Введение

В качестве объекта для анализа выбрана студия разработки игр SOBAKAGAMING. На текущий момент это микробизнес в стадии формирования: команда представлена одним разработчиком (соло-разработчик), который выполняет все функции — от написания кода до публикации и маркетинга. Основным рынком сбыта является платформа Яндекс Игры (браузерные и веб-приложения). Стратегической целью студии является выход на новые, более крупные и маржинальные рынки: магазины приложений Steam (ПК) и Google Play (мобильные устройства). Данный переход требует кардинального пересмотра не только самих игр, но и информационных процессов, сопровождающих их разработку и распространение.

1. Анализ текущей ситуации (AS-IS)

В настоящее время информационная система и процессы в студии SOBAKAGAMING носят «ремесленный» характер и завязаны исключительно на одном человеке.

- Разработка и контроль версий:
- Используется локальная среда разработки на единственном ПК. Системы контроля версий (Git, SVN) не используются, либо используются эпизодически, так как разработчик работает один. Весь код хранится локально, бэкапы делаются нерегулярно (на внешний диск «по настроению»).
- Сборка проекта (билд) происходит на том же компьютере.
- Публикация и дистрибуция:
- Процесс заточён под портал Яндекс Игры. Разработчик вручную загружает собранную HTML5-версию игры в личный кабинет Яндекс.Дистрибьютора.
- Отсутствует какая-либо автоматизация выкладки обновлений.
- Учет пользователей и монетизация:
- Статистика по игрокам (количество сессий, донаты) отслеживается вручную через стандартную аналитику Яндекс Игр (или вовсе «на глаз»).
- Данные о доходах приходят на почту, разработчик вручную заносит их в таблицу Excel для отслеживания общей прибыли.
- Управление задачами:
- Все задачи (что нужно доделать, какие баги исправить) хранятся в голове у разработчика, в блокноте или в стикерах на мониторе. Четкого бэклога (списка задач) нет.
- Взаимодействие с внешней средой:
- Связь с редкими тестировщиками или контрагентами (например, фрилансер-художник) происходит через мессенджеры. Передача файлов — по почте или ссылками на облако.

Ключевые проблемы текущей системы:

1. Критическая зависимость от одного человека (Bus Factor = 1). Если с единственным разработчиком что-то случится или сломается его ПК, бизнес и исходный код будут потеряны безвозвратно.

2. Невозможность масштабирования. Процессы, работающие для одной-двух игр на Яндексе, станут катастрофой при выходе на Steam и Google Play. Для Steam нужны совершенно другие билды, версионность, описание, сообщества — все это невозможно делать вручную и держать в голове.

3. Разрозненность данных. Информация о пользователях, доходах и ошибках разбросана по разным кабинетам и почте. Невозможно быстро получить сводку о состоянии дел.

4. Отсутствие контроля версий. При разработке сложного проекта (под Steam) невозможно будет откатиться к старой версии, если новая «сломает» сохранения игроков, так как вся история изменений не хранится.

2. Цели реинжиниринга

Реинжиниринг информационных процессов студии SOBAKAGAMING необходим для достижения следующих бизнес-целей:

1. Обеспечение безопасности и сохранности активов. Создание надежной системы хранения кода и артефактов (билдов) для защиты интеллектуальной собственности.

2. Подготовка к мультиплатформенности. Наладить процессы, позволяющие собирать и публиковать игру одновременно на несколько платформ (Яндекс, Steam, Google Play) без увеличения трудозатрат в 3 раза.

3. Повышение качества продуктов. Внедрение системы учета багов и пожеланий для более структурированной доработки игр.

4. Автоматизация рутины. Освободить время разработчика от ручного заполнения таблиц и выкладки обновлений для концентрации на создании игрового контента.

3. Общие предложения по реинжинирингу (TO-BE)

Предлагается переход от «кустарной» разработки к профессиональным методологиям и инструментам, адаптированным под нужды соло-разработчика.

• По функциям:

• Система контроля версий (VCS): Внедрение Git. Даже для одного разработчика это необходимо как «машина времени» для кода. Репозиторий будет храниться в облаке (GitHub, GitLab). Это решает проблему бэкапов и позволяет работать с разных устройств.

• Автоматизация сборки (CI/CD): Появится функция автоматической сборки проекта под разные платформы (Windows для Steam, APK для Google Play, HTML5 для Яндекс) одним нажатием кнопки. Например, при пуше кода в определенную ветку на GitHub, сервис (GitHub Actions) сам собирает проект и готовит его к выкладке.

• Трекер задач: Внедрение простого таск-менеджера (Trello, YouGile, Kaiten). Появятся доски с колонками: «Идеи», «В разработке», «Тестирование», «Готово». Это структурирует процесс разработки и не дает забыть о найденных багах.

• Централизованная аналитика: Подключение единой системы

аналитики (например, Firebase для мобилок и Steamworks для ПК), которая будет собирать данные в одном окне, а не в разных кабинетах.

• По архитектуре и технологиям:

• Инфраструктура: Переход от одного ПК к комбинации: локальная машина (для разработки) + облачные сервисы (GitHub для кода, Яндекс.Облако/Google Cloud для хранения билдов).

• Инструментарий:

• Git (система контроля версий).

• GitHub/GitLab (удаленное хранилище кода).

• Unity Build Automation / GitHub Actions (инструменты автоматической сборки).

• Steamworks SDK и Google Play Console API (для автоматизации выкладки).

• Trello/Youtrack (ведение задач).

• По данным:

• Структура хранения: Переход от хаотичных папок на диске C:\ к структурированному репозиторию в Git. Вся история изменений кода будет сохранена.

• Метаданные: Введение файлов конфигурации (например, .json), которые хранят настройки игры отдельно от кода. Это позволит менять параметры (цену, описание) для разных магазинов без перекомпиляции всего кода.

• По интеграции:

• Новая информационная система (связка Git + CI/CD + Магазины) должна обеспечить бесшовный обмен данными между этапами:

• Репозиторий -> Система сборки: При обновлении кода автоматически запускается сборка.

• Система сборки -> Магазины приложений: Готовый билд должен автоматически отправляться в личные кабинеты Steam и Google Play (в режим черновика или закрытого тестирования).

• Магазин -> Аналитика: Данные о продажах и поведении пользователей должны стекаться в единую аналитическую панель.

• По компетенциям (переобучение):

• Разработчику (как единственному сотруднику) необходимо переобучиться работе с новым стеком. Это потребует времени, но это инвестиция в будущее:

• Изучить основы Git (в рамках 2-3 дней).

• Настроить CI/CD пайплайн (потребуется изучение документации, возможно 1 неделя).

• Освоить интерфейсы Steamworks и Google Play Console для публикации (отличаются от Яндекс Игр).

4. Ожидаемый результат (TO-BE)

После внедрения предложенных изменений работа студии SOBAKAGAMING изменится кардинально:

1. Безопасность: Код игры хранится в облачном репозитории. Даже если

сгорит единственный ПК разработчика, новый компьютер настраивается за час простой командой git clone. Страх потери данных исчезает.

2. Мультиплатформенность: Разработчик пишет код один раз. Нажав одну кнопку (запустив скрипт), он получает три разных билда: под Windows, Android и Web. Ручная сборка и переделывание уходят в прошлое.

3. Скорость выхода обновлений: Если игрок на Steam нашел баг, разработчик правит код, пушит в Git, и через 30-60 минут (пока идут автоматические тесты и сборка) новая версия уже доступна игрокам. Разработчик не отвлекается от написания кода на рутинные действия.

4. Контроль: Разработчик заходит в Trello и видит точный план работ на неделю. Он заходит в единую панель аналитики и видит, сколько денег принесли вчера Яндекс Игры, а сколько Steam, не открывая 10 вкладок браузера.

5. Предварительная оценка рисков

1. Временные затраты на обучение (риск «простоя»). Соло-разработчику придется потратить от 2 до 4 недель на настройку всей инфраструктуры вместо написания нового кода.

• Митигация: Внедрять изменения поэтапно. Начать с Git (это самый простой и нужный шаг), затем подключить трекер задач, и только потом браться за сложную автоматизацию сборки.

2. Непринятие новых инструментов (психологический риск). Разработчик может посчитать, что «овчинка выделки не стоит», и бросить реинжиниринг на полпути, вернувшись к привычному хаосу.

• Митигация: Четко осознавать, что без этих шагов выход на Steam и Google Play будет либо невозможен, либо приведет к огромному количеству ошибок и негативных отзывов от игроков.

3. Сложность переноса старых проектов. Внедрить Git для новой игры легко, но переносить текущий код старой игры (которая уже на Яндексе) в Git с сохранением всей истории может быть сложно.

• Митигация: Для старых проектов можно просто начать использовать Git с текущего момента, не таща за собой историю (инициализировать репозиторий в папке с проектом сейчас).

4. Финансовые риски. Хотя большинство инструментов (Git, Trello, GitHub для публичных репозиториев) бесплатны, для автоматизации сборки и хранения больших билдов могут потребоваться платные подписки (например, платные минуты для GitHub Actions).

• Митигация: Использовать бесплатные квоты, которых для соло-разработчика обычно хватает с головой. Следить за расходами в облаке.

Заключение

Для студии SOBAKAGAMING реинжиниринг информационных процессов является не просто желательным улучшением, а критическим условием для выживания при переходе на новые рынки. Внедрение систем контроля версий, автоматизации сборки и управления задачами позволит превратить студию из «одного человека с компьютером» в профессиональную команду (даже если в ней по-прежнему один человек), способную конкурировать в

жестких условиях магазинов Steam и Google Play. Это переход от количества потраченного времени к качеству результата и надежности бизнеса.