От репозитория к конвейеру:

современные средства разработки ПО – требования, вызовы и решения

Максим КОЗЛОВ,
технический директор GitFlic, «Группа Астра»

С 2022 г. российский рынок средств разработки программного обеспечения столкнулся с заметной трансформацией. В 2022–2023 гг. многие еще ориентировались на западные продукты, рассчитывая, что ситуация вернется к прежней. Однако в 2024–2025 гг. стало ясно: импортозамещение уже не опция, а стратегическая необходимость.

Параллельно возникли серьезные риски использования зарубежных платформ: от внезапного отказа в доступе и поддержке до уязвимостей в цепочке поставок, которые особенно ощутили компании, работающие с критической инфраструктурой.

В новой реальности российские команды и сообщество разработчиков активизировались. Стало понятно, что нужен переход от простого хостинга кода к полноценным платформам разработки с CI/CD, артефакт-менеджментом и диктатом безопасности. В связи с этим эксперты группы компании «Астра» рассказали о ключевых трендах и вызовах на рынке средств разработки в России, а также ответили на вопросы о том, почему импортозамещение DevOps-инструментов стало необходимостью, и какие решения помогают компаниям строить безопасные и эффективные конвейеры поставки ПО.

Безопасная разработка: мягкое регулирование

Разработка безопасного программного обеспечения (РБПО) – комплекс мер, направленных на предотвращение появления и устранение уязвимостей программы. Цель – исключить внутренние ошибки, которые в процессе эксплуатации могут привести к нерегламентированному поведению. Вопросы РБПО регулируются, в частности, ГОСТом Р 56939–2024 «Защита информации. Разработка безопасного программного обеспечения. Общие требования». По сути, это локализованный аналог стандарта Secure SDLC, причем не ярлык, а проверяемая дисциплина: трассируемость требований до тестов и релизов, контроль изменений, воспроизводимые сборки, управляемые зависимости, статус-чеки на уязвимости и лицензии. Такой подход становится предпосылкой обновления критической инфраструктуры: без него сложно пройти аудит и сертификацию, обеспечить предсказуемость релизов и управляемость рисков.

Внедрению РБПО препятствуют «приземленные» факторы. Без жестких регуляторных требований мотивация «перелопачивать» инструменты разработки заметно ниже, чем, например, импортозамещать клиентские приложения или базовые офисные стеки. Во многих организациях среды разработки стоят последними в очереди на замену. Отдельная проблема – массив неатрибутированного ПО: исторический код без корректных лицензий, без аудита зависимостей, без воспроизводимых сборок. Частая ошибка – начинать с безопасности, когда процессы еще не формализованы и не автоматизированы: сканеры SAST/DAST закуплены, но их эффективность невелика, потому что проверки не встроены в поток и не блокируют вредоносные изменения.

Дискуссия о государственном репозитории за прошедшие годы тоже эволюционировала. Если в 2022–2023 гг. она казалась ответом на все вопросы, то сегодня рынок имеет достаточное количество устойчивых коммерческих площадок, и тема сместилась к точечному регулированию. Это единые правила безопасной публикации и распространения кода, процедуры реагирования на уязвимости и запрещенный контент, а также совместимость с регуляторными требованиями.

Слишком жесткое регулирование может тормозить конкуренцию и инновации, тогда как аккуратные общие принципы, наоборот, повысят качество и предсказуемость. В любом случае мы остаемся вне политики и строго работаем в правовом поле РФ: соблюдаем требования регуляторов, оперативно реагируем на законные запросы и строим процессы так, чтобы пройти все проверки.

Чего ждут современные разработчики от средств разработки

Современным командам недостаточно просто репозитория. Нужна платформа, которая превращает коммиты в проверенные артефакты и управляемые релизы, а затем в наблюдаемую эксплуатацию. Поддержка стеков должна быть широкой: Java/JVM, Python, Go, C/C++, C#, Kotlin, Rust, PHP, JavaScript/TypeScript и др. Под «поддержкой» мы понимаем не только хранение кода, но и шаблоны пайплайнов, зрелый артефакт-менеджмент (реестры пакетов и контейнеров), анализ зависимостей и лицензий, сканирование уязвимостей, детерминированные сборки, подпись и SBOM. На практике в «узких местах» часто оказываются артефакты: популярные экосистемы покрыты «из коробки», для остальных приходится дополнять платформу внешними сервисами или писать обвязку, особенно когда речь об on-prem.

С точки зрения DevOps-инженеров ожидания прагматичные. Им нужны полная автоматизация CI/CD, интеграция с Kubernetes и контейнеризацией, инфраструктура как код (Terraform/Ansible), секрет-менеджмент, единая наблюдаемость (метрики, логи, трейсы) и алертинг по SLO/SLA. Плюс соответствие международным практикам: DORA-метрики (частота деплоев, lead time, процент неудачных изменений, MTTR), shiftleft-проверки, элементы безопасной цепочки поставки (подпись артефактов, SBOM, аттестации происхождения). Миграция с западных систем должна проходить безболезненно: перенос репозиториев, задач, пользователей, пайплайнов, сопоставление ролей и агентов, сопоставимые API и вебхуки. В идеале платформа становится единым «источником правды» для разработчиков, QA, DevOps/SRE, служб безопасности и менеджмента.

ИИ-ассистенты разработчика занимают в процессе свое полезное, но вспомогательное место. В продуктивной практике они ускоряют рутину: генерацию шаблонов (тестов, конфигураций, начальных «скелетов» сервисов), подготовку рефакторинговых планов, навигацию по большому коду и объяснение чужих фрагментов. Однако они не заменяют дисциплины конвейера, тестирования и ревью. Любые предложения модели проходят через те же quality gates (сборку, тесты, линтеры, SAST/DAST, approvals), что и «ручной» код. В организациях с повышенными требованиями к приватности и авторским правам внедрение ИИ требует явных архитектурных и юридических оговорок: запрета на отправку исходников во внешние облака, selfhosted-моделей или прокси с анонимизацией, четких пользовательских политик (какой код можно показывать ассистенту), контроля лицензий и исключения лицензионной контаминации (например не допускать генерации фрагментов, явно совпадающих с GPL-кодом). На практике это оформляется через локальное развертывание LLM, RAG-подход с внутренними базами знаний (политики, стандарты кодирования, архитектурные решения), аудит логов подсказок, а также интеграцию результатов работы ассистента в пайплайны так, чтобы они были «первого класса» артефактами процесса: проверяемыми, версионируемыми и атрибутированными.

Два подхода к организации платформ разработки

В индустрии ужились два полюса. Первый – фрагментированный подход в стиле Atlassian, когда предлагаются отдельные продукты под разные функции: задачник, документация, репозитории кода, CI/CD, артефакты. Сильные стороны такого подхода – глубина каждого инструмента и богатая экосистема плагинов. Слабые – разорванность процесса, сложные интеграция и обновления, рост TCO и риски на стыках. В российском контексте добавляется фактор окончания поддержки Atlassian Server: выбор между Data Center/Cloud и альтернативами требует серьезных усилий и бюджета, особенно в сценариях on-prem.

Второй подход – интегрированная платформа, где GitLab стал де-факто стандартом конвейера разработки: репозитории, merge-процессы, CI/CD, артефакты, безопасность и управление проектами идут в одном потоке. Рынок постепенно голосует за интеграцию: скорость внедрения выше, наблюдаемость лучше, контроль качества и безопасность встроенные, а стоимость владения ниже, особенно когда нужны on-prem и «воздушные зазоры». В России одна часть решений исторически ближе к сборке из «кубиков», другая – к конвейеру как ядру. Мы, разработчики GitFlic, осознанно развиваемся во втором направлении: конвейер – сердце платформы, все остальное – логичное продолжение этого потока.

Миграция с фрагментированных связок в стиле Jira + Bitbucket + Bamboo + Artifactory/Nexus – это не только перенос данных, но и шанс пересобрать сам процесс. Практически успешно работают сценарии, где миграция используется как точка входа для внедрения DORA-метрик (частота деплоев, lead time, доля неудачных изменений, MTTR), стандартизации стратегий ветвления, выстраивания единых quality

gates (обязательные тесты, статический анализ, пороги покрытия, approvals, политика Code Owners), а также для построения безопасной цепочки поставки (SBOM, подпись и аттестация артефактов, контроль лицензий). Технически это означает подготовку организационных шаблонов пайплайнов, унификацию чек-листов ревью, настройку единых ролей и политик веток, этапную миграцию: сначала пилотные команды, затем распространение паттернов на остальные. Важный элемент – измеримость до/после: фиксация базовых показателей до миграции и контроль динамики после нее позволяет не верить, а видеть эффект в цифрах.

Проблема фрагментированности и ее влияние на разработку

Фрагментированность почти всегда влечет за собой лишнюю ручную работу: интеграционные шины, синхронизацию прав и ролей, мониторинг на «стыках», сведение статусов из разных систем. Отсюда типовые потери: «зеленый» билд внезапно не воспроизводится в соседнем контуре, артефакты теряют метаданные, проверки дублируются или противоречат друг другу, а единый ответ на простые вопросы – что именно идет в продакшн, из чего это собрано, кто согласовал изменение – получить довольно сложно. Это «тормозит» релизы, мешает прогнозировать сроки и повышает риск инцидентов.

Здесь хотелось бы привести аналогию с производством, конкретно – с автомобилестроением. Один из всемирных лидеров индустрии компания Toyota исповедует подход, включающий такие элементы, как непрерывный поток, встроенное качество (jidoka) и право останавливать линию при дефекте (andon) – это про связанный процесс с ответственными на каждом этапе. Разорванные конвейеры, где брак «ловится» потом, неизбежно дороже: в ИТ это превращается в поздние «сюрпризы» перед релизом, стресс и неустойчивые поставки. Связанный конвейер снижает вариативность, делает сроки и качество предсказуемыми и переводит обсуждение из плоскости героических ночных выкладок в плоскость управляемых метрик и дисциплины.

Безопасность и качество в современной разработке

Безопасность и качество в 2025-м – это не дополнение, а свойства конвейера. SAST/DAST, анализ зависимостей и контейнеров работают «по-настоящему» только в потоке: проверки запускаются на каждый merge request, в релизных ветках, их статусы попадают в quality gates и могут блокировать слияние. Российские инструменты – статический анализ от Positive Technologies, «Ростелеком Солар», решения ИСП РАН и других ИБ-разработчиков уверенно интегрируются в такие пайплайны. Кроме того, сообщество осваивает практики безопасной цепочки поставки: SBOM (SPDX/CycloneDX), подпись артефактов, аттестации происхождения, приватные реестры и репликации для изолированных контуров.

Качество зиждется на ранних проверках и понятных «воротах». Линтеры, стилевые и архитектурные правила, юнит и интеграционные тесты, покрытие, обязательные code review и approvals – все это превращается в статус-чеки, без которых MR «не сольется». Важна воспроизводимость: фиксированные версии инструментов и зависимостей, детерминированные сборки, изолированные раннеры, одинаковые шаблоны пайплайнов для команд. И правильная последовательность РБПО, когда сначала формализуем процессы, затем автоматизируем поток, потом системно строим качество и уже после усиливаем безопасность. Иначе инвестиции в сканеры и аудиты растворяются в «ручном хаосе».

Экономическая эффективность современных средств разработки

Экономика DevOps – это не про сокращение инженеров конвейера, их и так мало, а их миссия – стабильность и эволюция линии, а не производственные нормы. Реальная экономия появляется в производительности разработчиков и снижении инцидентов: меньше циклов доработок благодаря ранним проверкам, меньше «пожаров» у заказчиков, быстрее time-to-market и выше пропускная способность команды. Там, где конвейер выстроен, спор «скорость против качества» исчезает: при правильных quality gates они растут вместе.

Механика здесь понятная. Автоматизация рутины – от шаблонов пайплайнов до автопубликации артефактов – убирает «невидимую ручную работу». Shiftleft радикально уменьшает стоимость исправления дефектов: найденный на первом этапе баг на порядок дешевле «позднего» и не порождает «хвостов» в смежных командах. Прозрачность по DORA-метрикам позволяет заранее выявлять деградации, снижать MTTR и аргументированно принимать решения о релизах. По нашему опыту внедрений, переход с «зоопарка» интеграций и ручных стыков на интегрированный конвейер дает кратный прирост скорости для команд, которые прежде много времени тратили на синхронизацию и согласования, и заметно уменьшает дефекты в проде.

Экосистема и интеграции

Индустрия смещается от точечных интеграций к сквозным платформам – от IDE до эксплуатации. В России появляются зрелые инициативы уровня OpenIDE («Группа Астра», Axiom

JDK, Haulmont), которые закрывают «точку входа» разработчика и нативно встраиваются в общий поток. Для сценариев on-prem критично, чтобы все шаги – сборки, тесты, публикации и эксплуатация – жили внутри периметра, с локальной поддержкой и полным аудитом.

На уровне интеграций вырисовывается обязательный минимум:

- Тест-менеджмент (например TestIT) для автоматизации и воспроизводимости сценариев;
- инструменты безопасности для кода, зависимостей, контейнеров и лицензий как этапы пайплайна и части quality gates;
- системное управление контейнерами и эксплуатацией (Kubernetes и, в частности, «Боцман») как продолжение конвейера после сборки;
- наблюдаемость: метрики, логи, распределенные трейсы и алерты по SLO;
- публикация в корпоративных каталогах и на маркетплейсах (например RuStore) с управлением версиями и отзывами.

Платформенный подход означает, что все роли – разработчики, QA, DevOps/SRE, безопасность, менеджмент – работают в одном «источнике правды», а с партнерами выстраиваются глубокие, стандартизованные интеграции вместо единичных коннекторов.

GitLab в России: риски зависимости от международного стандарта

Сервис GitLab стал одним из наиболее популярных конвейеров благодаря интегрированности и полноте. Он охватывает все ключевые этапы от репозитория до релиза, безопасности и управления с возможностью начать с малого и масштабироваться, не ломая парадигму. Эту модель принимают и стартапы, и корпорации: удобно, когда весь SDLC живет в одном потоке и в одном домене ответственности. Параллельно GitLab расширяет зону ответственности в сторону требований и портфельного управления, что отражает тренд «меньше систем – больше глубины».

В российских реалиях, однако, есть риски, которые нельзя не принимать во внимание: сложности с легальным приобретением лицензий и получением поддержки, опасность внезапного ограничения доступа и критическая зависимость от зарубежной инфраструктуры: аккаунтов, реестров, удостоверяющих сервисов, что в контурах с повышенными требованиями к автономности превращается в уязвимость. Для многих компаний это не столько технический, сколько юридический и операционный риски, которые трудно закрыть договорами. Поэтому местные команды обоснованно смотрят на локальные альтернативы, которые сохраняют преимущества международного стандарта процесса (парадигму конвейера), но добавляют независимость: локальную поддержку, соответствие требованиям РБПО и интеграции с российской экосистемой. Это позволяет строить процессы без компромиссов в части безопасности с опорой на практики, устоявшиеся в мировом сообществе.

GitFlic в контексте современных требований к разработке

По оценкам аналитиков, рынок DevOps и автоматизации в стране растет, отчеты говорят об устойчивом увеличении спроса на CI/CD-решения. Важно отметить, что сегодня в России шесть вендоров развивают Git-подобные решения, а комьюнити разработчиков все активнее использует отечественные платформы, что само по себе является маркером зрелости рынка. Мы сознательно развиваем GitFlic как интегрированную платформу: знакомые индустрии парадигмы и интерфейсы, возможность миграции с западных систем и работы on-prem и в замкнутых контурах. Наша цель – чтобы разработка, тестирование, безопасность и эксплуатация складывались в единый воспроизводимый поток, где каждое изменение прослеживается от задачи до релиза, а артефакт – от исходников и зависимостей до подписи и SBOM. В этом смысле GitFlic – не система, в которой есть конвейер, а сам конвейер разработки, вокруг которого организована работа команд. На уровне практик качества и безопасности мы используем «ворота» с обязательными автоматическими проверками (линтерами, тестами, статическим и динамическим анализом, анализом зависимостей и контейнеров, лицензий) и человеческими согласованиями: code review, approvals, code owners, релизными ролями. Результаты российских сканеров безопасности включаются в quality gates и реально влияют на допуск изменений, артефакты подписываются, сопровождаются SBOM, а доступы и действия прозрачно аудируются. Для корпоративных сценариев важны управление доступом и аутентификацией (SSO, LDAP/AD/SAML, 2FA), изоляция сред, раздельные контуры для «горячих» и «холодных» веток, а также масштабирование до кластеров с высокой доступностью и репликациями.

Мы уделяем внимание миграции: переносим репозитории, задачи, артефакты и пайплайны, настраиваем сопоставимость ролей и агентов, сохраняем привычную разработчикам парадигму. В экосистеме «Группы Астра» соединяем Open IDE как рабочее место разработчика, GitFlic как конвейер и «Боцман» как платформу эксплуатации Kubernetes. В партнерстве с TestIT реализуем тест-менеджмент и автотесты, вместе с TeamStorm – управление проектами, сотрудничаем с российскими вендорами ИБ-решений. В результате выстраивается сквозной цикл: от нажатия клавиш в IDE до наблюдаемой эксплуатации у заказчика – с единым «источником истины» или данных, воспроизводимостью и управляемыми рисками. Наша задача – не замещать ради замещения, а дать командам промышленный инструмент, соответствующий международным стандартам процесса и требованиям локальной регуляторики. ■