Разработка на платформе – как достичь эффективности



Максим Цепков

IT-архитектор и бизнес-аналитик

Навигатор и эксперт по миру Agile,
методов самоуправления и моделей soft skill

mtsepkov mtsepkov.org

Analyst Days | Москва
20-21 ноября 2026

Немного обо мне

Создание и внедрение больших корпоративных систем (30+ лет)

- Знание практик операционного управления и ведения проектов в коммерческих и государственных организациях и банках
- Опыт управления проектами в IT: от инженерного подхода и PMBOK – к современным Agile-методам (с 2007 года)
- Опыт перестройки организаций при внедрении систем

Навигация в менеджменте цифрового мира

- Agile и самоуправление: бирюзовые организации, холакратия и социократия (**книга**, статьи и выступления)
- Модель спиральной динамики (с 2013) и другие модели soft skills, **модели личности** (**книга**) и **самоопределения** (**книга**)

Корпоративная культура Китая – лидера новой волны технологических революций (книга)

На моем сайте mtsepkov.org – выступления, блог и много другого



О чем будет рассказ?

Платформа – что это и как она рождается?

- Если в компании разрабатывают несколько приложений, то естественное желание – **не повторять разработку** интеграции и других общих частей
- Цель: масштабирование и низкий порог разработки новых приложений
- Эволюция: **общая библиотека** → **шаблон приложения** → **платформа**
- Или революция: сначала – каждый по-своему, потом – стандартизация
- Этот путь проходят вендоры и крупный inhouse, а с развитием вайбкодинга может стать актуальным для небольших компаний

Я проходил этот путь много раз и поделюсь опытом

- Архитектура: что включать в платформу и как обобщать частные решения
- Организация: взаимодействие команды платформы и команд приложений

Что хотелось бы повторно использовать?

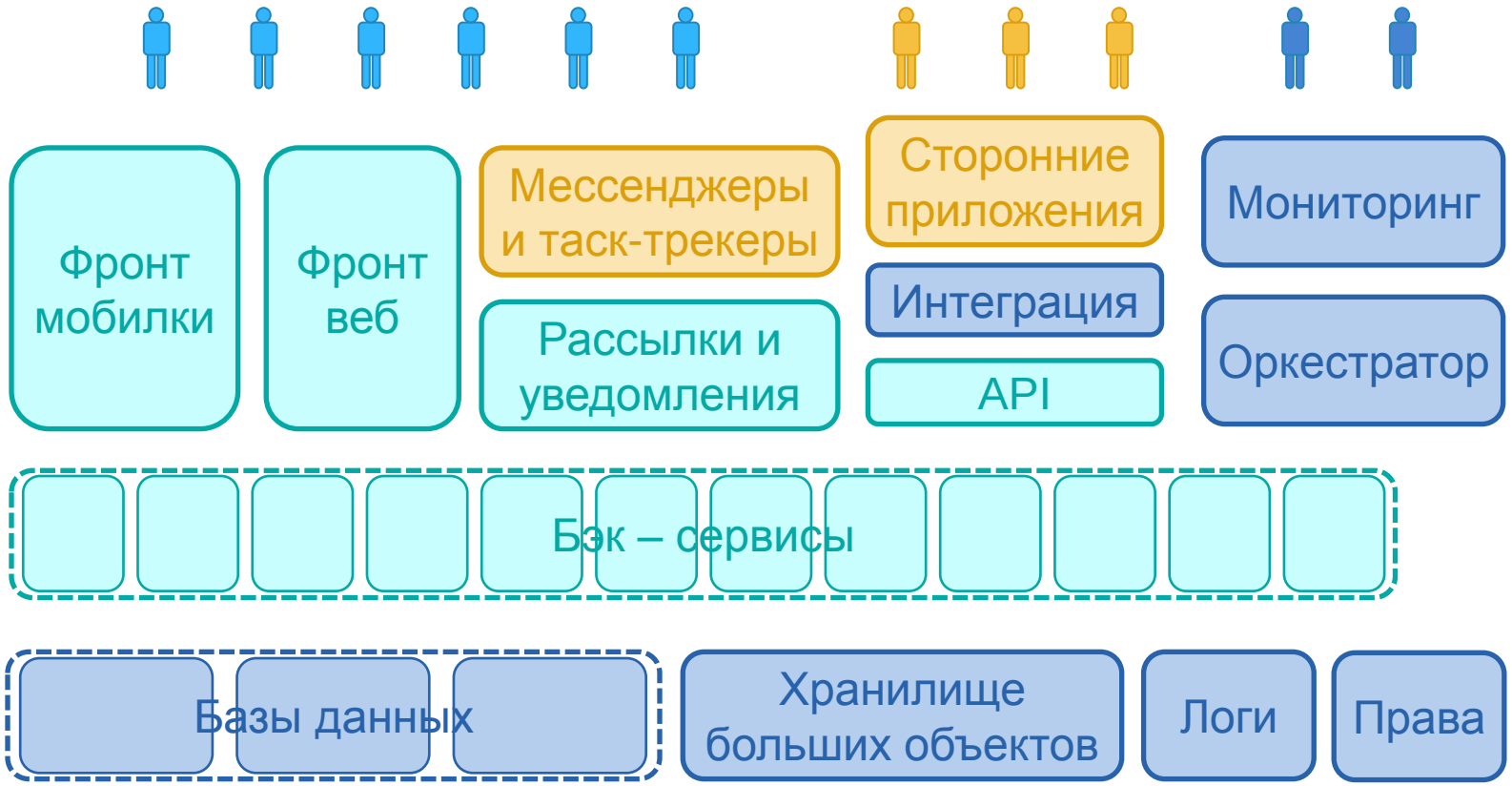
- Таких частей много, и они – разного масштаба
  - Работа с текстами: сумма прописью, упаковка и распаковка xml и json, календарь
  - Сообщения: шаблоны, перекодировка ключей в названия, управление рассылкой
  - Запись лога событий и управлением логом, подключение к мониторингу
  - Интеграция: запись и чтение очередей, включая ожидание события, большие объекты
  - Выполнение SQL-операторов, кэш сущностей, ORM
  - Действия по расписанию с их настройкой, учитывающей взаимосвязи
  - Шаблоны отчетов, печатные формы
  - Система прав с админкой
  - Workflow для документов
- Взять не просто, **нет изоляции**: в сумме прописью – кэш названий валют
- Хочется не взять на старте, а получать новые функции: новый мессенджер для уведомлений или создание задачи в jira по ошибке в приложении

Архитектура – что поддерживает платформа?

Состав современного приложения

- фронт + бэк + БД – классика
- API + интеграция для сторонних приложений
- Рассылки, уведомления для включения в инфраструктуру
- Логи и мониторинг
- Оркестрация
- ИИ в сервисах или через интеграцию
- и многое другое…

И везде есть части, которые интересно повторно использовать



От библиотеки к комплексной платформе

- Легкий вариант платформы – библиотека отдельных компонентов: права, авторизация, логи, ORM, кэш, SQL-запросы, интеграция, уведомления
- Для легкого старта новой разработки – шаблон приложения и **harness**
- В платформа компоненты собраны в **интегрированную конструкцию**:
  - Устойчивая к сбоям интеграция сервисов, включая доступ к базовым справочникам
  - Взаимодействие ORM, кэша, SQL массового изменения, транзакции и откат по ошибке
  - Инфраструктура: подключение логов к единому мониторингу, обновление приложения
  - Вертикальная интеграция: от сущности с БД до REST-API и формы на интерфейсе
  - Однородный пользовательский опыт для всех приложений на платформе
  - Типовые бизнес-задачи: печатные формы, workflow документов, действия по расписанию, BPMN-движок, учет, простые справочники, конструктор отчетов…
- Аналитику необходимо знать возможности и ограничения компонентов и точек расширения, **чувствовать выход за рамки платформы**
- А в новой реальности – уметь решать типовые задачи ИИ-разработкой

Вопросы вертикальной интеграции решений

- Есть ли в платформе понятие бизнес-объекта и на каком уровне?
  - Есть ли контроль бизнес-объектов на уровне интеграции и какой именно
  - Какова поддержка вертикальных решений: как атрибут появится в БД, API и на форме
  - Поддержаны ли русские названия атрибутов для использования на формах
  - Есть понятие пользовательского названия объекта для сообщений об ошибках и др.
  - Есть ли списки значений, и в каком объеме, как из списка сделать справочник?
- Как устроена интеграция и асинхронная работа?
  - Каково состояние после ошибки, что сделает rollback, а о чем надо заботиться?
  - Как показать изменения документа, если у тебя CQRS и два сервера с репликацией?
  - Как отслеживать прохождение сквозных процессов через системы по логу?
- Фронт и бэк:
  - Как синхронизовать бизнес-логику проверки значений и т.п.
  - Гигиена работы с таблицей: фильтры, сортировки, экспорт в Excel, буфер обмена…

23 analyst days

О базовых составляющих приложений

Проектируйте хорошую интеграцию

Главное – не технологии, а **устойчивость и быстрое решение инцидентов**

- Хорошая админка, чтобы видеть проблемные сообщения, уметь их повторно обрабатывать и посылать, быстро искать проблемные объекты
- Устойчивый протокол: идемпотентные операции, учет последовательности
  - CRUD и REST — не идемпотентны, нужен **UPSERT**
  - Нет протоколов exactly once, потери или дубли неизбежны
  - Кейс: сверка на хранилище в большой логистической компании нашла тысячи коробок
- Работа с ошибками на большом потоке сообщений
- Обязательна возможность технологичной сверки и устранения проблем

Часть вопросов решает платформа, другие оставлены приложениям



Мои выступления про интеграцию: Что такое – хорошая интеграция на Saint Highload-2021 и Обеспечиваем устойчивость интеграции на SQAdays весной 2025

Логи и мониторинг: проектирование и анализ

- Логи – не только разбор инцидентов, это еще мониторинг и анализ
  - Как идут сквозные бизнес-процессы через несколько систем, где задержки обработки
  - Как идут длинные процессы, связанные с бизнес-сделками
  - Нет ли потерь и зависших объектов, выпавших из внимания
- Разбирать инциденты и вести анализ должно быть удобно с интерфейса
- Необходимо гибко управлять накоплением лога, а также его отражении в дашбордах мониторинга (Grafana, Prometheus и других)
- Для анализа эргономики интерфейса часто интересны действия пользователей, но их логирование часто не предусмотрено
- Анализ документов дает статику, а анализ логов – динамику работы
- **Аналитику** – разбираться в устройстве и учитывать при проектировании

Организация взаимодействия сервисов

- На платформу может быть возложена оркестрация сервисов, роль оркестратора может выполнять BPMN-движок, например, Camunda
- Возможна хореография, когда сервисы передают объект по цепочке
- Бывают композитные решения: сервисы владеют объектами (клиентами, счетами и др.), и владелец подтверждает операцию через платформу
- Платформа может **обеспечивать доступ к мастер-данным**: клиентам, каталогу продуктов, тарифным планам и так далее
- Платформа может контролировать структуры данных при передаче



Перегрузка платформы функционалом приводит к тому, что **команда платформы становится узким местом** для прикладной разработки

Платформа и домены

- Платформа может давать стандартную точку входа ко многим сервисам
  - Почтовые уведомления, мессенджеры и подобные инфраструктурные сервисы
  - Общезначимые справочники и НСИ: клиенты, валюты, продукты, тарифы
  - Отраслевые внешние сервисы – платежные гейты, отсылка отчетов регулятору и т.п.
  - Общезначимые бизнес-сервисы, такие как 1С бухгалтерия или корпоративный BI
- Во многих случаев платформа дает интеграционный гейт, за которым располагается доменный сервис **со своей командой**
- Если запрос требует расширения API, то взаимодействуют три команды:
  - Прикладная команда разработки
  - Команда платформы
  - Команда общезначимого сервиса, или админ/devops, который его поддерживает
- **Аналитику** надо представлять карту ответственных и взаимодействовать

Развитие платформы: поиск новых решений

Новый функционал в платформе

- Всегда возникают кейсы, когда функционала платформы недостаточно, и нужно доработать платформу или сделать специальное решение
- Есть конфликт интересов: прикладной команде надо сделать фичу, а платформа развивается по своей траектории
- Не всегда проблема вскрывается на этапе обсуждения и проектирования, часто реализация уже обещана бизнесу
- Может ли прикладная команда доработать платформу и как?
  - Fork кодовой базы платформы для частного решения, чтобы проверить реализацию
  - Выбор: обобщение или точка расширения в платформе, или оставим частное решение
  - Команда платформы **не должна быть** внутренним вендором – **тормозом** разработки
- **Аналитику надо уметь фасилитировать и договариваться**: поиск решения, разделение ответственности, сроки, бюджеты

Пример: большие документы

- **Проблема**: В конкретном приложении требовались товарные документы в 10000 позиций, при их вычитке на интерфейсе возникали проблемы
- **Уточнение**: пользователю не надо видеть все эти строки, надо уметь найти нужное, сделать полный экспорт в Excel или напечатать
- **Решение**: фильтр по позициям для поиска и специальные механизмы экспорта и печати без полного вывода на интерфейсе

Аналитик прояснил сценарии использования и совместно с разработчиком искал приемлемые решения

Пример: передача большого объекта

- **Задача**: при заказе доставки в магазине покупатель иногда рисует схему проезда, которую надо донести до транспортной службы и курьера
- **Проблема**: заказ проходит несколько систем, интеграция рассчитана на передачу простых типов данных и не сможет передать скан, да и не нужен он складу и офису
- **Решение**: использована существующая система хранения фото товаров – не по назначению, зато это не потребовало отдельной инфраструктуры для редкой задачи

Аналитик фасилитировал поиск существующей инфраструктуры, а затем совместно с командой-владельцем прорабатывал ее использование

Пример: инфраструктура обновления

- **Проблема**: на пилоте нового решения на 100+ серверов магазинов выяснили, что инфраструктура рассчитана на редкие доставки обновлений для базового софта, а обновлять версии надо будет 1-2 раза в месяц
- **Решение**: доставка обновлений средствами репликаций, используемых для обновления каталога товаров, потребовалась доработка для запуска файла с обновлениями и кастомизация системы мониторинга
- **Урок**: разворачивание и **поддержка** новой инфраструктуры часто сильно дороже приспособления на существующей для новых задач

Lifecycle платформы

- Платформу драйвят 1-2 приложения, остальным важна стабильная версия, хотя бывают исключения – массовый переход на новую платформы
- Сама платформа не однородна: есть области развития и стабильные
- При доработке платформы важна обратная совместимость, в том числе – с учетом накопленных данных, когда переименование поля в БД или заполнение нового поля нельзя сделать при простом накате обновлений
- Хорошая практика: **человек из платформы в прикладной команде**
  - Задача: видеть прикладные проблемы и подсказывать решения
  - Очень часто разработка платформы сосредоточена на наращивании функционала, но не видит, что типовые задачи требуют много прикладного кода
- **Аналитик в платформе** должен представлять многообразие применений
- **Прикладной аналитик** должен быть готов к поиску решений
- Отдельная задача – проектирование сценариев через много приложений

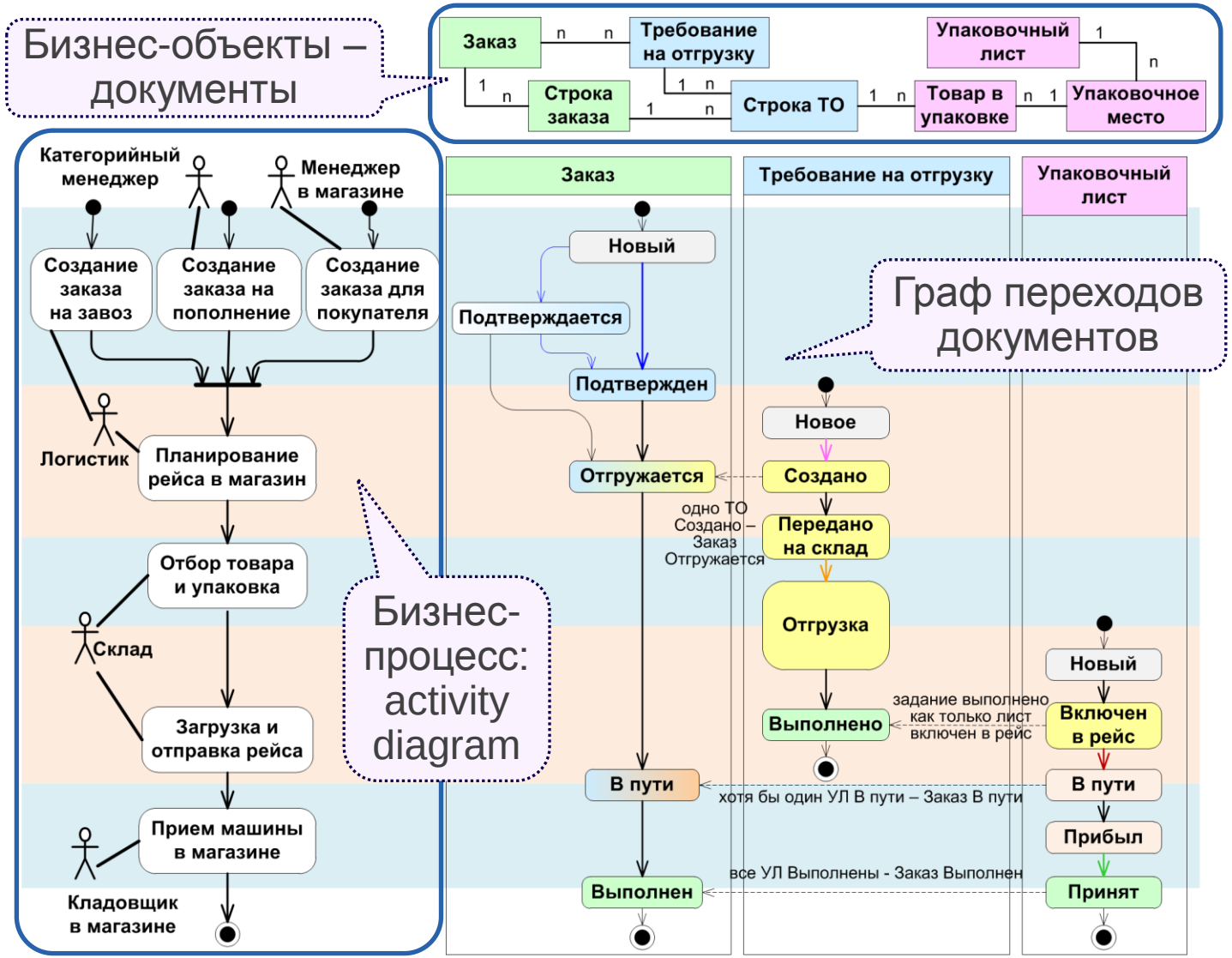
Документооборот и учет в платформе



Такие задачи могут возникать в небольшом количестве сервисов, однако их качественное комплексное решение сделать не просто, поэтому есть смысл вынести его в платформу

Workflow документов

Пример: снабжение магазинов



Что дает платформа?

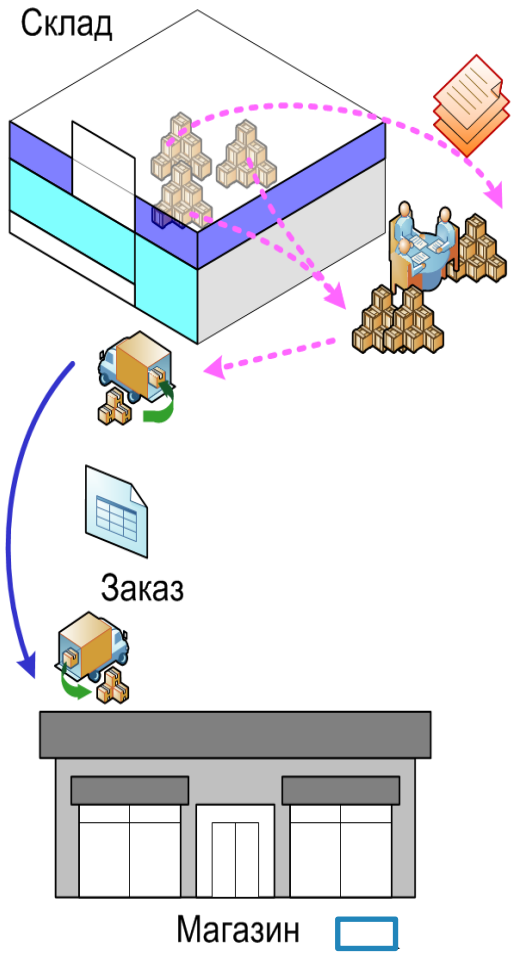
- Для документа описываем граф переходов состояний
- Переход отражает выполнение бизнес-функции или ее шаги
- Состояние определяет набор переходов и других действий и права на их выполнение
- Выполнение перехода – событие
- Ход бизнес-процессов отражается системами мониторинга

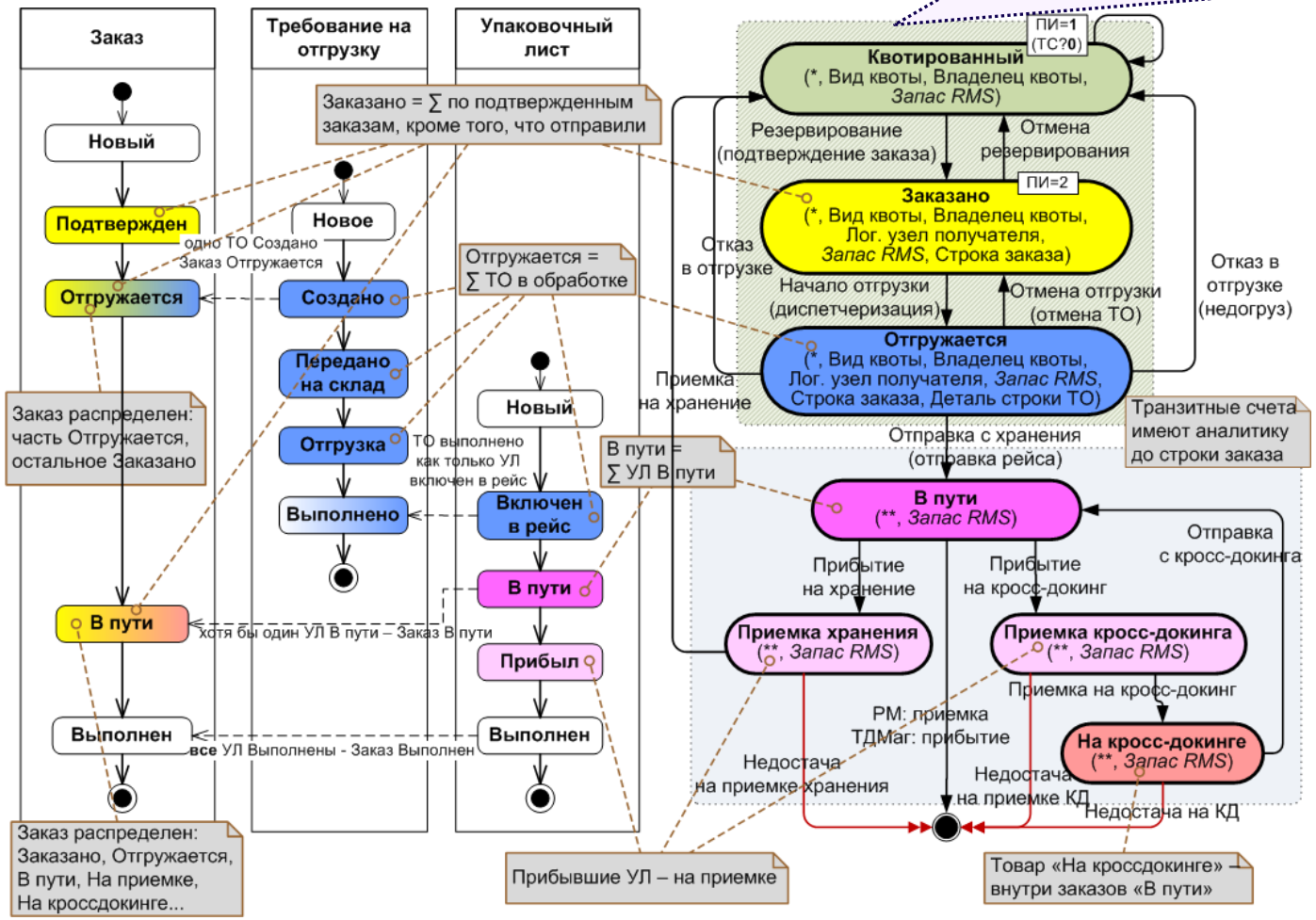
Ценность – платформа дает механизм **комплексного решения**

Подробнее смотри «DDD: модели вместо требований 9 лет спустя» (ЛАФ-2023)

От документов к учету

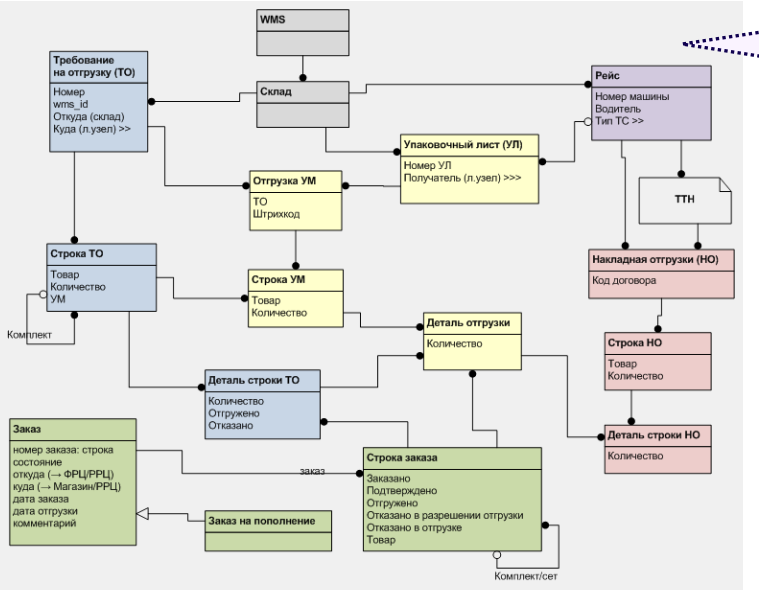
События на переходах документов порождают проводки в учете и меняют остатки. Эффективная реализация учетной машины – не простая задача, и ее полезно сделать в платформе, а не в каждом приложении



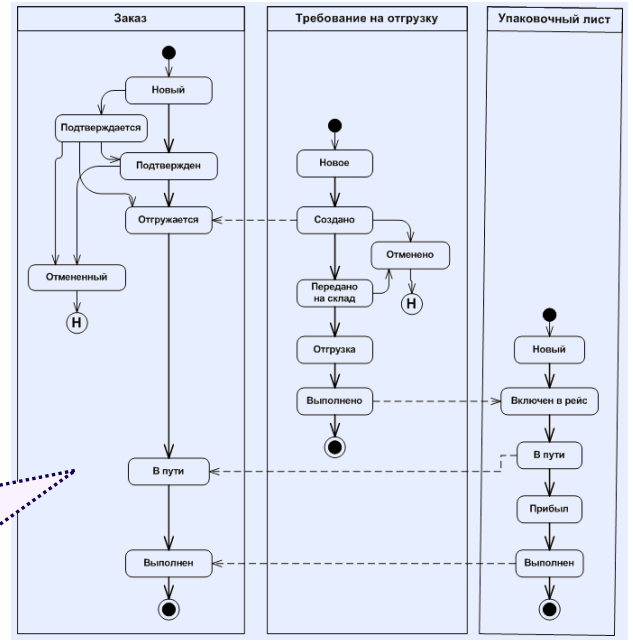


Подробнее об учете – статья «Когда всем понятно» и другие материалы

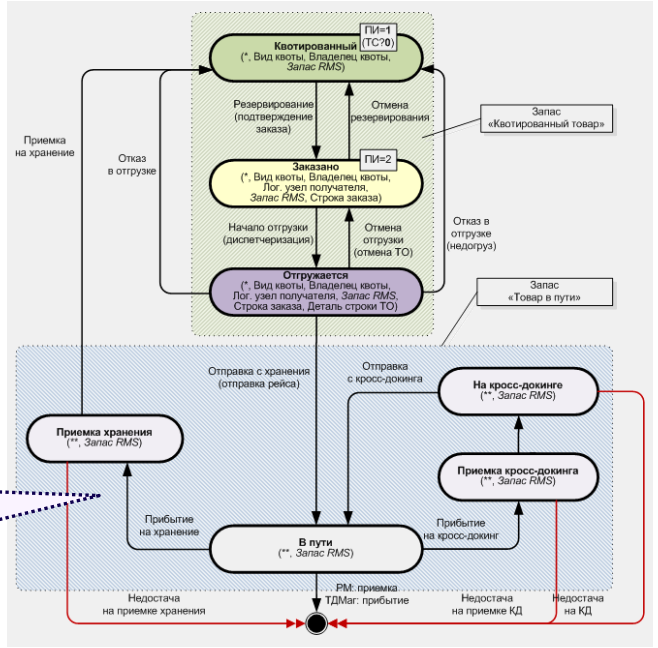
Шаблон для корпоративного приложения



Пользователи создают документы, по необходимости заполняют справочники – используем **объектную модель**



Потом документы исполняют – используем переходы документов между состояниями (State Entity)



При этом меняются учетные данные, которые влияют на исполнение документов и отражаются в отчетах – используем **учетную модель**

Подробнее смотри «DDD: модели вместо требований 9 лет спустя» (ЛАФ-2023) и другие материалы

Оркестрация сервисов и приложений

- Оркестрацию может обеспечить BPMN-движок (Camunda и др.)
  - Для каждой сделки или задачи запускаем отдельный экземпляр процесса
  - Мониторим ход исполнения, проверяем, что нет зависших процессов
  - Сложности – для процессов, которые пошли не по плану, нужна обработка исключений
- Альтернатива – хореография сервисов и отдельные компоненты
  - Запуск задач по расписанию, с учетом фактического исполнения и зависимости
  - Контроль через них зависших процессов – документов в промежуточном состоянии
- Оркестрация в платформе обеспечивает централизованный мониторинг
- Аналитику надо понимать, как принято обеспечивать оркестрацию и учитывать это при проектировании конкретных решений

Примеры отдельных блоков

Отчеты на основе SQL-запроса

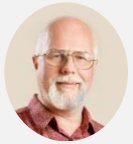
- **Задача**: построение разовых или регулярных отчетов по запросам бизнеса
- **Противоречие**:
  - Отчет нужен быстро, часто для решения разовой проблемы – нет смысла делать фичу
  - Даже разовый отчет повторяется 5-6 раз и просить разработчика – его отвлекать
  - Конструктор отчетов, который сделает сложные запросы – дорог и слишком сложен, обычные пользователи не смогут его использовать, а запросы будут не эффективны
- **Решение**: механизм, который позволяет встроить написанный SQL-запрос в интерфейс пользователя с минимальным описанием (типы параметров)



Решение оказалось актуальным и востребованным, тиражировалось в большом количестве приложений

Типовые решения в платформе

- В любом приложении есть много **небольших справочников**
  - Полезно типовое решение, которое на основе описания структуры обеспечивает CRUD на всех уровнях платформы: БД, бэк, API, фронт
  - Для специальной бизнес-логики – реакция на события
  - Надо обеспечить не только первую генерацию, но и изменения
- Типовые **товарные документы** в торговых приложениях и логистике
  - Обобщенный документ перемещения товара между учетными складами
  - Механизмы настройки для описания связи атрибутов документа с учетом
- Возможны другие типовые механизмы: для комиссий, биллинга, договоров



Задача аналитика: **увидеть повторение** частных решений в разных приложениях и **обобщить** до включаемой компоненты или общего сервиса, в зависимости от того, нужна ли **независимость по обновлениям** функционала

Редкие документы и операции

- В бизнесе очень часто встречаются исключения из стандартного процесса
  - Они могут занимать 1-2-5% случаев, но отличаться громадным разнообразием
  - Они часто связаны с VIP-клиентами их нельзя игнорировать
  - Реализация для каждого случая полноценного решения – очень дорога
- Часто это обнаруживают, распиливая монолит или проводя реинжиниринг: 80-90% сделок сделано в новых красивых решениях, осталась непонятное
- Вариант обобщенного решения или шаблонов разбивается о сложность
- **Решение**: отдельные сервисы или документы работы на микроуровне
  - Придумать простые операции, которые обеспечат обработку редких документов – обычно бизнес-логика собирает из них обработка частых документов
  - Реализовать без жесткого workflow, с ручным отражением проводками в учете
  - Для опытных сотрудников со специальными правами доступа
  - Полезна системой образцов документов для копирования
- Решение не красивое, но без него legacy будет жить вечно…

Итоги: зачем платформа и как с ней жить

- Платформа решает две задачи
  - Эффективное решение сложных задач, обеспечивающие масштабирование
  - Низкий порог для разработки новых приложений
- Всегда будут задачи, для которых функций платформы не достаточно
  - Надо уметь находить решение для таких задач в диалоге с командой платформы
  - Надо уметь обобщать частные решения, развивая платформу
- Вайбкодинг снижает порог разработки не только приложений, но и платформ



Максим Цепков

http://mtsepkov.org

Tg: @MaximTsepkov

Mx: @channel_tsepkov

maks.tsepkov@ya.ru