

УДК 004.05:519.81

DOI: [10.26102/2310-6018/2026.61.10.010](https://doi.org/10.26102/2310-6018/2026.61.10.010)

Архитектура и верификация системы поддержки принятия решений качества программных проектов в конвейере непрерывной интеграции

М.М. Пуговкин✉, А.А. Сидоров

*Томский государственный университет систем управления и радиоэлектроники,
Томск, Российская Федерация*

Резюме. Релизное решение – допустить версию программного продукта к выпуску, вернуть на доработку или перераспределить ресурсы контроля качества – является базовым актом управления в организации, разрабатывающей программное обеспечение. В рамках его воспроизводимости и систематичности механизм управления качеством должен быть реализован как автоматизированный сервис, встроенный в конвейер непрерывной интеграции и доставки. В работе представлена система поддержки принятия решений, реализующая механизм релизных решений на основе некомпенсируемой интегральной оценки качества и адресованная лицу, принимающему релизное решение – руководителю службы обеспечения качества или техническому руководителю проекта. Описаны трехслойная архитектура с отчуждаемым вычислительным ядром, реляционная модель данных из пяти сущностей, обеспечивающая прослеживаемость решений и сопоставимость оценок во времени, и управленческий контур: расчет интегральной оценки и субиндексов, трехзонное решающее правило релизного гейта, темпоральные триггеры деградации, рекомендации с разверткой до метрик и исполняемый гейт конвейера. Предложена стратегия верификации, в которой формальные свойства свертки оценки закрепляются как приемочные тесты; дискриминирующая способность этих тестов подтверждена негатив-контролем и мутационным тестированием, реализация воспроизводит указанные свойства при высоком покрытии кода, а вычислительное ядро показывает линейную сложность. Работоспособность контура продемонстрирована на реальном обезличенном портфеле проектов с прослеживанием деградации от метрик до управленческого решения.

Ключевые слова: поддержка принятия решений, управление качеством, интегральная оценка, релизный гейт, непрерывная интеграция, верификация программного обеспечения, неполная компенсация, организационная система.

Благодарности: Работа выполнена в рамках государственного задания Минобрнауки России; проект FEWM-2026-0011.

Для цитирования: Пуговкин М.М., Сидоров А.А. Архитектура и верификация системы поддержки принятия решений качества программных проектов в конвейере непрерывной интеграции. *Моделирование, оптимизация и информационные технологии*. 2026;14(10). URL: <https://moitvvt.ru/ru/journal/article?id=2542> DOI: 10.26102/2310-6018/2026.61.10.010

Architecture and verification of a decision support system for software project quality in a continuous integration pipeline

M.M. Pugovkin✉, A.A. Sidorov

*Tomsk State University of Control Systems and Radioelectronics, Tomsk,
the Russian Federation*

Abstract. A release decision – to admit a software version, return it for rework, or reallocate quality-control resources – is a basic management operation in a software development organization. For such decisions to be made reproducibly, the quality-management mechanism must be realized as an

automated service embedded in the continuous integration and delivery pipeline. This paper presents a decision support system that implements a release-decision mechanism based on a non-compensatory integral quality assessment and is addressed to the release decision maker: the head of the quality assurance service or the technical lead of the project. A three-layer architecture with a detachable computational core, a five-entity relational data model that makes decisions traceable and scores comparable over time, and the management loop are described: computation of the integral score and subindices, a three-zone release-gate decision rule, temporal degradation triggers, metric-level drill-down recommendations, and an executable pipeline gate. A verification strategy is proposed in which the formal properties of the assessment aggregation are fixed as acceptance tests; their non-vacuity is confirmed by negative control and mutation testing, the implementation reproduces these properties at high code coverage, and the core exhibits linear complexity. The loop is demonstrated on a real-world anonymized project portfolio, tracing degradation from metrics to a management decision.

Keywords: decision support, quality management, integral assessment, release gate, continuous integration, software verification, incomplete compensation, organizational system.

Acknowledgments: This research was funded by the Ministry of Science and Higher Education of the Russian Federation; project FEWM-2026-0011.

For citation: Pugovkin M.M., Sidorov A.A. Architecture and verification of a decision support system for software project quality in a continuous integration pipeline. *Modeling, Optimization and Information Technology*. 2026;14(10). (In Russ.). URL: <https://moitvvt.ru/ru/journal/article?id=2542> DOI: 10.26102/2310-6018/2026.61.10.010

Введение

Решение о выпуске очередной версии программного продукта остается одним из самых ответственных управленческих актов в производственно-технологическом цикле разработки. За ним стоит выбор среди нескольких альтернатив: передать сборку в эксплуатацию, вернуть ее на доработку или перераспределить ограниченные ресурсы службы обеспечения качества на наиболее уязвимые участки. Этот выбор делает лицо, принимающее решение (ЛПР), – руководитель, отвечающий за выпуск версии; в организациях-разработчиках эту роль исполняет руководитель службы обеспечения качества либо технический руководитель проекта. Цена ошибки несимметрична. Преждевременный выпуск дефектной версии оборачивается отказами в продуктивной среде, репутационными потерями и дорогостоящими экстренными исправлениями, тогда как неоправданная задержка замораживает поставку ценности и расходует ресурс команды впустую [1, 2]. Чтобы такие решения принимались осознанно и на основе данных, а не на базе разрозненных интуитивных суждений отдельных инженеров [3], организация нуждается в воспроизводимом механизме управления качеством, замыкающем контур «оценка состояния – решение о выпуске – управляющее воздействие – переоценка».

В современной инженерии программного обеспечения этот контур не может существовать вне инструментальной среды. Разработка ведется в конвейере непрерывной интеграции и доставки (CI/CD), где сборки появляются десятками в день, а данные о качестве рассредоточены по разнородным источникам [4, 5]. Поэтому механизм принятия релизных решений имеет смысл, лишь будучи реализованным как программное обеспечение, встроенное в этот конвейер: оно должно само собирать показатели, само вычислять оценку и само возвращать сигнал в точке, где решение фактически принимается.

Существующие инструментальные «ворота качества» (quality gate) такую функцию выполняют лишь частично. Промышленные решения, прежде всего Quality

Gate в SonarQube, по публичной документации продукта¹ задают набор независимых условий по отдельным метрикам и пропускают сборку, если ни одно из них не нарушено; близкую логику автономных порогов используют и инструменты управления техническим долгом, встраиваемые в конвейер [6, 7]. Каждое условие здесь срабатывает самостоятельно, а итогового заключения о готовности сборки как единой величины не формируется: набор раздельных порогов компенсаторен на уровне портфеля и тренда, поскольку благополучие по большинству условий заслоняет критический дефицит по одному из них, и целостная картина качества так и не складывается в число, на котором можно основывать ресурсные решения. Интегральные же оценки качества на основе агрегации разнородных программных метрик в литературе предложены [8], однако свойство неполной компенсации (провал по одной метрике не маскируется благополучием остальных) в них специально не выделяется, а сами они не доведены до развертываемой реализации в конвейере. Кроме того, и это существенно, они не верифицированы на соответствие исходному методу: нет гарантии, что работающая система вычисляет именно ту оценку, которая описана математически. В отличие от такой схемы предлагаемая система сводит показатели в единую некомпенсируемую интегральную оценку, дополняет ее некомпенсируемым порогом по каждой группе показателей (субиндексу) и трактует верификацию свойств метода как условие приемки самой реализации, тогда как у Quality Gate в SonarQube условия проверяются независимо друг от друга, и итоговая готовность сборки в виде одного числа не выводится. Ближайший методологический аналог предлагаемому решающему правилу – некомпенсируемые пороговые (outranking) модели многокритериального принятия решений [9]: они реализуют тот же отказ от полной компенсации через пороги несравнимости и согласия. Однако такие модели возвращают парные отношения превосходства и ранжирование альтернатив без развертываемой интеграции в конвейер CI/CD и без единой агрегированной величины оценки качества, пригодной для управления по тренду и распределения ресурсов службы качества; предлагаемый же гейт строит единую некомпенсируемую оценку, дополняет ее субиндексным порогом и исполняемым сигналом в конвейер и закрепляет свойства метода как условие приемки реализации. Поэтому задача развертываемого релизного гейта на основе некомпенсируемой интегральной оценки прямых аналогов среди рассмотренных подходов не имеет.

Отсюда формируется исследовательский вопрос настоящей работы: как реализовать механизм релизных решений на основе некомпенсируемой интегральной оценки качества в виде развертываемой системы поддержки принятия решений (СППР), адресованной ЛПР и встроенной в конвейер CI/CD, обеспечив воспроизводимость оценок, их верифицируемость относительно метода и пригодность для автоматического управления выпуском?

Цель работы – доведение механизма поддержки принятия релизных решений на основе некомпенсируемой интегральной оценки качества до верифицированной программной реализации, встроенной в конвейер CI/CD. Достижение цели требует решения четырех задач: сформулировать требования к программному обеспечению механизма, вытекающие из управленческой природы задачи, и предложить отвечающую им архитектуру; реализовать управленческий контур – от вычисления оценки и трехзонного решающего правила до темпоральных триггеров, адресных рекомендаций и исполняемого сигнала в конвейер; верифицировать соответствие реализации методу, подтвердив дискриминирующую способность самих проверок негатив-контролем и

¹ *Understanding quality gates*. Sonar documentation. URL: <https://docs.sonarsource.com/sonarqube-server/quality-standards-administration/managing-quality-gates/introduction-to-quality-gates> (дата обращения: 17.09.2026).

мутационным анализом; продемонстрировать работу полного контура на реальном портфеле программных проектов.

Вклад работы, сформулированный в управленческой парадигме, состоит в следующем.

- Релизное решение принимается по единой некомпенсируемой оценке, воспроизводимой при тех же входных данных, а не по набору автономных условий, как в промышленных воротах качества¹: двенадцать метрик собираются из инструментов разработки и эксплуатации без участия человека, и ЛПР получает одну величину вместо разрозненных показателей. Подтверждается описанием контура в разделе «Материалы и методы» и демонстрацией в разделе «Результаты».

- Решающее правило релизного гейта реализовано как трехзонная классификация с субиндексным порогом: в отличие от пороговых моделей многокритериального выбора [9], возвращающих отношения превосходства между альтернативами, правило выдает исполняемое решение по единой величине, а критический дефицит по любой группе показателей становится ранним управляющим сигналом, а не растворяется в усредненной оценке. Подтверждается негатив-контролем и сквозным кейсом деградации в разделе «Результаты».

- Контур замкнут темпоральными триггерами деградации и приоритизированными рекомендациями с разверткой до конкретных метрик, что переводит оценку в адресное распределение ресурсов службы качества; известные интегральные оценки программных метрик [8] до такого контура не доведены. Подтверждается демонстрацией на реальном портфеле в разделе «Результаты»: 226 алертов и рекомендации по проблемному проекту.

- Обеспечена темпоральная сопоставимость ряда оценок за счет фиксации весов и параметров нормализации на калибровочном окне, тогда как при свободном пересчете энтропийных весов [10, 11] ряд оценок несопоставим и управление по тренду невозможно. Подтверждается моделью данных в разделе «Материалы и методы» и траекториями за пятьдесят две сборки в разделе «Результаты».

- Свойства метода закреплены как приемочные тесты реализации, а дискриминирующая способность самих тестов измерена негатив-контролем и мутационным анализом; промышленные ворота качества верифицируют подсчет метрик, но не свойства решающего правила. Подтверждается итогами верификации в разделе «Результаты».

- Механизм оформлен как исполняемый гейт CI/CD [4, 5], останавливающий сборку при критической деградации, и встроен непосредственно в точку принятия решения. Подтверждается описанием команды расчета и демонстрацией срабатывания в разделе «Результаты».

Материалы и методы

Прежде чем описывать устройство системы, стоит зафиксировать, для кого она создается и каким требованиям должно отвечать программное обеспечение, претендующее на роль механизма поддержки принятия релизных решений, а не просто очередного отчетного инструмента. Адресат системы – ЛПР: руководитель службы обеспечения качества или технический руководитель проекта; именно он принимает решение о выпуске версии и распределяет ресурс контроля качества. Вторая группа пользователей – инженеры службы качества, исполняющие адресные рекомендации системы, а конвейер CI/CD выступает потребителем ее исполняемого сигнала. Требования к системе вытекают из управленческой природы задачи. Решение принимается часто и в темпе конвейера, поэтому показатели должны извлекаться из

инструментов разработки и эксплуатации без участия человека (автоматический сбор). При одних и тех же входных данных и параметрах система обязана давать тождественный результат (воспроизводимость), иначе оценка не может служить основанием управленческого решения. Должна существовать возможность проверить, что реализация вычисляет именно описанную математически оценку (верифицируемость относительно метода). Механизм встраивается в конвейер и возвращает в него управляющий сигнал (интеграция в CI/CD). Наконец, оценки разных моментов времени должны быть сравнимы между собой (темпоральная сопоставимость), иначе управление по тренду теряет смысл.

Интегральная оценка складывается из двенадцати метрик, распределенных по трем группам в соответствии с моделями качества программного обеспечения^{2,3} [12, 13] (Таблица 1). Внутреннее качество кода характеризуют цикломатическая сложность (CC), коэффициент технического долга (TD), дублирование кода (Dup) и нарушения стандартов кодирования (Viol), собираемые из статического анализатора [14]; характеристики процесса – покрытие тестами (Cov), плотность дефектов (DD), среднее время восстановления (MTTR) и доля успешных сборок (BSR), поступающие из системы управления задачами и конвейера; внешнее качество – доступность (Avail), время отклика (RT), частота инцидентов после релиза (PIR) и удовлетворенность пользователей (USS) из системы мониторинга [15]. Метрики различаются направленностью: у выгодных рост значения означает улучшение качества, у затратных – ухудшение, поэтому затратные метрики обращаются при нормализации. Разнородные метрики приводятся к сопоставимому виду нормализацией (по умолчанию сигмоидальное преобразование либо адаптивный выбор по статистике данных) [16]: сигмоидальное преобразование не требует знания границ шкалы и устойчиво к единичным выбросам, тогда как минимаксная нормализация растягивает шкалу по экстремальным значениям. Веса назначаются объективно, по энтропийному принципу Шеннона, на двух уровнях – внутри групп и между группами [10, 11]; энтропийные веса предпочтены экспертным как воспроизводимые по данным и не требующие согласования суждений.

Таблица 1 – Состав метрик интегральной оценки

Table 1 – Composition of the integral-assessment metrics

Группа (индекс <i>g</i>) и источник данных	Метрика	Обозначение	Направленность
Внутреннее качество кода (int), статический анализатор	циклматическая сложность	CC	затратная
	коэффициент технического долга	TD	затратная
	дублирование кода	Dup	затратная
	нарушения стандартов кодирования	Viol	затратная
Характеристики процесса (proc), система управления задачами и конвейер	покрытие тестами	Cov	выгодная
	плотность дефектов	DD	затратная

² 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: Стандартинформ; 2015. 29 с.

³ ISO/IEC 25010:2023. Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Product quality model. Geneva: ISO; 2023. 22 p.

Таблица 1 (продолжение)
Table 1 (continued)

	среднее время восстановления	MTTR	затратная
	доля успешных сборок	BSR	выгодная
Внешнее качество (ext), система мониторинга	доступность	Avail	выгодная
	время отклика	RT	затратная
	частота инцидентов после релиза	PIR	затратная
	удовлетворенность пользователей	USS	выгодная

Свертка выполняется вложенными функциями с постоянной эластичностью замещения (CES) [17]; сама модель оценки – двухуровневая свертка энтропийно взвешенных метрик – разработана авторами ранее и воспроизводится здесь в объеме, необходимом для описания реализации. Субиндекс группы вычисляется по формуле:

$$I_g = (\sum_j \hat{w}_j (n_j + \varepsilon)^{\alpha_g})^{1/\alpha_g}, g \in \{int, proc, ext\}, \quad (1)$$

где I_g – субиндекс группы g ; $g \in \{int, proc, ext\}$ – группы внутреннего качества кода, характеристик процесса и внешнего качества; j – индекс метрики внутри группы; $n_j \in (0; 1)$ – нормализованное значение j -й метрики; $\hat{w}_j$ – энтропийный вес метрики внутри группы (веса метрик группы нормированы к единице); α_g – параметр замещения группы (эластичность замещения равна $1/(1 - \alpha_g)$, так что меньшим α_g отвечает менее компенсаторная свертка); $\varepsilon = 10^{-3}$ – стабилизирующая константа, которая сдвигает аргумент степенной свертки от нуля, исключает вырождение при нулевых нормализованных значениях и удерживает субиндекс строго положительным. Интегральная оценка получается сверткой субиндексов тем же оператором:

$$Q = (\sum_g \hat{w}_g I_g^{\alpha_{top}})^{1/\alpha_{top}}, Q \in (0, 1 + \varepsilon], \quad (2)$$

где Q – интегральная оценка качества проекта; $\hat{w}_g$ – энтропийный вес группы (веса групп нормированы к единице); α_{top} – параметр эластичности верхнего уровня. При нормализации показателей в интервал $(0; 1)$ итоговая оценка строго положительна и не превосходит $1 + \varepsilon$: стабилизирующая константа может вывести ее за единицу не более чем на 10^{-3} , что для решающего правила несущественно. Параметры эластичности по умолчанию равны $\alpha_{int} = 0,4$, $\alpha_{proc} = 0,3$, $\alpha_{ext} = 0,8$, $\alpha_{top} = 0,5$. Эластичность задает меру взаимозаменяемости показателей внутри свертки: чем она ниже, тем строже некомпенсируемость, поэтому для внутреннего качества кода и характеристик процесса, где провал по любому показателю критичен, выбраны более низкие значения, а для внешнего качества, метрики которого естественнее замещают друг друга при наблюдении за продуктом, эластичность выше. Допустимый домен эластичности – открытый интервал $\alpha \in (0; 1)$: предел $\alpha \rightarrow 1$ отвечает вырожденно-линейной (полностью компенсаторной) свертке и потому исключается, тогда как реализация ядра принимает и граничное значение $\alpha = 1$, трактуя его как этот линейный случай: аддитивная свертка нужна как эталон сравнения в приемочных тестах неполной компенсации. Приведенные значения α и ε , как и веса, являются значениями по умолчанию и подлежат настройке организацией под ее политику выпуска; ядро выносит их в параметры расчета, а каждый прогон фиксирует использованные параметры. Ключевое управленческое свойство такой свертки – неполная компенсация: критический дефицит по одной метрике роняет субиндекс группы и итоговую оценку, не маскируясь высокими значениями остальных

показателей. Именно это свойство и делает оценку пригодной для раннего сигнала о неготовности к выпуску.

Система построена как три слоя с однонаправленными зависимостями, что изолирует управленческую логику от деталей сбора данных и представления (Таблица 2, Рисунок 1) [18]. Верхний слой представления образуют командный интерфейс (CLI), удобный для встраивания в конвейер, и веб-дашборд для наблюдения за состоянием проектов. Прикладной слой *sqa_dss* несет управленческий контур: импорт данных из инструментов, сервисы расчета и принятия решений, персистентность и единый фасад контура. Нижний слой – вычислительное ядро *sqa_core*, выполняющее нормализацию, расчет энтропийных весов и двухуровневую CES-агрегацию. Зависимости направлены только вниз: представление обращается к прикладному слою, прикладной слой – к ядру, обратных связей нет.

Таблица 2 – Слои и компоненты СППР
Table 2 – Layers and components of the DSS

Слой	Компоненты	Назначение
Представление	CLI, веб-дашборд	точка принятия решения в конвейере, наблюдение за состоянием
Прикладной (<i>sqa_dss</i>)	импорт данных, сервисы контура, персистентность, фасад	управленческий контур: оценка, решение, триггеры, рекомендации
Ядро (<i>sqa_core</i>)	нормализация, энтропийные веса, CES-агрегация	воспроизводимое вычисление оценки Q и субиндексов I_g

Вычислительное ядро намеренно выделено в самостоятельный, отчуждаемый компонент. Оно не знает ни об источниках данных, ни о способе их хранения, ни о форме представления результата, и предоставляет управленческому слою лаконичный интерфейс модели качества с операциями настройки весов по данным [19], расчета оценки и упорядочения проектов [20, 21]. Такое разделение служит требованию верифицируемости: ядро можно протестировать и зарегистрировать как отдельный объект, проверив соответствие его выходов методу независимо от обвязки.

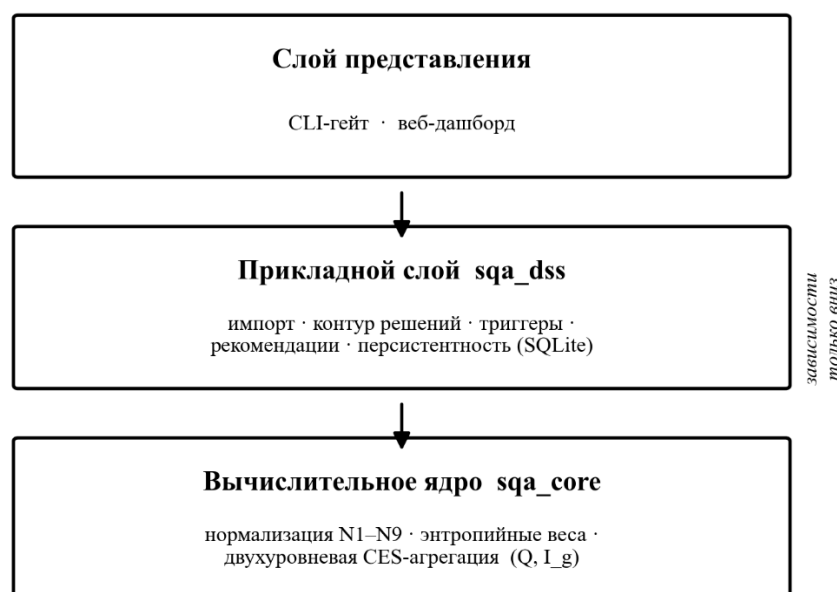


Рисунок 1 – Трехслойная архитектура СППР с однонаправленными зависимостями
Figure 1 – Three-layer architecture of the DSS with unidirectional dependencies

Персистентность реализована реляционной схемой на встраиваемой СУБД SQLite из пяти сущностей (Рисунок 2), спроектированной так, чтобы каждое релизное решение было прослеживаемо до входных данных и параметров, по которым оно получено. Сущность `projects` хранит перечень оцениваемых единиц портфеля; ее ключ – идентификатор проекта. Центральная сущность `scoring_runs` описывает каждый расчет: момент выполнения, имя и полный набор параметров конфигурации, способ нормализации, зафиксированные веса трех уровней (внутри групп, между группами и итоговые) и ссылку `baseline_run_id` на опорный расчет, нормализация и веса которого используются в режиме замороженных весов. Веса и параметры прогона не пересчитываются ретроспективно, что и обеспечивает сравнимость оценок во времени. Три сущности подчинены расчету по внешнему ключу `run_id` и привязаны к проекту по его идентификатору: `score_results` содержит для каждого проекта оценку Q , ранг, цветовую зону готовности, субиндексы I_g , нормализованные значения и журнал вкладов метрик; `snapshots` сохраняет входные значения двенадцати метрик вместе с источником и временной меткой для аудита и повторного расчета; `alerts` фиксирует сработавшие триггеры с указанием правила, области срабатывания (оценка, субиндекс или метрика), важности, значений до и после и величины падения. Структуры переменной размерности – веса, субиндексы, нормализованные значения, вклады и снимки метрик – хранятся в полях формата JSON, поэтому схема не меняется при изменении состава метрик. Связи «один ко многим» от расчета к результатам, снимкам и алертам образуют полный след решения: от снимка метрик через параметры и веса расчета к оценке, зоне готовности и сработавшим триггерам.

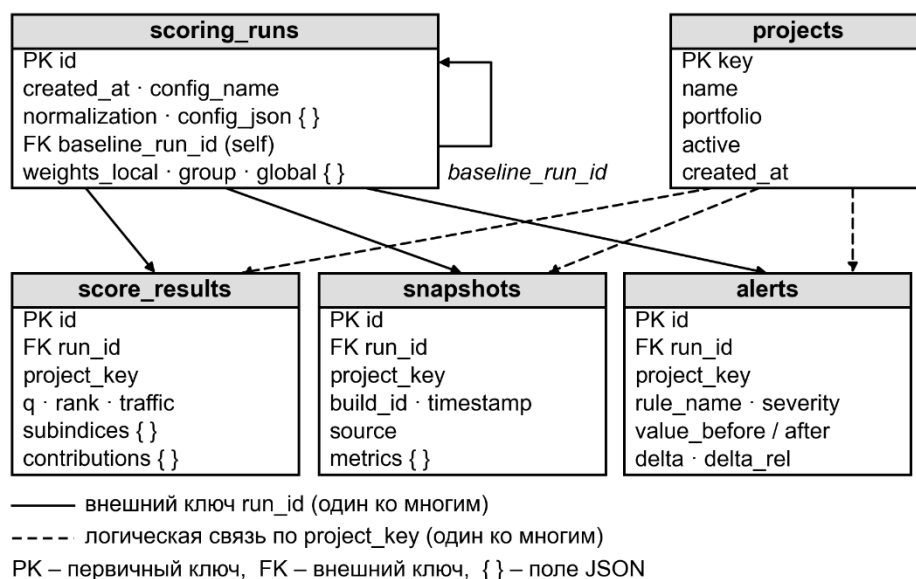


Рисунок 2 – Реляционная модель данных СПИР: сущности, ключи и связи

Figure 2 – Relational data model of the DSS: entities, keys and relationships

Управленческий контур превращает вычисленную оценку в последовательность управленческих операций: классификацию готовности к выпуску, обнаружение деградации, адресные рекомендации и исполняемый сигнал в конвейер. Расчет начинается с того, что прикладной слой собирает по проекту значения двенадцати метрик и передает их ядру. Ядро нормализует показатели, вычисляет энтропийные веса либо использует зафиксированные веса калибровочного окна, сворачивает нормализованные значения в субиндексы I_g , а субиндексы – в интегральную оценку Q .

Вместе с оценкой возвращается журнал вкладов: для каждой метрики фиксируется, насколько она приблизила или отдалила проект от эталона.

Решение вырабатывает не сама оценка, а решающее правило релизного гейта, применяемое к ней: оно реализовано как трехзонная цветовая классификация готовности (зеленая, желтая и красная зоны). Красная зона назначается, если $Q < \theta_Q$ или хотя бы один субиндекс $I_g < \theta_g$, то есть либо интегральная оценка ниже допустимого порога, либо хотя бы один субиндекс группы провалился ниже своего порога. Второе условие – прямая эксплуатация неполной компенсации: критический дефицит целой грани качества блокирует выпуск даже при высокой общей оценке, выступая ранним сигналом. Зеленая зона назначается, если $Q \geq \theta_Q^*$ и при этом ни один субиндекс не критичен, где θ_Q^* – порог уверенного выпуска. Все промежуточные случаи относятся к желтой зоне, требующей внимания, но не блокирующей выпуск: решение по ней принимает ЛПР, опираясь на предъявленные системой оценку, субиндексы и рекомендации, – в этом и состоит поддержка решения, а не его подмена.

Пороги по умолчанию равны $\theta_Q = 0,35$, $\theta_g = 0,30$ и $\theta_Q^* = 0,6$; это значения по умолчанию, подлежащие калибровке организацией по двум каналам – экспертному (согласованное суждение службы качества о допустимом уровне риска) и ретроспективному (настройка на калибровочном окне прошлых релизов с известным исходом). Сами уровни заданы из содержательных соображений. Порог блокировки $\theta_Q = 0,35$ размещен в нижней трети шкалы $[0; 1]$: красная зона отсекает лишь отчетливо неблагоприятные релизы, а не любое отклонение от идеала. Субиндексный порог $\theta_g = 0,30$ взят ниже θ_Q , поскольку относится к отдельной грани качества, а не к интегральной оценке: он перехватывает уже глубокий провал одной группы метрик, не штрафуя за умеренную неравномерность. Порог уверенного выпуска $\theta_Q^* = 0,6$ поднят заметно выше θ_Q , чтобы между ними оставалась буферная желтая зона требующих внимания релизов, и проект попадал в зеленую зону лишь при ощутимом запасе по интегральной оценке. Все три порога вынесены в настройки, поскольку приемлемый уровень риска определяется организацией, а не зашит в механизм.

Цветовая классификация оценивает мгновенное состояние, тогда как управление выпуском требует и реакции на динамику. Поэтому контур дополнен темпоральными триггерами, сравнивающими текущий прогон с предыдущим. Падение интегральной оценки на 10 % и более относительно прошлого прогона порождает алерт высокой важности; просадка любого субиндекса I_g на 15 % и более порождает алерт средней важности. Пороги срабатывания триггеров заданы по умолчанию и подлежат калибровке организацией на калибровочном окне прошлых релизов – так же, как пороги цветовой классификации. Их относительные уровни выбраны из содержательных соображений. Порог по интегральной оценке (10 %) взят строже, поскольку Q усредняет все грани качества и потому устойчивее: даже умеренное относительное падение свернутой оценки отражает заметное общее ухудшение. Порог по субиндексу (15 %) поднят выше, так как субиндекс относится к отдельной грани качества и опирается на меньшее число метрик, а потому обладает большей естественной волатильностью от прогона к прогону; более высокий порог отсекает шумовые колебания одной группы, оставляя триггер реагирующим на действительно значимую просадку. Так система отлавливает ухудшение еще до того, как оценка пробьет абсолютный порог красной зоны, давая ЛПР время на упреждающее вмешательство.

Сигнал о неготовности полезен ровно настолько, насколько он указывает, куда направить усилия. Поэтому контур формирует приоритизированные рекомендации, разворачивающиеся от агрегированной оценки до конкретных метрик. Приоритет c_j метрики j определяется по формуле (3) как ее вклад в отставание проекта от эталона:

$$c_j = \hat{w}_j^{glob} \cdot \max(0, 1 - n_j), \quad (3)$$

где c_j – приоритет (вклад в отставание) j -й метрики; $\hat{w}_j^{glob}$ – итоговый вес метрики в общей свертке, равный произведению ее веса внутри группы $\hat{w}_j$ на вес группы $\hat{w}_g$; n_j – нормализованное значение метрики, так что множитель $\max(0, 1 - n_j)$ есть величина ее отставания от эталона. Метрики ранжируются по этому вкладу, и наибольшие из них предъявляются как первоочередные точки приложения ресурсов службы качества.

Энтропийные веса зависят от статистики данных, поэтому при свободном пересчете они менялись бы от прогона к прогону, и ряд оценок $Q(t)$ перестал бы быть сравнимым: изменение оценки нельзя было бы отличить от изменения линейки измерения. Чтобы управление по тренду оставалось корректным, предусмотрен режим замороженных весов и параметров нормализации. Веса и настройки нормализации вычисляются один раз на калибровочном окне и далее фиксируются; последующие прогоны оцениваются по этой замороженной линейке. Тогда динамика $Q(t)$ отражает именно изменение качества, а не дрейф методики.

Завершающее звено возвращает управляющий сигнал туда, где принимается решение. Команда расчета в командном интерфейсе устроена как исполняемый гейт: при попадании проекта в красную зону по причине критической деградации она завершается с ненулевым кодом выхода. Конвейер CI/CD трактует ненулевой код как провал шага и останавливает сборку, не пропуская дефектную версию дальше [4, 5]. Так управленческое решение исполняется автоматически, без ручного посредника, прямо в точке выпуска. Перечисленные звенья соединены в единый процесс приема и обработки данных: при поступлении новых метрик система выполняет оценку, классифицирует результат по цветовым зонам, сохраняет прогон с зафиксированными весами, проверяет триггеры относительно предыдущего прогона и относительно калибровочного окна, после чего формирует приоритизированные рекомендации. Замкнутая последовательность «оценка – цветовая классификация – сохранение прогона – проверка триггеров – рекомендации» и образует тот контур управления качеством, программной реализацией которого является описанная СППР.

Верификация реализации выстроена вокруг одного принципа: формальные инварианты, которые вытекают непосредственно из определения CES-свертки (формулы (1)–(2)), рассматриваются не как пояснение к коду, а как критерий его приемки. Реализация считается корректной только в том случае, если она воспроизводит эти инварианты автоматически и воспроизводимо. Такой перенос математического свойства метода в приемочный (регрессионный) тест закрепляет за каждым содержательным утверждением исполняемую проверку, которая срабатывает при любом изменении кода. Подход опирается на сложившуюся практику верификации программного обеспечения [22] и на технику property-based testing, реализованную средствами библиотеки hypothesis [23], для контроля инвариантов на случайных входных данных.

Методика включает пять взаимодополняющих инструментов. Приемочные регрессии закрепляют три инварианта CES-свертки, выводимые из формул (1)–(2): инвариантность оценки к эквивалентным линейным преобразованиям шкалы метрик, согласованность ранжирования на синтетическом тест-стенде из двенадцати проектов с эталоном и антиэталом (набор зафиксирован в тестовых фикстурах ядра и воспроизводится детерминированно) и собственно неполную компенсацию – более сильный, чем у аддитивной свертки при тех же весах, сдвиг ранга при деградации отдельной метрики. Property-based проверки контролируют те же свойства не на единичных примерах, а на сгенерированных случайных матрицах метрик. Дискриминирующая способность набора подтверждается негатив-контролем: в код ядра вносятся заведомо неверные изменения, и фиксируется, приводят ли они к падению

приемки. Мутационное тестирование [24] дополняет негатив-контроль систематически: операторные мутанты – замены арифметических и сравнительных операций – автоматически вносятся в модули агрегации и взвешивания, после чего на каждом мутанте прогоняется весь набор тестов. Наконец, вычислительная пригодность ядра для работы в темпе конвейера оценивается микробенчмарком конвейера расчета с сигмоидальной нормализацией при $m = 12$ метриках на портфелях от ста до пятидесяти тысяч проектов. Замеры выполнены на рабочей станции с процессором Apple M4 (10 ядер) и 32 ГБ оперативной памяти под управлением macOS 27.0 в окружении Python 3.13.7 и NumPy 2.4.4; для каждого размера портфеля после прогрева фиксируется минимальное из семи повторов время полного конвейера расчета (нормализация, веса, свертка), измеренное монотонным таймером высокого разрешения.

Работа полного контура демонстрируется на реальном портфеле организации – маркетинговой платформы, для которой доступна история автоматически собираемых метрик качества: четырнадцать программных проектов, прослеженных на протяжении пятидесяти двух еженедельных сборок, – всего 728 снимков метрик. Веса агрегации заморожены по первой сборке (режим опорной базы), что обеспечивает темпоральную сопоставимость оценок на всем горизонте наблюдения; пороги решающего правила взяты по умолчанию. Приводимые далее результаты получены в окружении Python 3.13.7 с библиотеками numpy, scipy и pandas для численных расчетов и pytest с hypothesis для приемочного и свойство-ориентированного тестирования. Машиночитаемый портфель и конфигурация прогона предоставляются вместе с исходным кодом ядра `sqa_core` и прикладного слоя `sqa_dss` (по запросу), что обеспечивает воспроизводимость приводимых чисел и траекторий.

Результаты

Результаты приводятся в двух частях: сначала итоги верификации реализации относительно метода, затем сквозная демонстрация управленческого контура на реальном портфеле проектов.

Ядро `sqa_core`, реализующее расчет субиндексов I_g , нормализацию метрик n_j , оценку весов $\hat{w}_j$ и $\hat{w}_g$ и свертку в интегральную оценку Q , покрыто 101 автоматическим тестом. Все тесты проходят без единого падения. Покрытие кода составляет 99 % при 524 операторах, что для вычислительного ядра механизма принятия решений означает практически полное отсутствие непроверяемых ветвей. Дополнительно ядро снабжено строгой статической типизацией (статический анализ типов в строгом режиме дает ноль ошибок), линтером, строгим реестром маркеров и набором property-based тестов на инварианты, которые проверяют сохранение свойств оценки не на единичных примерах, а на сгенерированных случайных матрицах метрик.

Приемочные регрессии воспроизводят все три инварианта CES-свертки, выводимые из формул (1)–(2) (Таблица 3; в ней приведены критерии приемки, все они выполнены). Во-первых, инвариантность оценки к эквивалентным линейным преобразованиям шкалы метрик: поскольку такие преобразования отличаются лишь масштабом и сдвигом, при согласованных весах они обязаны давать тождественный порядок проектов. На фиксированной выборке расхождение значений не превышает $1 \cdot 10^{-12}$, а на случайных матрицах метрик property-based проверка подтверждает совпадение с относительной погрешностью $1 \cdot 10^{-6}$. Во-вторых, согласованность ранжирования на тест-стенде: на синтетическом наборе из двенадцати проектов, включающем эталонный проект с высокими значениями всех метрик и антиэталонный с равномерно низкими, эталон устойчиво получает ранг 1, антиэталон – ранг 12, а состав первой шестерки воспроизводится точно. Ранговая корреляция Спирмена с эталонным

порядком, заданным самой конструкцией тест-стенда (от эталонного проекта к антиэталонному), а не внешней экспертизой, составляет $\rho \geq 0,95$. В-третьих, свойство неполной компенсации, которое следует из эластичности замещения $\alpha < 1$ в формулах (1)–(2): деградация доступности лучшего проекта (Avail с 99,99 % до 90,0 %) при CES-свертке сдвигает его ранг не менее чем на 5 позиций, тогда как аддитивная свертка при тех же весах смещает его не более чем на 3 позиции. Веса здесь намеренно уравнены, чтобы изолировать эффект самого оператора свертки.

Таблица 3 – Инварианты CES-свертки как приемочные проверки
Table 3 – Invariants of the CES aggregation as acceptance checks

Тип проверки	Что подтверждается	Критерий приемки (выполнен)
Инвариантность к линейной нормализации	эквивалентные линейные шкалы дают согласованный порядок	расхождение $1 \cdot 10^{-12}$ (фикс. выборка), $1 \cdot 10^{-6}$ (property-based)
Согласованность ранжирования	устойчивость порядка проектов на тест-стенде	эталон ранг 1, антиэталон ранг 12, $\rho \geq 0,95$
Неполная компенсация	CES-свертка штрафует провал сильнее аддитивной	сдвиг ранга ≥ 5 против ≤ 3 при равных весах

Те же свойства воспроизводятся и на уровне прикладного слоя `sqa_dss`, который связывает ядро с хранилищем, сигнализацией и интерфейсом. Слой покрыт 122 автоматическими тестами (ноль падений, покрытие 98 %, строгий анализ типов без ошибок). Часть из них – сквозные регрессии, повторяющие перечисленные свойства уже через весь контур, а не только через ядро. Тем самым подтверждается, что управленческие выводы, которые получает ЛППР на выходе системы, опираются на тот же проверенный расчет, что и ядро.

Дискриминирующую способность приемки проверил негатив-контроль. Замена мультипликативной CES-свертки на аддитивную немедленно приводит к падению приемочных тестов неполной компенсации (`test_multiplicative_shifts_more_than_additive`, `test_multiplicative_q_drop_is_larger`), поскольку аддитивный оператор компенсирует провал по одной метрике и не сдвигает ранг на требуемые позиции. Нарушение целостности CES-ядра (подмена эластичной свертки аддитивной) приводит к падению и тестов предельного поведения CES. Тем самым упомянутые свойства закрепились не декларативно: при их нарушении в коде приемка падает.

Мутационное тестирование ядра дало следующий результат: из 24 операторных мутантов, внесенных в модули агрегации и взвешивания, тестами выявлены 17, то есть доля выявленных мутантов составила 70,8 % (при $n = 24$ 95-процентный интервал Уилсона широк, от 51 до 85 %, поэтому оценка характеризует порядок величины, а не точное значение). Семь выживших мутантов относятся к эквивалентным или защитным ветвям – в частности, к стражу нормировки весов (условие $\sum w > 0$), нулевая ветвь которого недостижима на корректных входных данных и потому не может быть выявлена содержательным тестом. Эта оценка ограничивает долю невыявляемых дефектов и характеризует фактическую разрешающую способность набора проверок.

Вычислительная пригодность ядра проверена микробенчмарком: сложность конвейера расчета по числу проектов n линейна, $O(n \cdot m)$ (Таблица 4, Рисунок 3). Время растет пропорционально объему портфеля: при $n = 1000$ расчет занимает 0,56 мс, при $n = 10000$ – 5,11 мс, при $n = 50000$ – 27,0 мс. Удельное время на проект остается практически постоянным на всем диапазоне (0,00051–0,00056 мс), что и подтверждает линейный режим; заметная на малых портфелях постоянная составляющая

инициализации с ростом n становится пренебрежимой. Такая производительность означает, что пересчет оценок крупного портфеля укладывается в доли секунды и не становится узким местом непрерывного мониторинга. В Таблице 4 приведены три характерные точки, полный ряд из шести размеров портфеля показан на Рисунке 3. Конфигурация ЭВМ и методика замера приведены в разделе «Материалы и методы»; на другом оборудовании абсолютные значения будут иными, тогда как линейный характер зависимости от оборудования не зависит.

Таблица 4 – Время расчета в зависимости от размера портфеля

Table 4 – Computation time versus portfolio size

Число проектов n	Время расчета	Удельное время на проект
1 000	0,56 мс	0,00056 мс
10 000	5,11 мс	0,00051 мс
50 000	27,0 мс	0,00054 мс

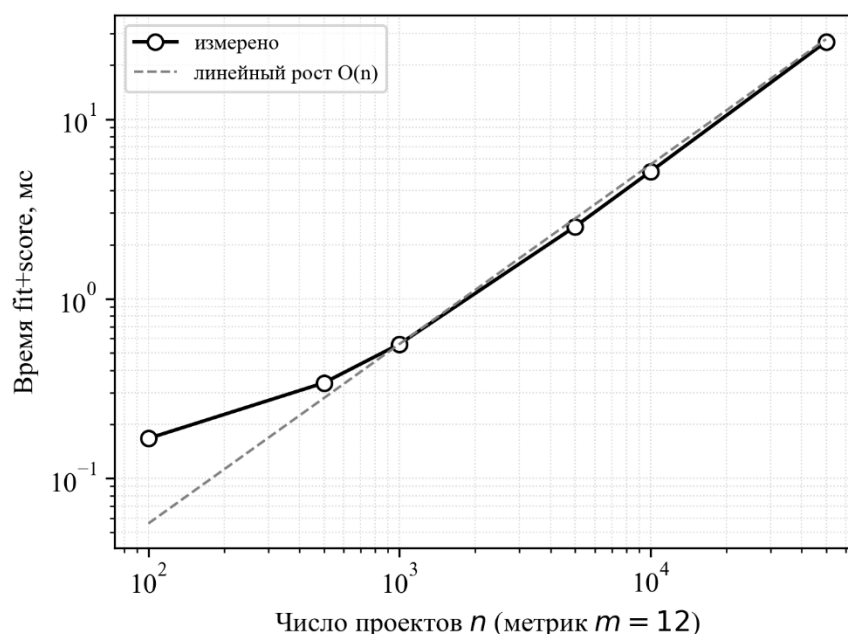


Рисунок 3 – Линейная вычислительная сложность ядра: время расчета в зависимости от числа проектов

Figure 3 – Linear computational complexity of the core: computation time versus the number of projects

Совокупность этих результатов позволяет считать реализацию верифицированной относительно метода в части его формальных инвариантов: она не просто работает, а воспроизводит инварианты CES-свертки, причем негатив-контроль и мутационное тестирование подтверждают, что приемочные тесты эти инварианты действительно проверяют, а не подтверждают их по построению.

Сквозная демонстрация контура на описанном портфеле дала следующую картину. За пятьдесят два прогона система зафиксировала 728 результатов расчета и 226 алертов деградации – 50 высокой важности (падение интегральной оценки Q между смежными сборками не менее чем на 10 %) и 176 средней важности (проседание субиндекса группы не менее чем на 15 %). Проекты покрывают характерные управленческие траектории: сервисы *lifetime-value-api* и *kpi-metrics-api* держатся устойчиво зелеными (зеленая зона на всех сборках); *auth-service* демонстрирует

кратковременную просадку в желтую зону с последующим восстановлением; data-pipeline остается хронически проблемным, не выходя из красной зоны ни на одной сборке, с интегральной оценкой $Q = 0,21$ на последней. Главный же сюжет демонстрации – сервис creative-editor, на котором прослеживается весь путь сигнала.

Итоговую картину портфеля на последней сборке удобно представить как цветовую классификацию зон (Рисунок 4, Таблица 5): из четырнадцати проектов пять зеленых, шесть желтых и три красных. Рядом с деградировавшим creative-editor цветовая классификация показывает устойчиво благополучный lifetime-value-api и хронически проблемный data-pipeline, что и составляет типичную управленческую сводку: где все спокойно, где наметился спад, а где проблема носит застарелый характер.

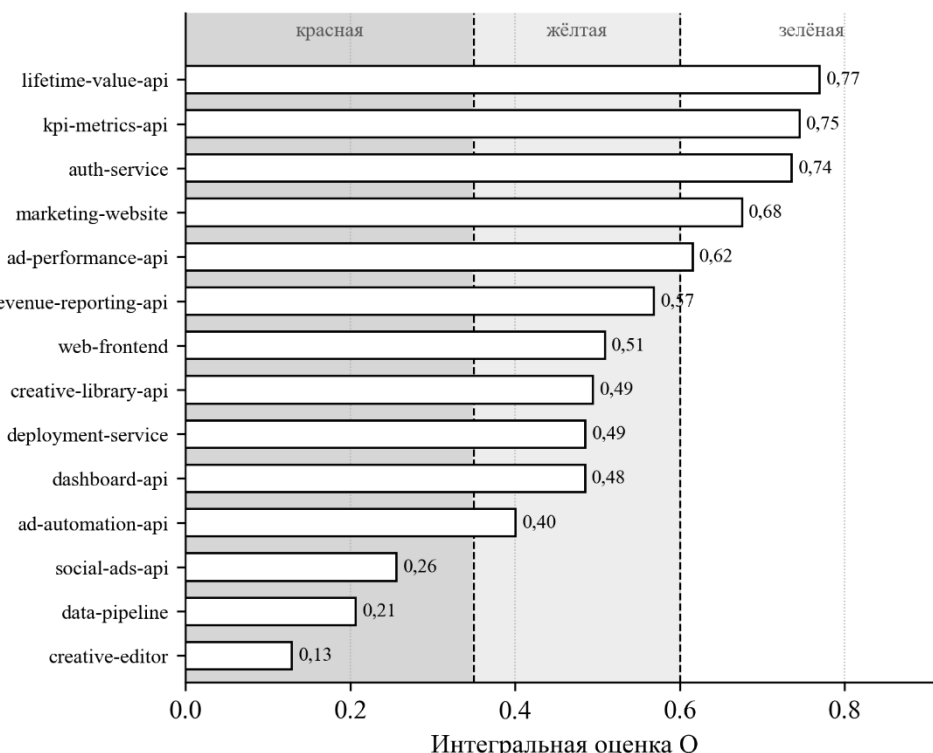


Рисунок 4 – Цветовая классификация состояния портфеля на последней сборке
Figure 4 – Traffic-light status of the demonstration portfolio at the last build

Таблица 5 – Представительные траектории проектов портфеля на последней сборке
Table 5 – Representative project trajectories of the portfolio at the last build

Проект	Интегральная оценка Q	Статус	Характер траектории
lifetime-value-api	0,77	зеленый	стабильно благополучен, зеленый на всех 52 сборках
creative-editor	0,13	красный	деградация из желтой зоны в красную, 37 из 52 сборок в красной
data-pipeline	0,21	красный	хронически проблемный, красный на всех сборках

Деградация creative-editor показывает контур в действии как сквозной кейс (Рисунок 5). Интегральная оценка проекта снижается с 0,464 на первой сборке до 0,129 на последней с устойчивым нисходящим трендом (наклон линейной аппроксимации $-0,007$): начавшись в желтой зоне, она пересекает блокирующий порог θ_Q и закрепляется в красной, так что из пятидесяти двух сборок проект проводит тридцать семь в красной

зоне. Эта траектория сама по себе еще не управленческий вывод – вывод формирует надстройка над оценкой. Сигнализация помечает наиболее резкие просадки Q – до 31 % между смежными сборками – как триггеры высокой важности, а сопутствующие проседания субиндексов групп ниже порога – как триггеры средней важности. К каждому срабатыванию прилагается рекомендация с разверткой по группам метрик, указывающая конкретные метрики-виновники, то есть переводящая факт ухудшения оценки в адресное указание, где искать причину.

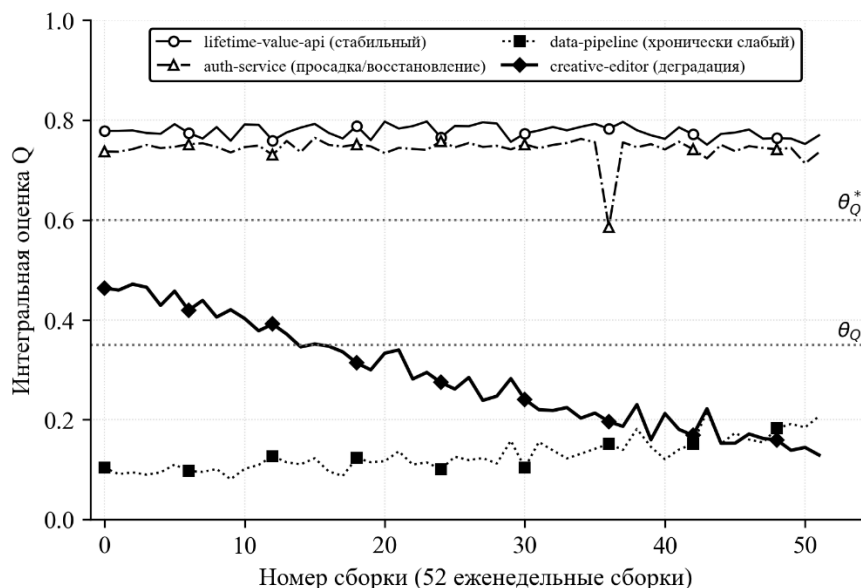


Рисунок 5 – Динамика интегральной оценки Q проектов портфеля от сборки к сборке
Figure 5 – Dynamics of the integral score Q of portfolio projects across builds

Доступ к этой информации организован через дашборд с вкладками «Портфель», «Динамика Q », «Группы метрик», «Сравнение», «Алерты» и «Рекомендации», то есть от обзорной цветовой классификации ЛПР может спуститься к траектории отдельного проекта и далее к разбору по группам метрик. Отчеты выгружаются в форматы электронных таблиц и PDF для включения в регламентную отчетность. Наконец, контур замыкается на процесс разработки через исполняемый гейт CI/CD: команда расчета с заданным порогом важности триггеров возвращает ненулевой код выхода при критической деградации, что позволяет встроить проверку качества прямо в конвейер сборки [4, 5]. В демонстрации именно деградация creative-editor довела бы такой гейт до срабатывания, остановив продвижение проблемной сборки. Подчеркнем еще раз, что все приведенные числа и траектории – интегральные оценки Q , падения Q между сборками, пять зеленых, шесть желтых и три красных проекта последней сборки, а также 728 результатов расчета и 226 алертов за пятьдесят два прогона – получены на реальном, но обезличенном портфеле одной организации; они служат демонстрацией работоспособности и связности механизма.

Экранные формы дашборда показывают, как эти результаты предстают перед ЛПР. Обзорная форма (Рисунок 6) сводит портфель к счетчикам цветных зон, ранжированию проектов по интегральной оценке и сводке состояния портфеля; форма рекомендаций (Рисунок 7) для проблемного проекта показывает приоритет вмешательства и разложение интегральной оценки до отдельных метрик с наибольшим вкладом в отставание, что позволяет определить, на какие из них направить усилия службы качества.

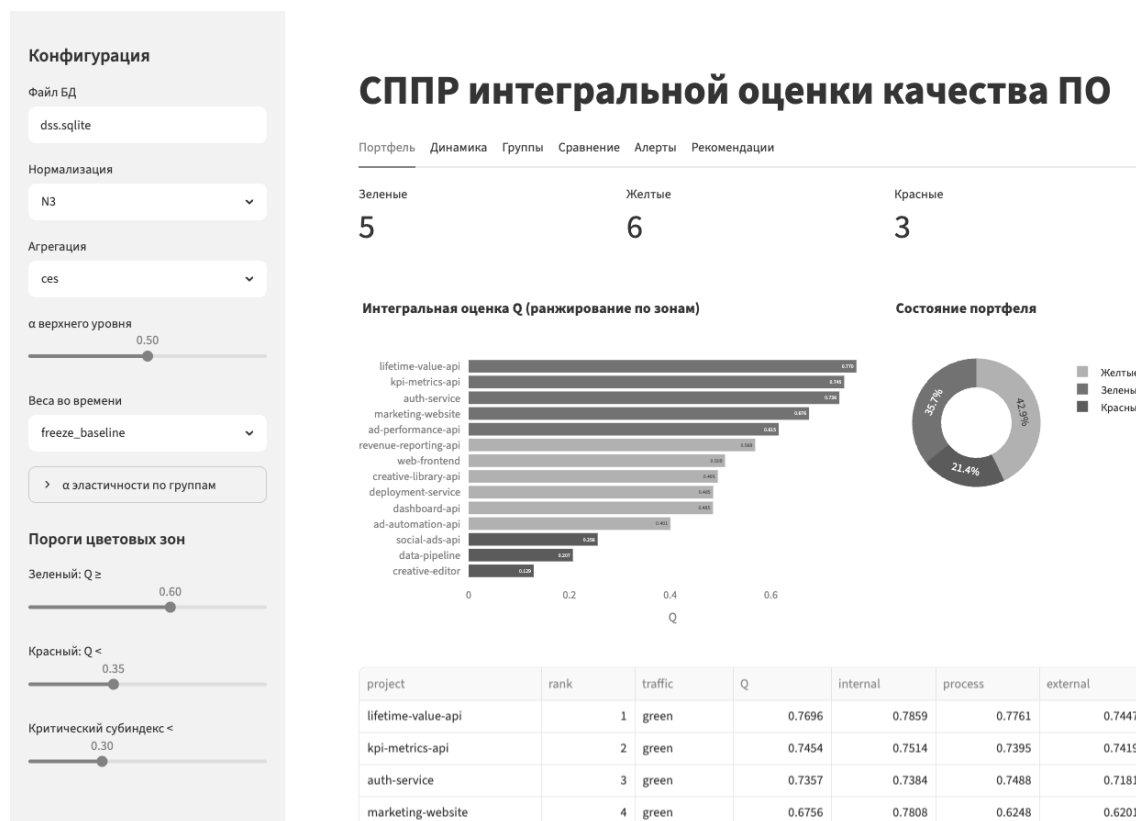


Рисунок 6 – Экранная форма дашборда: вкладка «Портфель» – счетчики цветных зон, ранжирование проектов и сводка состояния портфеля

Figure 6 – Dashboard screen form: the "Portfolio" tab – colour-zone counters, project ranking and portfolio state summary

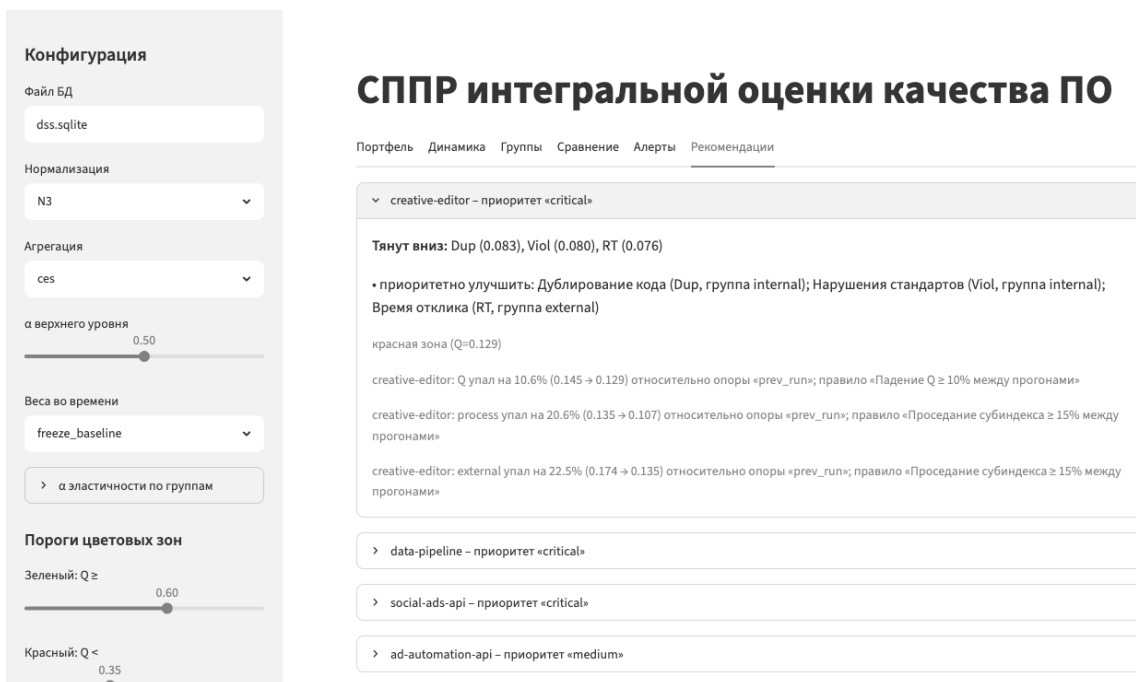


Рисунок 7 – Экранная форма дашборда: вкладка «Рекомендации» – приоритизация вмешательства и разложение оценки до метрик с наибольшим вкладом в отставание

Figure 7 – Dashboard screen form: the "Recommendations" tab – intervention prioritization and breakdown of the score down to the metrics with the largest contribution to the lag

Обсуждение

Предложенная СППР отличается от распространенных инструментальных ворот качества прежде всего природой оценки, на которой основано решение. Типовой quality gate проверяет набор независимых порогов по отдельным метрикам и пропускает сборку, если ни один из них не нарушен; пороговые проверки в конвейере используют близкую логику автономных условий [6, 7]. Показательный промышленный пример – Quality Gate в SonarQube, решающая логика которого по публичной документации продукта¹ представляет собой набор условий вида «метрика сравнивается с порогом», объединяемых по правилу «провал хотя бы одного условия блокирует ворота». Каждое условие здесь автономно и оценивается само по себе; жесткие fail-условия по отдельной метрике возможны, но связи между метриками нет, а единой агрегированной оценки качества, по которой принимается решение, такой механизм не строит. Из-за автономности условий проверка остается компенсаторной по существу: благополучие по большинству показателей не штрафует за провал по одной критической грани, а на уровне портфеля и тренда метрики складываются без учета их взаимного дефицита – достаточно подтянуть формальные показатели, чтобы ворота открылись. Описанный механизм устроен иначе. Он сводит метрики в единую некомпенсируемую оценку и дополнительно контролирует субиндексы групп, поэтому критический дефицит по одной грани снижает целостную оценку независимо от благополучия остальных, а субиндексный порог блокирует выпуск при провале отдельной группы. Решение принимается по целостной величине, пригодной к тому же для распределения ресурсов и управления по тренду, чего пороговый gate с независимыми условиями по построению не дает.

Дополнительное отличие – способ обоснования доверия к самому инструменту: научные свойства метода закреплены как приемочные тесты и проверены негатив-контролем, тогда как промышленные ворота верифицируют корректность подсчета метрик, но не математические свойства решающего правила.

В терминах теории управления организационными системами СППР является программной реализацией механизма комплексного оценивания применительно к автоматически собираемым метрикам качества [25, 26]. Вычислительное ядро играет роль свертки частных оценок в комплексную, а прикладной слой замыкает ее в управленческий контур организационной системы разработки. Авторы не утверждают эквивалентности предложенной непрерывной свертки классическим матричным механизмам; речь идет о функциональном соответствии при ином способе задания свертки и автоматической, а не экспертной, настройке весов.

Отчуждаемость ядра позволяет переиспользовать его независимо от обвязки, а сама стратегия верификации, закрепляющая научные свойства метода как приемочные тесты, отвечает требованию доверия к ПО механизма принятия решений: дискриминирующий набор приемочных тестов, проверенный негатив-контролем и мутационным анализом, подтверждает, что работающая система воспроизводит именно ту оценку, которая обоснована в методе.

Результаты следует трактовать с учетом ограничений. Опытная демонстрация проведена на реальном, но обезличенном портфеле одной организации и подтверждает работоспособность и связность контура, однако не заменяет апробацию на данных промышленного внедрения в нескольких организациях. Пороги решающего правила заданы по умолчанию и подлежат калибровке под организацию по экспертному и ретроспективному каналам. Оценка производительности получена на одной рабочей станции, характеристики которой и методика замера приведены в разделе «Материалы и

методы», и характеризует масштабируемость ядра, а не всей системы с учетом ввода-вывода.

Заключение

Механизм принятия релизных решений на основе некомпенсируемой интегральной оценки качества доведен до развертываемой системы поддержки принятия решений, встроенной в конвейер непрерывной интеграции и доставки. Система автоматически собирает двенадцать метрик, вычисляет интегральную оценку и субиндексы воспроизводимым вычислительным ядром, классифицирует готовность к выпуску трехзонным решающим правилом с прямым использованием неполной компенсации, отслеживает деградацию темпоральными триггерами, формирует адресные рекомендации с разверткой до метрик и исполняет управленческое решение как гейт CI/CD.

Основные результаты следующие. Предложена трехслойная архитектура СППР с отчуждаемым вычислительным ядром, реализующая контур управления качеством в организационной системе разработки. Научные свойства метода закреплены как приемочные тесты, и реализация их воспроизводит: 101 тест ядра и 122 теста прикладного слоя без падений при покрытии 99 % и 98 %, воспроизведение инвариантности к линейным преобразованиям шкалы, согласованности ранжирования и неполной компенсации. Дискриминирующая способность приемочных тестов подтверждена негатив-контролем и мутационным анализом: замена мультипликативной свертки на аддитивную закономерно приводит к падению тестов неполной компенсации, а из 24 операторных мутантов ядра агрегации и взвешивания тестами выявлено 17 (70,8 %), причем выжившие классифицированы как эквивалентные либо защитные ветви, недостижимые на корректных входах. Тем самым свойства метода не просто декларированы, а удержаны реализацией, что подтверждено дискриминирующим набором тестов. Подтверждена линейная вычислительная сложность ядра, пригодная для непрерывного мониторинга крупных портфелей. Обеспечена темпоральная сопоставимость оценок за счет замороженных весов, а управленческое решение оформлено как исполняемый гейт. Работоспособность полного контура показана на реальном обезличенном портфеле организации – четырнадцать проектов за пятьдесят две сборки.

Дальнейшая работа предполагает калибровку порогов решающего правила на накопленных релизах и публикацию полученных значений как параметров по умолчанию, а также расширение набора коннекторов к инструментам разработки. Исходный код ядра `sqa_core` и прикладного слоя `sqa_dss` предоставляется по запросу.

СПИСОК ИСТОЧНИКОВ / REFERENCES

1. Giray G., Bennin K.E., Köksal Ö., Babur O., Tekinerdogan B. On the use of deep learning in software defect prediction. *Journal of Systems and Software*. 2023;195:111537. <https://doi.org/10.1016/j.jss.2022.111537>
2. Matloob F., Ghazal T.M., Taleb N., et al. Software Defect Prediction Using Ensemble Learning: A Systematic Literature Review. *IEEE Access*. 2021;9:98754–98771. <https://doi.org/10.1109/ACCESS.2021.3095559>
3. Коцюба И.Ю., Буюкова Е.С. Методы и модели интеллектуальной поддержки принятия решений по управлению программным сервисом обработки обращений граждан. *Экономический вестник*. 2026;5(2):74–85. <https://doi.org/10.66621/2949-4648-2026-5-2-74-85>

- Kotsyuba I.Yu., Buyakova E.S. Methods and models for intelligent decision support in the management of software services for handling citizens' enquiries. *Economic Bulletin*. 2026;5(2):74–85. (In Russ.). <https://doi.org/10.66621/2949-4648-2026-5-2-74-85>
4. Donca I.-C., Stan O.P., Misaros M., Gota D., Miclea L. Method for Continuous Integration and Deployment Using a Pipeline Generator for Agile Software Projects. *Sensors*. 2022;22(12):4637. <https://doi.org/10.3390/s22124637>
 5. Soares E., Sizilio G., Santos J., da Costa D.A., Kulesza U. The effects of continuous integration on software development: a systematic literature review. *Empirical Software Engineering*. 2022;27(3):78. <https://doi.org/10.1007/s10664-021-10114-1>
 6. Biazotto J.P., Feitosa D., Avgeriou P., Nakagawa E.Y. Technical debt management automation: State of the art and future perspectives. *Information and Software Technology*. 2023;167(2):107375. <https://doi.org/10.1016/j.infsof.2023.107375>
 7. Sas D., Avgeriou P. An Architectural Technical Debt Index Based on Machine Learning and Architectural Smells. *IEEE Transactions on Software Engineering*. 2023;49(8):4169–4195. <https://doi.org/10.1109/TSE.2023.3286179>
 8. Ulan M., Löwe W., Ericsson M., Wingkvist A. Weighted software metrics aggregation and its application to defect prediction. *Empirical Software Engineering*. 2021;26(5):86. <https://doi.org/10.1007/s10664-021-09984-2>
 9. Govindan K., Jepsen M.B. ELECTRE: A comprehensive literature review on methodologies and applications. *European Journal of Operational Research*. 2015;250(1). <https://doi.org/10.1016/j.ejor.2015.07.019>
 10. Keshavarz-Ghorabae M., Amiri M., Zavadskas E.K., Turskis Z., Antucheviciene J. Determination of Objective Weights Using a New Method Based on the Removal Effects of Criteria (MEREC). *Symmetry*. 2021;13(4):525. <https://doi.org/10.3390/sym13040525>
 11. Mukhametzyanov I. Specific character of objective methods for determining weights of criteria in MCDM problems: Entropy, CRITIC and SD. *Decision Making: Applications in Management and Engineering*. 2021;4(2):76–105. <https://doi.org/10.31181/dmame210402076i>
 12. Colakoglu F.N., Yazici A., Mishra A. Software Product Quality Metrics: A Systematic Mapping Study. *IEEE Access*. 2021;9:44647–44670. <https://doi.org/10.1109/ACCESS.2021.3054730>
 13. Siebert J., Joeckel L., Heidrich J., et al. Construction of a quality model for machine learning systems. *Software Quality Journal*. 2022;30(2):307–335. <https://doi.org/10.1007/s11219-021-09557-y>
 14. Lomio F., Moreschini S., Lenarduzzi V. A machine and deep learning analysis among SonarQube rules, product, and process metrics for fault prediction. *Empirical Software Engineering*. 2022;27(7):189. <https://doi.org/10.1007/s10664-022-10164-z>
 15. Barletta V.S., Caivano D., Colizzi L., Dimauro G., Piattini M. Clinical-chatbot AHP evaluation based on "quality in use" of ISO/IEC 25010. *International Journal of Medical Informatics*. 2022;170(4):104951. <https://doi.org/10.1016/j.ijmedinf.2022.104951>
 16. Vafaei N., Ribeiro R.A., Camarinha-Matos L.M. Assessing Normalization Techniques for Simple Additive Weighting Method. *Procedia Computer Science*. 2022;199:1229–1236. <https://doi.org/10.1016/j.procs.2022.01.156>
 17. Arrow K.J., Chenery H.B., Minhas B.S., Solow R.M. Capital-Labor Substitution and Economic Efficiency. *The Review of Economics and Statistics*. 1961;43(3):225–250. <https://doi.org/10.2307/1927286>
 18. Velepucha V., Flores P. A Survey on Microservices Architecture: Principles, Patterns and Migration Challenges. *IEEE Access*. 2023;11:88339–88358. <https://doi.org/10.1109/ACCESS.2023.3305687>

19. Jiménez-Fernández E., Sánchez Á., Ortega-Pérez M. Dealing with weighting scheme in composite indicators: An unsupervised distance-machine learning proposal for quantitative data. *Socio-Economic Planning Sciences*. 2022;83(2):101339. <https://doi.org/10.1016/j.seps.2022.101339>
20. Kizielewicz B., Shekhovtsov A., Sałabun W. pymcdm—The universal library for solving multi-criteria decision-making problems. *SoftwareX*. 2023;22:101368. <https://doi.org/10.1016/j.softx.2023.101368>
21. Liern V., Pérez-Gladish B. Building Composite Indicators With Unweighted-TOPSIS. *IEEE Transactions on Engineering Management*. 2023;70(5):1871–1880. <https://doi.org/10.1109/TEM.2021.3090155>
22. Krichen M. A Survey on Formal Verification and Validation Techniques for Internet of Things. *Applied Sciences*. 2023;13(14):8122. <https://doi.org/10.3390/app13148122>
23. MacIver D.R., Hatfield-Dodds Z., et al. Hypothesis: A new approach to property-based testing. *The Journal of Open Source Software*. 2019;4(43):1891. <https://doi.org/10.21105/joss.01891>
24. Papadakis M., Kintis M., Zhang J., Jia Y., Le Traon Y., Harman M. Mutation Testing Advances: An Analysis and Survey. *Advances in Computers*. 2019;112:275–378. <https://doi.org/10.1016/bs.adcom.2018.03.015>
25. Burkov V.N., Goubko M., Korgin N., Novikov D. *Introduction to theory of control in organizations*. Boca Raton: CRC Press; 2015. 346 p. <https://doi.org/10.1201/b18152>
26. Burkov V.N., Korgin N.A., Novikov D.A. Control Mechanisms for Organizational-Technical Systems: Problems of Integration and Decomposition. *IFAC-PapersOnLine*. 2016;49(32):1–6. <https://doi.org/10.1016/j.ifacol.2016.12.180>

ИНФОРМАЦИЯ ОБ АВТОРАХ / INFORMATION ABOUT THE AUTHORS

Пуговкин Михаил Михайлович, аспирант, **Mikhail M. Pugovkin**, Postgraduate, Tomsk
Томский государственный университет систем управления и радиоэлектроники, Tomsk State University of Control Systems and
Томск, Российская Федерация. Radioelectronics, Tomsk, the Russian Federation.
e-mail: mpugovkin19@gmail.com
ORCID: [0009-0006-4444-2195](https://orcid.org/0009-0006-4444-2195)

Сидоров Анатолий Анатольевич, кандидат экономических наук, доцент, заведующий кафедрой автоматизации обработки информации, Томский государственный университет систем управления и радиоэлектроники, Томск, Российская Федерация. **Anatoly A. Sidorov**, Candidate of Economic Sciences, Docent, Head of the Department of Information Processing Automation, Tomsk State University of Control Systems and Radioelectronics, Tomsk, the Russian Federation.
e-mail: anatolii.a.sidorov@tusur.ru
ORCID: [0000-0002-9236-3639](https://orcid.org/0000-0002-9236-3639)

Статья поступила в редакцию 08.07.2026; одобрена после рецензирования 02.10.2026; принята к публикации 16.10.2026.

The article was submitted 08.07.2026; approved after reviewing 02.10.2026; accepted for publication 16.10.2026.