на шесть критериев, отражающих требования стандартов и практику эксплуатации: понятность для заинтересованных сторон, проверяемость и готовность к тестированию, полнота и непротиворечивость, трудоёмкость сопровождения, трассируемость к нефункциональным требованиям, пригодность к интеграциям. Применяется шкала от нуля до пяти, где ноль означает непригодность, три – приемлемость с оговорками, пять – полное соответствие назначению. Для фиксации приоритетов используются веса: для понятности и проверяемости – по 0,20; для полноты, трудоёмкости, трассируемости и пригодности к интеграциям – по 0,15; сумма равна единице.

Процедура включает подготовку по каждой задаче актуального пакета артефактов и их связей, независимую оценку двумя экспертами по указанной шкале, согласование расхождений и расчёт средневзвешенных значений по каждому артефакту в контексте задачи [8, 9, 10]. Итоги сводятся во вторую таблицу с числовыми результатами (таблица 2): в ней приведены баллы по критериям и итоговый средневзвешенный балл, что позволяет ранжировать артефакты и подтвердить выводы из таблицы 1 количественно. Если разрыв между лидером и ближайшим вариантом меньше 0,2 балла, оба решения считаются приемлемыми, окончательный выбор делается с учётом отраслевого контекста.

Отдельно подчёркивается, что методика ориентирована на воспроизводимость: при одинаковом наборе задач и одинаковых критериях разные эксперты должны приходить к близким

оценкам, а расхождения должны быть объяснимыми. Поэтому важной частью процедуры является фиксация аргументов к выставленным баллам (почему артефакт получил именно такую оценку по конкретному критерию). Это повышает прозрачность сравнения и превращает оценку не просто в «таблицу чисел», а в инструмент обоснования проектных решений.

После получения результатов проверяется принцип минимально достаточного набора. Если артефакт получает ниже трёх баллов по двум и более критериям, он исключается из базовой связки для соответствующей задачи, а его роль закрывается более сильным артефактом [11, 12]. Для проектов интеграции систем дополнительно проверяется полнота описания контекста и потоков данных,

**Таблица 1. Соответствие задач артефактам [составлено автором]**  
**Table 1. Matching tasks to artifacts [compiled by the author]**

Артефакт \ Задача	Описание процесса	Пользовательские цели	Интеграция	Жизненный цикл сущности	Данные и отчётность
Use Case	дополняющий	основной	избыточен	дополняющий	избыточен
BPMN	основной	дополняющий	дополняющий	избыточен	избыточен
DFD и контекст	дополняющий	избыточен	основной	избыточен	дополняющий
UML Sequence	дополняющий	дополняющий	основной	избыточен	избыточен
UML State	избыточен	избыточен	избыточен	основной	избыточен
ER и словарь	избыточен	избыточен	избыточен	дополняющий	основной
Stories и Gherkin	дополняющий	дополняющий	избыточен	избыточен	избыточен
OpenAPI и AsyncAPI	избыточен	избыточен	основной	избыточен	избыточен

наличие формального контракта API с версиями и кодами ошибок, отражение успешных и аварийных сценариев на диаграммах последовательностей с правилами повторов и идемпотентности, а при координации выполнения шагов между системами – корректность фрагмента BPMN и соответствие целевым уровням сервиса.

Для обеспечения качества сохраняются оценочные листы, версии артефактов и принятые веса критериев; повторная оценка выполняется после существенных изменений требований или архитектуры. Такой порядок обеспе-

чивает прозрачность сравнения и применимость рекомендаций при проектировании и сопровождении [13, 14, 15].

#### **Выводы.**

Матрица соответствий задачам и артефактам, представленная в таблице 1, показала устойчивую картину распределения ролей. Устойчивость картины означает, что артефакты проявляют себя предсказуемо в разных типах задач: одни лучше «держат» смысл и границы, другие – формализуют поведение, третьи – стабилизируют данные и интеграционный обмен. Это важно для практики, поскольку позволяет заранее

**Таблица 2. Оценка артефактов по критериям и итоговый балл [составлено автором]**  
**Table 2. Evaluation of artifacts by criteria and final score [compiled by the author]**

Артефакт	Понятность	Проверяемость	Полнота и согласованность	Сопровождение	Трассируемость к NFR	Пригодность к интеграциям	Итог (взвеш.)
SRS	4	4	5	4	4	3	4,00
Use Case	5	3	3	4	3	2	3,40
BPMN	4	4	4	3	3	4	3,70
DFD и контекст	4	3	3	5	3	5	3,80
UML Sequence	3	5	4	3	4	5	4,00
UML State	3	4	4	4	3	2	3,35
ER и словарь	3	3	5	4	4	3	3,60
Stories и Gherkin	4	4	3	4	3	2	3,40
OpenAPI и AsyncAPI	3	5	4	4	4	5	4,15

выбирать набор артефактов не по привычке команды, а по ожидаемому эффекту: снижению рисков, ускорению согласований и уменьшению числа дефектов на приёмке и в эксплуатации. Для описания процессов наилучший результат даёт BPMN как каркас маршрутизации, который дополняется прецедентами для фиксации целей и пользовательскими историями для проверяемости. Такая тройка работает как «слоёная структура»: BPMN задаёт маршрут и роли (кто и когда участвует), прецеденты фиксируют пользовательскую цель и варианты развития сценария (что считается успехом/неуспехом), а пользовательские истории с критериями приёмки обеспечивают проверяемость результата на уровне тестов и задач разработки. В результате команда получает одновременно понятную картину процесса и возможность однозначно подтвердить выполнение требований, не превращая описание в чрезмерно детализированный документ. Интеграционные сценарии требуют пакета из контекстной схемы и DFD, формальных контрактов API на основе OpenAPI или AsyncAPI и диаграмм последовательностей UML для согласования протоколов и поведения при сбоях; при наличии координации выполнения шагов между системами уместно добавить фрагмент BPMN. Задачи данных и отчётности опираются на ER-модель и словарь, тогда как жизненный цикл сущности надёжнее всего описывается диаграммой состояний UML.

Количественные оценки в таблице 2 подтвердили эти выводы цифрами. Контракты API и диаграммы последовательностей получили максимальные баллы по проверяемости и пригодности к интеграциям, DFD и контекстная диаграммная диаграмма – по сопровождению и интеграционной ясности, BPMN – по понятности и управляемости процесса. SRS удерживает высокий итоговый балл благодаря полноте и трассируемости к нефункциональным требованиям, но не заменяет поведенческие и интеграционные модели. Разрыв итоговых баллов между лидерами в пределах одной задачи невелик, что указывает на полезность комбинированных связей, однако, избыточные артефакты легко распознаются по низким оценкам сразу по нескольким критериям.

Практический эффект состоит в снижении риска несогласованности и избыточной документации. Применение принципа минимально достаточного набора позволяет исключить артефакты, которые не повышают проверяемость и не улучшают интеграции, и оставить то, что действительно поддерживает проектирование, тестирование и эксплуатацию. Связка артефактов должна быть привязана к измеримым целям по нефункциональным требованиям и соглашениям об уровне сервиса, чтобы поддерживать управляемость изменений и прозрачность на стыках систем.

Привязка к NFR/SLA позволяет сделать требования измеримыми: не просто «быстро», «надёжно» и «без ошибок», а с конкретными показателями времени ответа, доступности, допустимого процента ошибок, требований к восстановлению и правилами реагирования на инциденты. Для интеграций это особенно критично, потому что обмен может быть формально корректным, но непригодным по качеству сервиса (например, из-за частых таймаутов или нестабильной доставки сообщений). Поэтому методика предполагает согласование не только формата данных, но и ожидаемого уровня качества взаимодействия.

Предложенные решения согласуются с положениями международных стандартов и руководств по инженерии требований и жизненному циклу программного обеспечения, а также со спецификациями нотаций и описания интерфейсов. Это обеспечивает воспроизводимость подхода и даёт возможность применять его в проектах с разной отраслевой спецификой без наращивания «бумажного» объёма.

Итоговый вывод состоит в том, что предложенный подход формирует практическую «карту выбора» артефактов, которая применима как для новых проектов, так и для модернизации существующих систем. Он обеспечивает единый язык описания требований для разных ролей, повышает проверяемость и снижает риск интеграционных дефектов. Благодаря опоре на стандарты и спецификации подход может быть внедрён в организационные регламенты разработки и использоваться как основа для шаблонов документации и чек-листов контроля качества требований.

## Список источников

1. ISO/IEC/IEEE 29148:2018. Systems and software engineering – Life cycle processes – Requirements engineering. Geneva: ISO/IEC/IEEE, 2018. URL: <https://www.iso.org/standard/72089.html>
2. ISO/IEC/IEEE 12207:2017. Systems and software engineering – Software life cycle processes. Geneva: ISO/IEC/IEEE, 2017. URL: <https://www.iso.org/standard/63712.html>

3. ISO/IEC 25010:2011. Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models. Geneva: ISO/IEC, 2011. URL: <https://www.iso.org/standard/35733.html>
4. OMG. Business Process Model and Notation (BPMN) Version 2.0.2: Specification. Needham: Object Management Group, 2013. URL: <https://www.omg.org/spec/BPMN/2.0.2/About-BPMN>
5. OMG. Unified Modeling Language (UML) Version 2.5.1: Specification. Needham: Object Management Group, 2017. URL: <https://www.omg.org/spec/UML/2.5.1/About-UML>
6. IEEE Computer Society. Guide to the Software Engineering Body of Knowledge (SWEBOK V3.0). Los Alamitos: IEEE CS, 2014. 336 p. URL: <https://ieeecs-media.computer.org/media/education/swebok/swebok-v3.pdf>
7. IIBA. A Guide to the Business Analysis Body of Knowledge (BABOK® Guide), Version 3. Toronto: International Institute of Business Analysis, 2015. URL: <https://www.iiba.org/career-resources/a-business-analysis-professionals-foundation-for-success/babok/>
8. IREB. Certified Professional for Requirements Engineering (CPRE). Foundation Level Syllabus. Version 3.2. 2023. URL: [https://isqi.org/media/7f/9a/3e/1744288053/cpre\\_foundationlevel\\_syllabus\\_EN\\_v.3.2.pdf](https://isqi.org/media/7f/9a/3e/1744288053/cpre_foundationlevel_syllabus_EN_v.3.2.pdf)
9. OpenAPI Initiative. OpenAPI Specification. Version 3.1.0. 2021. URL: <https://spec.openapis.org/oas/v3.1.0.html>
10. AsyncAPI Initiative. AsyncAPI Specification (latest). 2025. URL: <https://www.asyncapi.com/docs/reference/specification/latest>
11. Cockburn A. Writing Effective Use Cases. Boston: Addison-Wesley, 2001. 270 p.
12. van Lamsweerde A. Requirements Engineering: From System Goals to UML Models to Software Specifications. Chichester: John Wiley & Sons, 2009. 682 p. URL: <https://www.wiley.com/en-nz/Requirements%2BEngineering%3A%2BFrom%2BSystem%2BGoals%2Bto%2BUML%2BModels%2Bto%2BSoftware%2BSpecifications-p-9780470012703>
13. DeMarco T. Structured Analysis and System Specification. Englewood Cliffs: Prentice-Hall, 1979. 352 p. URL: <https://archive.org/details/structuredanalysis0000dema>
14. ГОСТ 34.602–2020. Информационные технологии. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы. Введ. 01.01.2022. Москва: Росстандарт, 2022. URL: <https://protect.gost.ru/document1.aspx>
15. ГОСТ Р ИСО/МЭК 25010–2015. Информационные технологии. Системная и программная инженерия. Требования и оценка качества систем и программного обеспечения (SQuaRE). Модели качества систем и программных продуктов. Москва: Росстандарт, 2015. URL: <https://docs.cntd.ru/document/1200121069>

## References

1. ISO/IEC/IEEE (2018) ISO/IEC/IEEE 29148:2018. Systems and software engineering – Life cycle processes – Requirements engineering. Geneva: ISO/IEC/IEEE. Available at: <https://www.iso.org/standard/72089.html>
2. ISO/IEC/IEEE (2017) ISO/IEC/IEEE 12207:2017. Systems and software engineering – Software life cycle processes. Geneva: ISO/IEC/IEEE. Available at: <https://www.iso.org/standard/63712.html>
3. ISO/IEC (2011) ISO/IEC 25010:2011. Systems and software engineering – Systems and software quality requirements and evaluation (SQuaRE) – System and software quality models. Geneva: ISO/IEC. Available at: <https://www.iso.org/standard/35733.html>
4. Object Management Group (OMG) (2013) Business Process Model and Notation (BPMN) Version 2.0.2: Specification. Needham: Object Management Group. Available at: <https://www.omg.org/spec/BPMN/2.0.2/About-BPMN>
5. Object Management Group (OMG) (2017) Unified Modeling Language (UML) Version 2.5.1: Specification. Needham: Object Management Group. Available at: <https://www.omg.org/spec/UML/2.5.1/About-UML>
6. IEEE Computer Society (2014) Guide to the Software Engineering Body of Knowledge (SWEBOK V3.0). Los Alamitos: IEEE Computer Society, 336 p. Available at: <https://ieeecs-media.computer.org/media/education/swebok/swebok-v3.pdf>
7. International Institute of Business Analysis (IIBA) (2015) A Guide to the Business Analysis Body of Knowledge (BABOK® Guide), Version 3. Toronto: IIBA. Available at: <https://www.iiba.org/career-resources/a-business-analysis-professionals-foundation-for-success/babok/>
8. International Requirements Engineering Board (IREB) (2023) Certified Professional for Requirements Engineering (CPRE). Foundation Level Syllabus, Version 3.2. Available at: [https://isqi.org/media/7f/9a/3e/1744288053/cpre\\_foundationlevel\\_syllabus\\_EN\\_v.3.2.pdf](https://isqi.org/media/7f/9a/3e/1744288053/cpre_foundationlevel_syllabus_EN_v.3.2.pdf)
9. OpenAPI Initiative (2021) OpenAPI Specification, Version 3.1.0. Available at: <https://spec.openapis.org/oas/v3.1.0.html>
10. AsyncAPI Initiative (2025) AsyncAPI Specification (latest). Available at: <https://www.asyncapi.com/docs/reference/specification/latest>
11. Cockburn, A. (2001) Writing effective use cases. Boston: Addison-Wesley, 270 p.
12. van Lamsweerde, A. (2009) Requirements engineering: from system goals to UML models to software specifications. Chichester: John Wiley & Sons, 682 p. Available at: <https://www.wiley.com/en-nz/Requirements%2BEngineering%3A%2BFrom%2BSystem%2BGoals%2Bto%2BUML%2BModels%2Bto%2BSoftware%2BSpecifications-p-9780470012703>
13. DeMarco, T. (1979) Structured analysis and system specification. Englewood Cliffs: Prentice-Hall, 352 p. Available at: <https://archive.org/details/structuredanalysis0000dema>
14. Rosstandart (2022) GOST 34.602–2020. Information technology. System of standards for automated systems. Technical specification for the creation of an automated system. Moscow: Rosstandart. Available at: <https://protect.gost.ru/document1.aspx?control=31&id=241754> (In Russ.)
15. Rosstandart (2015) GOST R ISO/IEC 25010–2015. Information technology. Systems and software engineering. Systems and software quality requirements and evaluation (SQuaRE). Quality models for systems and software products. Moscow: Rosstandart. Available at: <https://docs.cntd.ru/document/1200121069> (In Russ.)

**Научный руководитель:**  
**Литвак Е.Г., доцент, канд. экон. наук**  
**кафедры информационных технологий,**  
**Донецкий филиал РАНХиГС**  
**Донецк, Донецкая Народная Республика,**  
**Российская Федерация**

*Автор заявляет об отсутствии конфликта интересов.*  
*The author declares no conflicts of interests.*

Поступила в редакцию (Reserved) 25.12.2025  
 Поступила после рецензирования 04.02.2026  
 Принята к публикации (Accepted) 10.02.2026