УДК 658.5

ОРГАНИЗАЦИЯ РАБОТЫ ПОЛУСТАЦИОНАРНОГО КОМПЛЕКСА МОНИТОРИНГА С ИСПОЛЬЗОВАНИЕМ КРОССПЛАТФОРМЕННОЙ ТЕХНОЛОГИИ KOTLIN/COMPOSE MULTIPLATFORM

© 2026 Е.Д. Коршунова, А.В. Капитанов, А.И. Тагаев

Московский государственный технологический университет «Станкин»
(МГТУ «Станкин»), г. Москва, Россия

Статья поступила в редакцию 07.03.2026

Статья посвящена обоснованию и практической организации применения кроссплатформенных технологий при проектировании полустационарного программно-аппаратного комплекса мониторинга в сфере организации производства. В качестве базового подхода рассматривается Kotlin Multiplatform/Compose Multiplatform, обеспечивающий единую кодовую базу и унифицированный HMI при неоднородности рабочих мест (панели станков, мобильные устройства, диспетчерские посты). Предлагается методика управленческого выбора архитектуры и порядка внедрения, увязывающая производственные ограничения (регламенты, интеграции с MES/ERP) с задачами оперативного планирования и диспетчеризации. Метод включает компактную экономическую модель с учётом доли общего кода, что позволяет оценивать трудозатраты разработки и сопровождения и выбирать между кроссплатформенной и гибридной схемами. Показано, что кроссплатформенный подход наиболее целесообразен для «некритичных по времени» экранов, повышая согласованность процессов, сокращая время вывода изменений и снижая издержки эксплуатации в производственных подразделениях.

Ключевые слова: IIoT; HMI; Kotlin Multiplatform; Compose Multiplatform; PoC; BLE; SCADA; MES/ERP.

DOI: 10.37313/1990-5378-2026-28-2(2)-475-482
EDN: AJFGNH

ВВЕДЕНИЕ

Актуальность исследования обусловлена одновременным ростом гетерогенности целевых платформ (панели станков, мобильные устройства, рабочие места диспетчеров) и усилением требований к сквозной интеграции с MES/ERP и архивами временных рядов в проектах «умного» производства. В этих условиях использование Kotlin Multiplatform / Compose Multiplatform позволяет опираться на *единую общую кодовую базу* и *унифицированный интерфейс оператора*, что сокращает дублирование реализации на разных платформах, уменьшает совокупную стоимость сопровождения и повышает согласованность производственных процессов (планирование, диспетчеризация, анализ) [1-5]. Вместе с тем жёсткие ограничения по времени отклика и функциональной безопасности в «горячих» контурах, а также современные нормативные требования к мониторингу состояний (ГОСТ 20911-89; ГОСТ Р ИСО 17359-2015) требуют аккуратного архитектурного выбора и прозрачных критериев внедрения: кроссплатформа целесообразна для экранов без жёстких требований по задержке, тогда как для критичных участков оправдана нативная или гибридная схема с сохранением ответственных функций на специализированных платформах [1-4].

СУЩНОСТЬ ПРИМЕНЕНИЯ КРОССПЛАТФОРМЕННОЙ ТЕХНОЛОГИИ В СФЕРЕ ОРГАНИЗАЦИИ ПРОИЗВОДСТВА

Основа концепции использования кроссплатформенных программных решений лежит в принципе *Write once, run anywhere (WORA) (Напиши один раз, используй везде)* предложенном в 1995 году компанией Sun Microsystems. Данный принцип не теряет своей актуальности и в Индустрии 4.0, где необходимо настраивать взаимодействие программной и аппаратной части, что является сложным и затратным со стороны разработки и поддержки его работы разработчиками и исследователями (которые могут калибровать его работу), за счет использования различных устройств, платформ и операционных систем [1-2].

Коршунова Елена Дмитриевна, профессор, доктор экономических наук, директор Института социально-технологического менеджмента.E-mail: ed.korshunova@stankin.ru , edkorshunova@yandex.ru
Капитанов Алексей Вячеславович, доктор технических наук, доцент, заведующий кафедрой автоматизированных систем обработки информации и управления. E-mail: av.kapitanov@stankin.ru
Тагаев Арсен Ильнурович, аспирант/ E-mail: ion8@bk.ru

Рассмотрим основные трудности при работе с полустационарным (вследствие возможности работы не только в стационарном формате) программно-аппаратным комплексом (ППАК) со стороны организации проведения мониторинга производственных систем (ПС), в которых необходимо поддерживать несколько цифровых платформ (или операционных систем), программных продуктов, написанных на разных языках программирования [6]:

- **Увеличение времени и стоимости внедрения изменений.** Для разработки нового функционала требуется переписывать заново многие модули различных платформ, с дополнительным учетом специфики их языка программирования и программной среды. Другими словами – затраты (которые включают в себя как трудовые, так и финансовые затраты предприятия) на создание нового функционала (FC – Feature Cost) в данном случае возрастает в n раз, где n – число поддерживаемых цифровых платформ и представляют в итоге суммарные затраты на разработку (которые также включают в себя как трудовые, так и финансовые затраты предприятия) – CD (Cost of Development), что указано в формуле 1 [5].

$$CD\ (n)\ =\ n\ *\ FC. \tag{1}$$

- **Усложнение и увеличение стоимости поддержки работы.** Как правило, программные инструменты используемые только в одной экосистеме устройства быстрее устаревают и меньше поддерживаются, в отличии от кроссплатформенных решений. Также, фокусируясь на применении подхода на ППАК мониторинга ПС для начального уровня технологической готовности изделий, следует указать что в нем применяются и портативные устройства (смартфоны, планшеты и т.д.), что увеличивает вариативность платформ и сценариев их применения и, как следствие, повышает необходимость внедрения кроссплатформенного подхода. [5-8].
- **Рост затрат на синхронизацию цифровых платформ.** Синхронизация работы нескольких платформ может выйти в отдельные затраты и расти квадратично (и даже экспоненциально, если учитывать более сложные зависимости). Экспоненцирование «Sync Costs» на степень FC моделирует эффект комбинаторного взрыва этих издержек (формула 2) [5].

$$CD_{sync}\ (n)\ =\ n\ *\ FC\ +\ Sync\ Costs^{FC}\ . \tag{2}$$

Все вышеуказанные риски позволит снизить применение кроссплатформенных решений, например на базе фреймворка Kotlin Multiplatform (Рис. 1) [7].

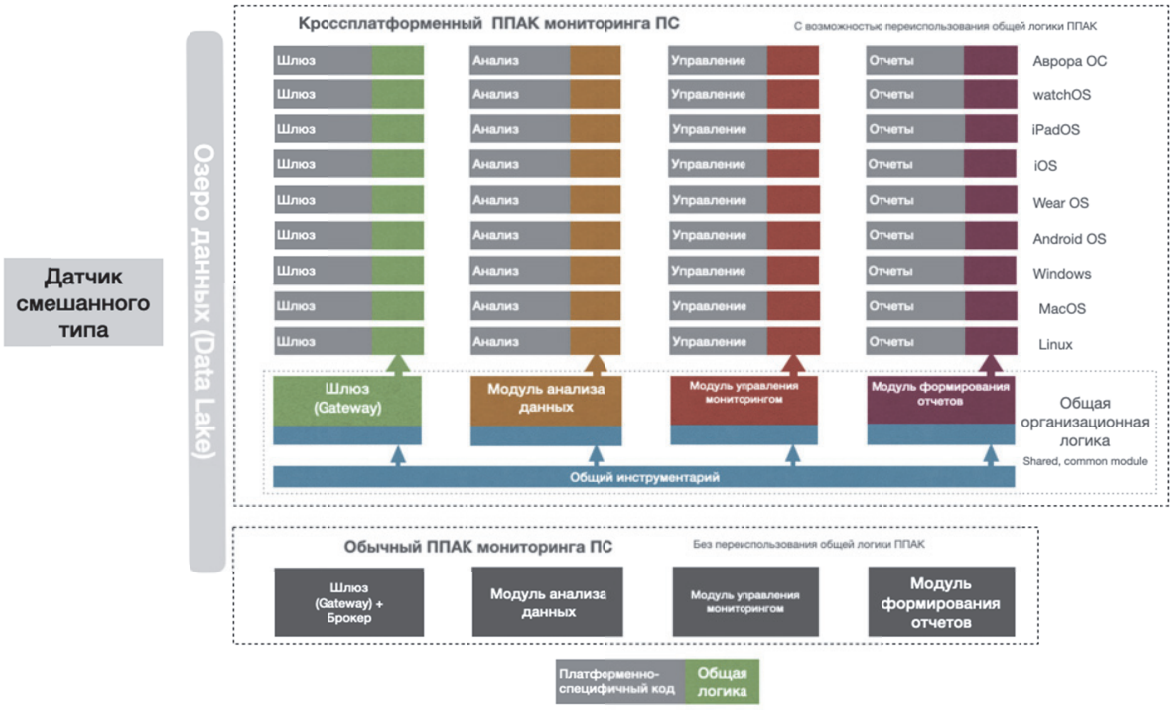

Рис. 1. Схема применения кроссплатформенного подхода в полустационарном программно-аппаратном комплексе (ППАК) мониторинга ПС для начальных уровней готовности изделия

В целом, типовыми сущностями, которые могут быть использованы на разных платформах одновременно и написаны на одном языке программирования с точки зрения применения их в ППАК мониторинга являются: модели данных (рис. 2), программные методы без платформенно специфических имплементаций, REST API клиенты, механизмы сохранения данных и так далее.

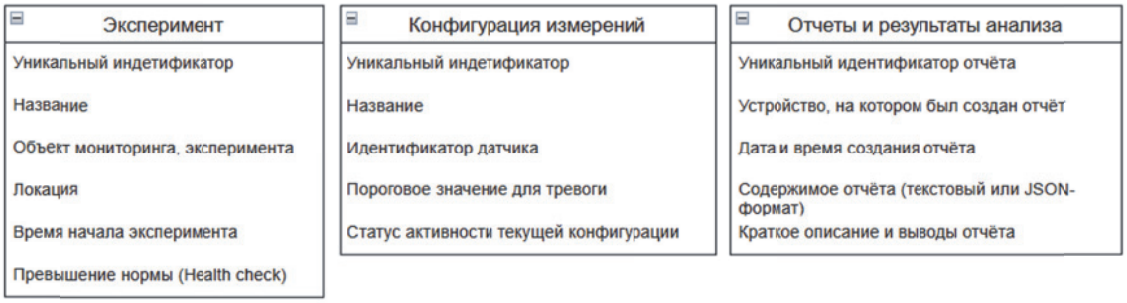

Рис. 2. Модели данных, которые используются в различных модулях (бизнес-логика, хранение данных) программной части ППАК мониторинга №2

Также наиболее общий инструментарий (декодеры, инкодеры, конвертеры и мапперы наиболее используемых сущностей в приложении, то есть если они используются не только в одном модуле) может распределяться среди общих модулей (“commonMain”, “commonTest”) (Рис. 3).

На рисунке 3 показана структура папок кроссплатформенного проекта на базе фреймворка KMP, исходя из которой можно сделать вывод, что он поддерживает распределение общей логики на MacOS (папка “appleMain”), Windows (папка “desktopMain”), iOS (папка “iosMain”) и так далее. Также распределение общего кода происходит исходя из архитектуры ЦПУ (табл. 1), например есть платформы с центральными процессорами на ARM64 и x86.

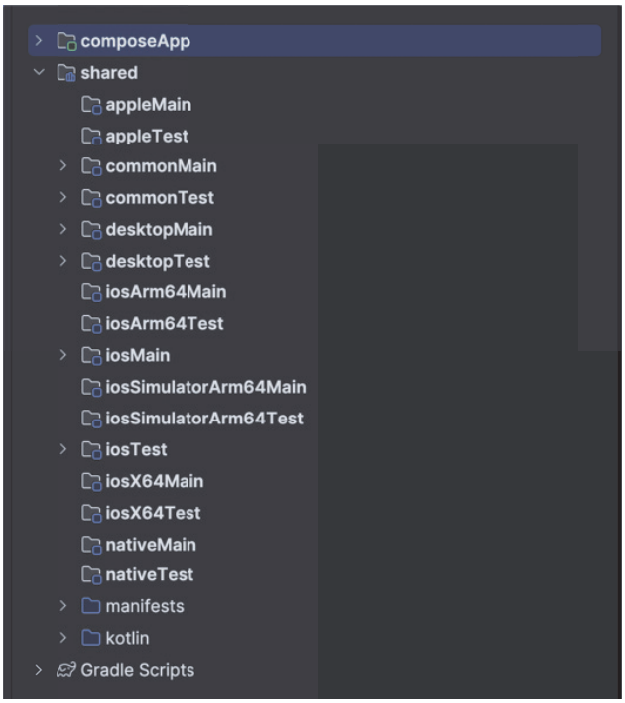

Рис. 3. Структура общей кодовой базы (Shared Module) в кроссплатформенном приложении (показано в среде разработки Inteliji IDEA), где каждая папка содержит код для определенной платформы – для бесшовности применения [7].

Таблица 1. Распределение поддерживаемых платформ и архитектур ЦПУ

Папки	Платформы	Архитектура ЦПУ
commonMain	*Общий модуль*	*Любая*
desktopMain	Windows / Linux / macOS (JVM)	x86_64, ARM64
appleMain	Apple экосистема (iOS, macOS, etc.)	ARM64, x86_64
iosMain	iOS (все варианты)	ARM64, x86_64
iosX64Main	iOS Симулятор на Intel Mac	x86_64
iosArm64Main	Физические iPhone/iPad	ARM64
iosSimulatorArm64Main	iOS Симуляторы на Apple Silicon	ARM64
nativeMain	macOS / Linux / Windows (Kotlin/Native)	ARM64, x86_64, x86
auroraMain	Aurora OS (мобильная и настольная версии)	ARM64, x86_64

Основное достоинство кроссплатформенного подхода лежит в размере модулей общей кодовой базы (shared module) α, охвате платформ и покрытия функционала, где $\alpha \in [0; 1]$ (1: весь код переведен в shared module, и 0: код в shared module отсутствует), формула 3 [5].

$$CD_{crossplatform}(n) = FC * [n * (1 - \alpha) + \alpha]. \quad (3)$$

Представим формулу 4 с учетом количества функциональных модулей равным *i*:

$$CD_{\text{ППАК}} = \sum_{i=1}^{m} FC_{C,i} * [n_i * (1 - \alpha_i) + \alpha_i], \quad (4)$$

где для каждого модуля *i*:

$FC_{C,i}$ – фиксированная стоимость/сложность реализации модуля «с нуля»,

n_i – число платформ, на которых этот модуль требуется (например, для мобильного клиента: $n_{mobile} = 2$ (в следствии наличия 2 наиболее популярных мобильных платформ, доступных в РФ), а для шлюза: $n_{gateway} = 1$),

α_i – доля общего кода этого модуля, переиспользуемая между платформами.

Сконфигурируем формулу 4 под ППАК мониторина начальных уровней готовности изделия ПС, с наличием 4 модулей (шлюз, анализатор, управление мониторингом и модуль формирования отчетов), которые портируются на Android OS, Windows и Linux, и отдельным пятым – BLE датчик вибрации. То есть, кроме датчика, остальные 4 модуля портируются на 3 основные платформы и у каждого из них *n* = 3, произведем расчет ниже в таблице 2 [7-11].

Таблица 2. Расчет стоимости разработки ППАК мониторинга

Модуль ПАК/ППАК и его краткое описание.	**Число платформ n, и доля общего кода α**	**Стоимость разработки одного модуля m.** $FC_{C,i} * [n_i * (1 - \alpha_i) + \alpha_i]$
Датчик программируется «native C/C++» под RTOS и на прочие платформы не портируется	$n_1 = 1,\ \alpha_1 \approx 0$	$FC_{C,1}[1 * (1 - 0) + 0]$
Шлюз/брокер пишется с 80% общего кода, вынесенного в shared module	$n_2 = 3, \alpha_2 \approx 0{,}8$	$FC_{C,2}[3 * (1 - 0{,}8) + 0{,}8]$
Модуль анализа с 90 % общего кода	$n_3 = 3, \alpha_3 \approx 0{,}9$	$FC_{C,3}[3 * (1 - 0{,}9) + 0{,}9]$
Модуль управления с 90 % общего кода	$n_4 = 3, \alpha_4 \approx 0{,}9$	$FC_{C,4}[3 * (1 - 0{,}9) + 0{,}9]$
Модуль отчета с 90 % общего кода	$n_5 = 3, \alpha_5 \approx 0{,}9$	$FC_{C,5}[3 * (1 - 0{,}9) + 0{,}9]$

$$\begin{aligned} CD_{\text{ППАК}} &= FC_{C,1}[n_1 * (1 - \alpha_1) + \alpha_1] + FC_{C,2}[n_2 * (1 - \alpha_2) + \alpha_2] \\ &+ FC_{C,3}[n_3 * (1 - \alpha_3) + \alpha_3] + FC_{C,4}[n_4 * (1 - \alpha_4) + \alpha_4] \\ &+ FC_{C,5}[n_5 * (1 - \alpha_5) + \alpha_5] \\ &= FC_{C,1}[1 * (1 - 0) + 0] \\ &+ FC_{C,2}[3 * (1 - 0{,}8) + 0{,}8] \\ &+ FC_{C,3}[3 * (1 - 0{,}9) + 0{,}9] \\ &+ FC_{C,4}[3 * (1 - 0{,}9) + 0{,}9] \\ &+ FC_{C,5}[3 * (1 - 0{,}9) + 0{,}9] \\ &= FC_{C,1} + FC_{C,2}[0.6 + 0.8] + FC_{C,3}[0.3 + 0.9] + FC_{C,4}[0.3 + 0.9] + FC_{C,5}[0.3 + 0.9] \\ &= FC_{C,1} + FC_{C,2} \cdot 1.4 + FC_{C,3} \cdot 1.2 + FC_{C,4} \cdot 1.2 + FC_{C,5} \cdot 1.2 \end{aligned}$$

Как видим, выражение 4 позволяет определить будущие или текущие трудозатраты для внедрения или поддержки существующего функционала ППАК мониторинга. В том модуле, где меньше всего вынесено в общую кодовую базу функционала – будет наблюдаться большая стоимость внедрения изменений или их поддержки. То есть, например $FC_{C,2} \cdot 1.4 > FC_{C,3} \cdot 1.2$ – стоимость, сложность внедрения изменений для модуля шлюза, брокера будет больше, чем у модуля анализа, за счет того, что у первого было выведено в shared module 80% функционала, а у второго 90%.

СОВРЕМЕННОЕ ОПРЕДЕЛЕНИЕ АКТУАЛЬНОСТИ ПРИМЕНЕНИЯ КРОССПЛАТФОРМЕННОЙ ТЕХНОЛОГИИ ДЛЯ ПРЕДПРИЯТИЯ

В условиях формирования цифровой интегрированной производственной среды, насыщенной средствами IIoT и неоднородной по типам операторских рабочих мест, актуальность кроссплатформенной технологии определяется не столько характеристиками программного стека, сколько её вкладом в организационную устойчивость процессов и воспроизводимостью процедур управления. Применение Kotlin Multiplatform / Compose Multiplatform целесообразно там, где требуется обеспечить единообразие человеко-машинного интерфейса для различных ролей (оператор, мастер, диспетчер), синхронизировать бизнес-логику на множестве устройств и сократить транзакционные издержки внесения изменений при сохранении регламентированной прослеживаемости данных и соответствия отраслевым требованиям к мониторингу состояния оборудования [11-15]. При этом различение контуров с жёсткими ограничениями по времени отклика и функциональной безопасности, с одной стороны, и контуров аналитики, визуализации трендов и отчётности, с другой стороны, задаёт естественную границу применимости кроссплатформенной реализации: для «горячих» экранов приоритет сохраняется за нативными/SCADA-компонентами, тогда как для «некритичных» задач кроссплатформенная архитектура обеспечивает требуемую вариативность, унификацию и снижение совокупной стоимости владения.

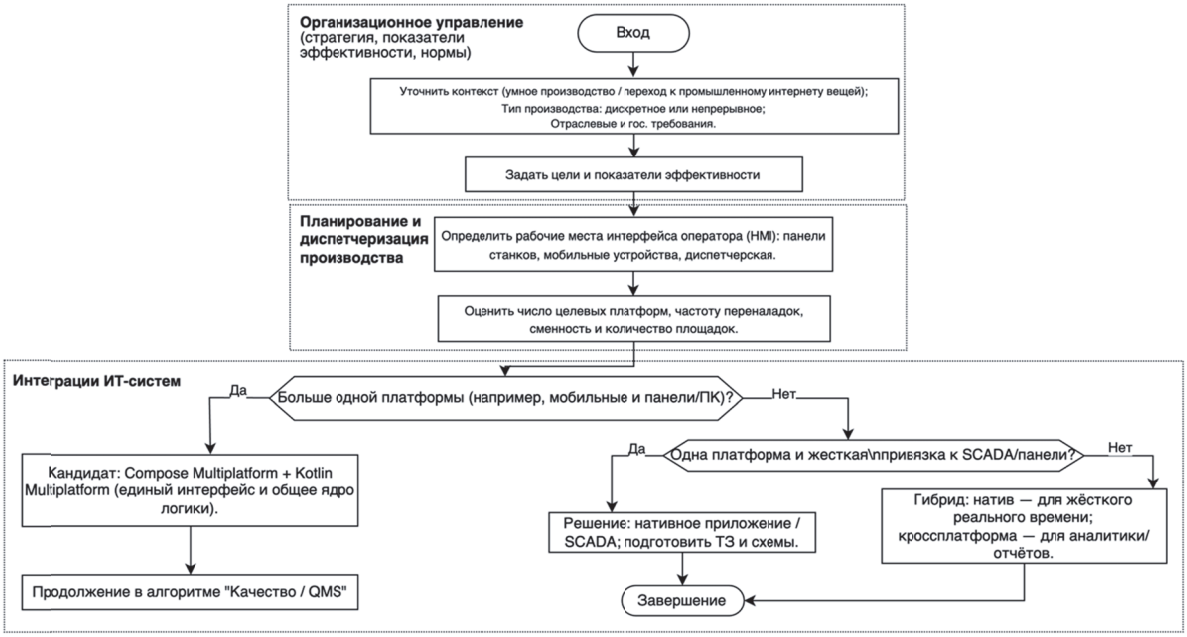

Рис. 4. Алгоритм предварительного выбора архитектуры HMI для ППАК мониторинга с учётом требований к быстродействию и безопасности технологического процесса, мультиплатформенности и интеграций (MES/ERP).

Предлагаемый подход опирается на двухуровневую визуальную модель принятия решений. На Рисунке 4 представлен алгоритм «предварительного выбора», которая последовательно фиксирует производственный контекст (тип производства, профиль технологических рисков, нормативные ограничения), структуру интерфейсных точек с корпоративными системами (MES/ERP, архивы временных рядов) и проверку условий реального времени/безопасности как входного «шлюза» архитектурного решения. Такая формализация переводит вопрос о применимости кроссплатформы из плоскости технологических предпочтений в плоскость управляемых критериев: при наличии критических ограничений решение закрепляется за нативными компонентами, при множественности целевых платформ и требованиях унифицированного UI/UX формируется кроссплатформенный кандидат, а при смешанных условиях рациональной оказывается гибридная конфигурация. Логика алгоритма согласует проектные решения с процессами планирования и диспетчеризации, минимизируя интерфейсную фрагментацию между подразделениями.

На Рисунке 5 алгоритм «реализации и ввода» операционализирует выбранную линию посредством компактного пилотного испытания с измеримыми метриками интерфейса (задержка реакции, частота обновления, совместимость источников IIoT-данных), последующей верификации компетенций команды и выпуском комплекта эксплуатационных документов. Такой порядок обеспечивает воспроизводимость внедрения на разных производственных участках: пилот подтверж-

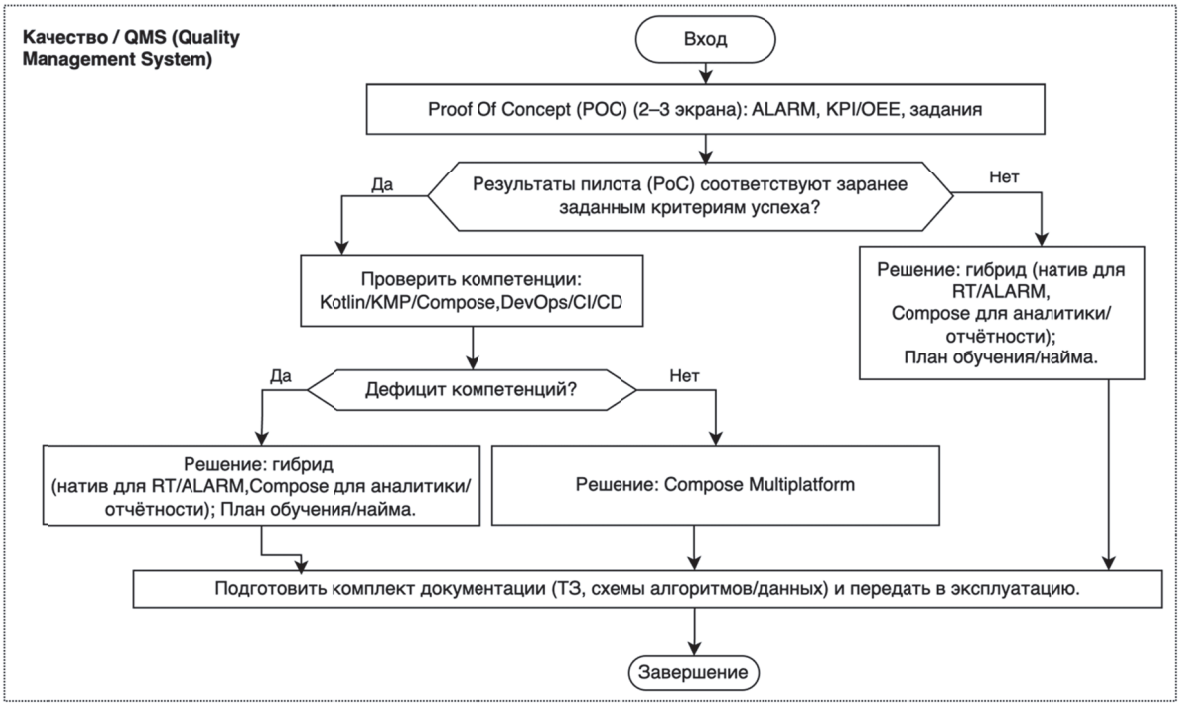

Рис. 5. Алгоритм порядка реализации и ввода в эксплуатацию: пилот (PoC), метрики интерфейса, проверка компетенций, выбор итоговой архитектуры (кроссплатформа/гибрид/натив) и ввод в эксплуатацию

дает достижимость целевых показателей в реальных условиях, оценка компетенций предотвращает организационные риски масштабирования, а документирование по установленным требованиям закрепляет технологическую дисциплину при передаче решения в эксплуатацию. В результате кроссплатформенная технология рассматривается как управленческий инструмент выравнивания интерфейсного ландшафта и ускорения оборота изменений, а не как самоцель; её применимость определяется степенью соответствия производственному профилю предприятия, зрелости IIoT-интеграций и готовности к сопровождению единой кодовой базы.

Сформулированный расчет стоимости разработки ППАК тем самым носит интегральный характер: обеспечивает связывание планирования и диспетчеризации работ, допускает дифференциацию по контурам реального времени и делает более согласованными действия подразделений за счёт унификации интерфейсов, регламентов и артефактов эксплуатации. Два представленных алгоритма выполняют роль нормативного «каркаса» для управленческого выбора и поэтапного ввода, допускающего расширение на другие участки без потери прозрачности процессов и качества данных [11-15].

ЗАКЛЮЧЕНИЕ

В работе предложена и обоснована методика управленческого выбора и поэтапного внедрения кроссплатформенных решений на примере Kotlin Multiplatform/Compose Multiplatform для полустационарного программно-аппаратного комплекса мониторинга. Методика ориентирована на задачи ***организации производства*** и объединяет: (1) формализованный алгоритм предварительного выбора архитектуры HMI с учётом ограничений реального времени/безопасности, числа целевых платформ и требований к интеграциям; (2) регламент операционной реализации через компактный пилот (PoC), проверку компетенций команды и выпуск комплекта эксплуатационной документации по установленным нормам.

Проведённый анализ показывает, что кроссплатформенный подход целесообразен прежде всего для «некритичных по времени» экранов, характерных для диспетчеризации, анализа и отчётности, где унификация интерфейса и повторное использование общей кодовой базы позволяют ***снизить трудозатраты разработки и сопровождения***, уменьшить время вывода изменений и повысить согласованность действий подразделений. Для контуров с жёсткими требованиями по отклику и безопасности оправдана ***гибридная схема***, при которой ответственные функции остаются нативными/SCADA, а надсистемные задачи реализуются кроссплатформенно. Экономическая интерпретация (модели на основе доли общего кода) подтверждает: увеличение «общего» слоя приводит к

статистически ожидаемому снижению совокупной стоимости разработки при фиксированном числе поддерживаемых платформ.

С точки зрения ***организации производства*** предложенный подход обеспечивает:
— связку между планированием и диспетчеризацией работ (единый HMI-шаблон для операторов, мастеров и инженеров);

— стандартизацию процессов разработки и ввода (типовые критерии пилота, метрики интерфейса, требования к совместимости и комплекту документации);

— управляемое развитие компетенций (матрица навыков по KMP/CMP, CI/CD, интеграциям) и прозрачное распределение ответственности;

— повышение управляемости изменения и прослеживаемости решений (явные «ворота» выбора архитектуры и критерии принятия).

Полученные результаты подтверждают практическую значимость кроссплатформенного подхода для предприятий, находящихся как в фазе перехода к «умному» производству, так и на стадии масштабирования уже внедрённых систем мониторинга.

СПИСОК ЛИТЕРАТУРЫ

1. *Herrera-Vidal G., et al.* Complexity Management Challenges in the Industry 4.0 Era: A Systematic Review in Production Systems // Results in Engineering. — 2025. — Vol. , — P. .. — URL: https://www.sciencedirect.com/ (дата обращения: 23.05.2025).
2. *Lee, Jay, Behrad Bagheri, and Hung-An Kao.* "A cyber-physical systems architecture for industry 4.0-based manufacturing systems." Manufacturing letters 3 (2015): 18-23.
3. *Korshunova E.D.* Organizational development as a tool for adapting an industrial enterprise to digital transformation // Science today: challenges and development prospects: materials of the international scientific and practical conference, Vologda, September 14, 2016. Vologda: OOO Marker, 2016. Pp. 24–26.
4. *Korshunova E.D., Svarnik P.N.* Business model of enterprises in the digital transformation of industry // Digital economy: equipment, management, human capital: materials of the all-Russian scientific and practical conference. Vologda, 2017. Publisher: OOO Marker. Pp. 36–38.
5. *Nagy, Robert*. Simplifying Application Development with Kotlin Multiplatform Mobile: Write robust native applications for iOS and Android efficiently. Packt Publishing Ltd, 2022.
6. *Kovalev, Ilya, Elena Korshunova, and Arsen Tagaev.* «Innovative, Semi-Permanent Sensors in the Era of Industry 4.0: Technological Solutions for Efficient Monitoring and Automation.» *2025 International Russian Smart Industry Conference (SmartIndustryCon)*. IEEE, 2025.
7. Kotlin Multiplatform Documentation // JetBrains. URL: https://www.jetbrains.com/help/kotlin-multiplatform-dev/get-started.html (дата обращения: 14.05.2025).
8. Android Support for Kotlin Multiplatform (KMP) to Share Business Logic Across Mobile, Web, Server, and Desktop // Android Developers Blog [Электроный ресурс] URL: https://android-developers.googleblog.com/2024/05/android-support-for-kotlin-multiplatform-to-share-business-logic-across-mobile-web-server-desktop.html (дата обращения: 14.05.2025).
9. *Tagaev, Arsen, Kovalev Ilya, and Murad Abdulzagirov.* «Cross-Platform Development Approach to Monitor Production Systems Using MQTT and BLE.» *2025 International Russian Automation Conference (RusAutoCon)*. IEEE, 2025.
11. Analysis of the structure of a semi-stationary hardware and software complex (HSSC) for monitoring the state of production systems in Industry 4.0. A. I. Tagaev, E. D. Korshunova // Fundamental and Applied Problems of Engineering and Technology, OSU named after Turgenev, 2025, issue No. 3.
12. *M. E. Porter and J. E. Heppelmann,* ''How smart, connected products are transforming competition,'' *Harvard Bus. Rev.*, vol. 92, no. 11, pp. 64–88, Nov. 2014. [Online]. Available: https://hbr.org/2014/11/how- smart-connected-products-are-transforming-competition
13. *Khalaifat, Walla, and Fawzi Abdulaziz*. "Challenges and Opportunities of DevOps for IoT Systems: DevOps-For-IoT." 2024 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT). IEEE, 2024.
14. *Oleff, Alexander, et al.* "Process monitoring for material extrusion additive manufacturing: a state-of-the-art review." Progress in Additive Manufacturing 6.4 (2021): 705-730.
15. *Jónasdóttir, Hanna, et al.* "Upgrading legacy equipment to industry 4.0 through a cyber-physical interface." Advances in Production Management Systems. Smart Manufacturing for Industry 4.0: IFIP WG 5.7 International Conference, APMS 2018, Seoul, Korea, August 26-30, 2018, Proceedings, Part II. Springer International Publishing, 2018.

ORGANIZING A SEMI-STATIONARY MONITORING COMPLEX USING KOTLIN/COMPOSE MULTIPLATFORM CROSS-PLATFORM TECHNOLOGY

© 2026 E.D. Korshunova, A.V. Kapitanov, A.I. Tagaev

Moscow State Technological University "Stankin"
(MSTU "Stankin"), Moscow, Russia

This article examines the rationale and practical application of cross-platform technologies in the design of a semi-stationary hardware and software monitoring complex for industrial automation. Kotlin Multiplatform/Compose Multiplatform is used as the basic approach, providing a unified code base and a unified HMI across heterogeneous workstations (machine panels, mobile devices, control stations). A methodology for managerial selection of architecture and implementation procedures is proposed, linking production constraints (regulations, integration with MES/ERP) with operational planning and dispatching tasks. The method includes a compact economic model taking into account the share of shared code, allowing for the assessment of development and maintenance costs and the choice between cross-platform and hybrid approaches. It is shown that a cross-platform approach is most appropriate for "non-time-critical" screens, improving process consistency, reducing change implementation time, and lowering operating costs in production units.
Keywords: IIoT; HMI; Kotlin Multiplatform; Compose Multiplatform; PoC; BLE; SCADA; MES/ERP.

DOI: 10.37313/1990-5378-2026-28-2(2)-475-482
EDN: AJFGNH

Elena Korshunova, Professor, Doctor of Economics, Director of the Institute of Social and Technological Management. E-mail: ed.korshunova@stankin.ru, edkorshunova@yandex.ru
Aleksey Kapitanov, Doctor of Engineering, Associate Professor, Head of the Department of Automated Information Processing and Control Systems. E-mail: av.kapitanov@stankin.ru
Arsen Tagaev, Graduate Student, E-mail: ion8@bk.ru