

Визуальные модели корпоративного приложения

Максим Цепков

Главный архитектор решений [CUSTIS](#)

<http://mtsepkov.org>

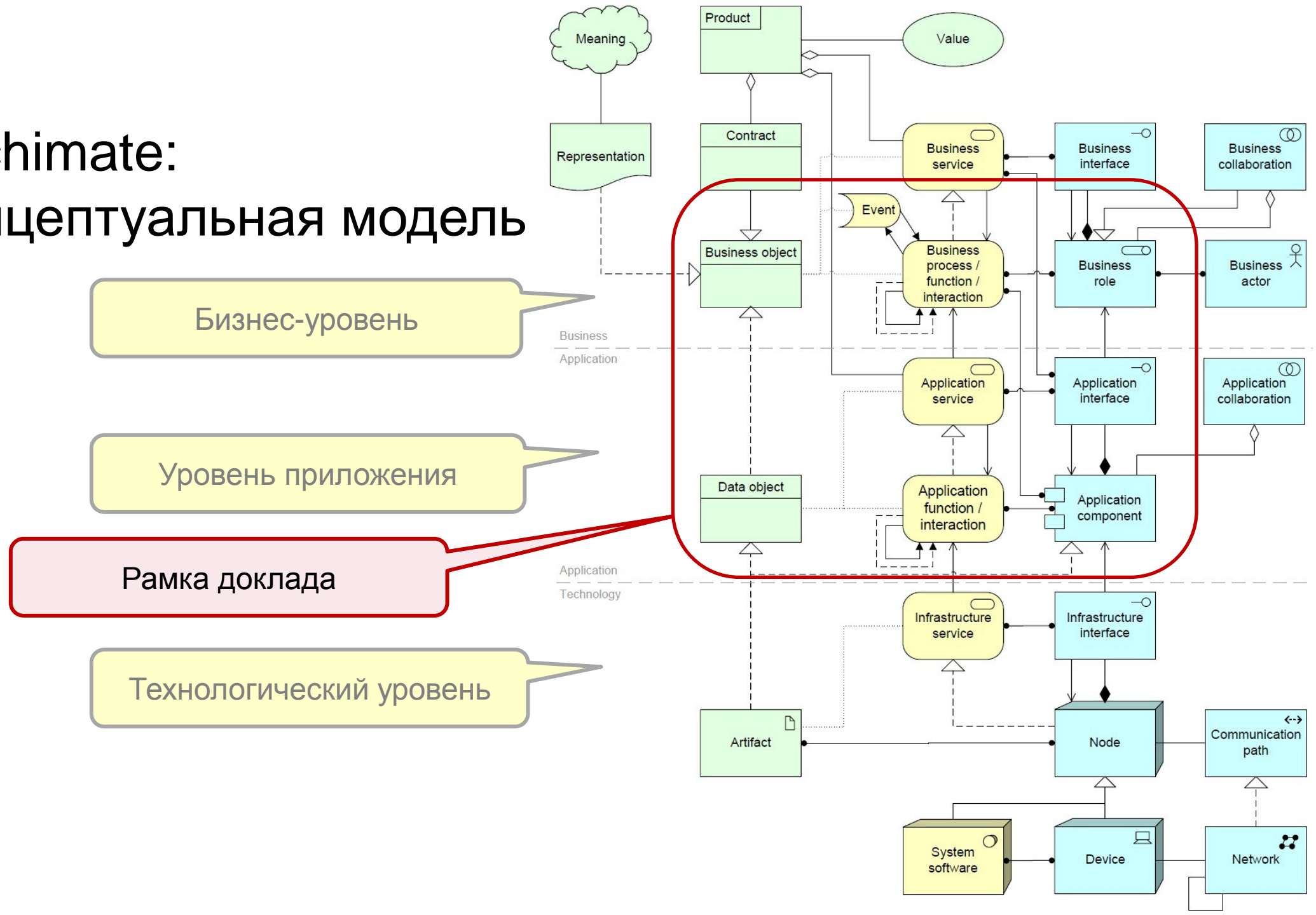
Москва, 1 ноября 2017

Корпоративное приложение – часть архитектуры предприятия

Для проектирования используем визуальные модели:

- Архитектура предприятия в целом и бизнес-уровень
- Метафора системы – выражение системы в целом
- DDD для корпоративного приложения –
шаблон Учетной машины и визуальные представления

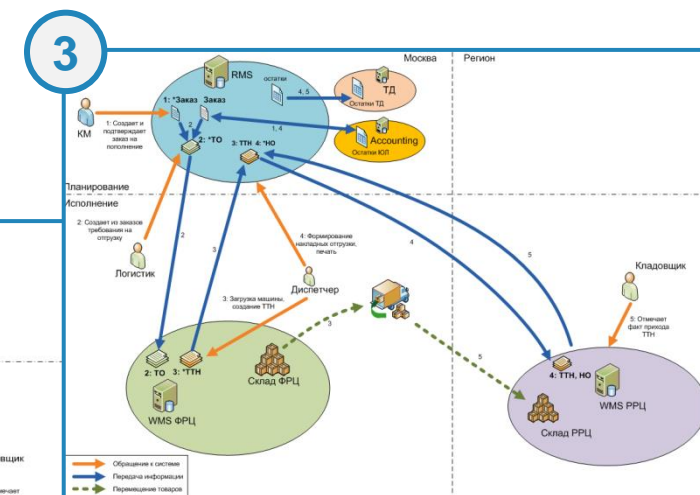
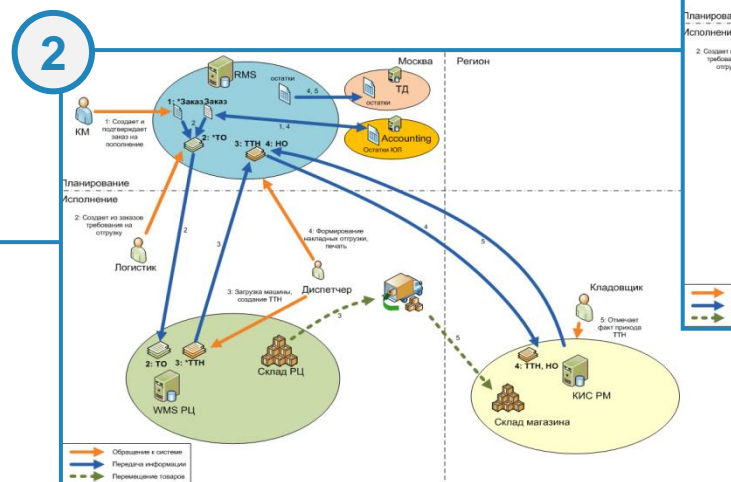
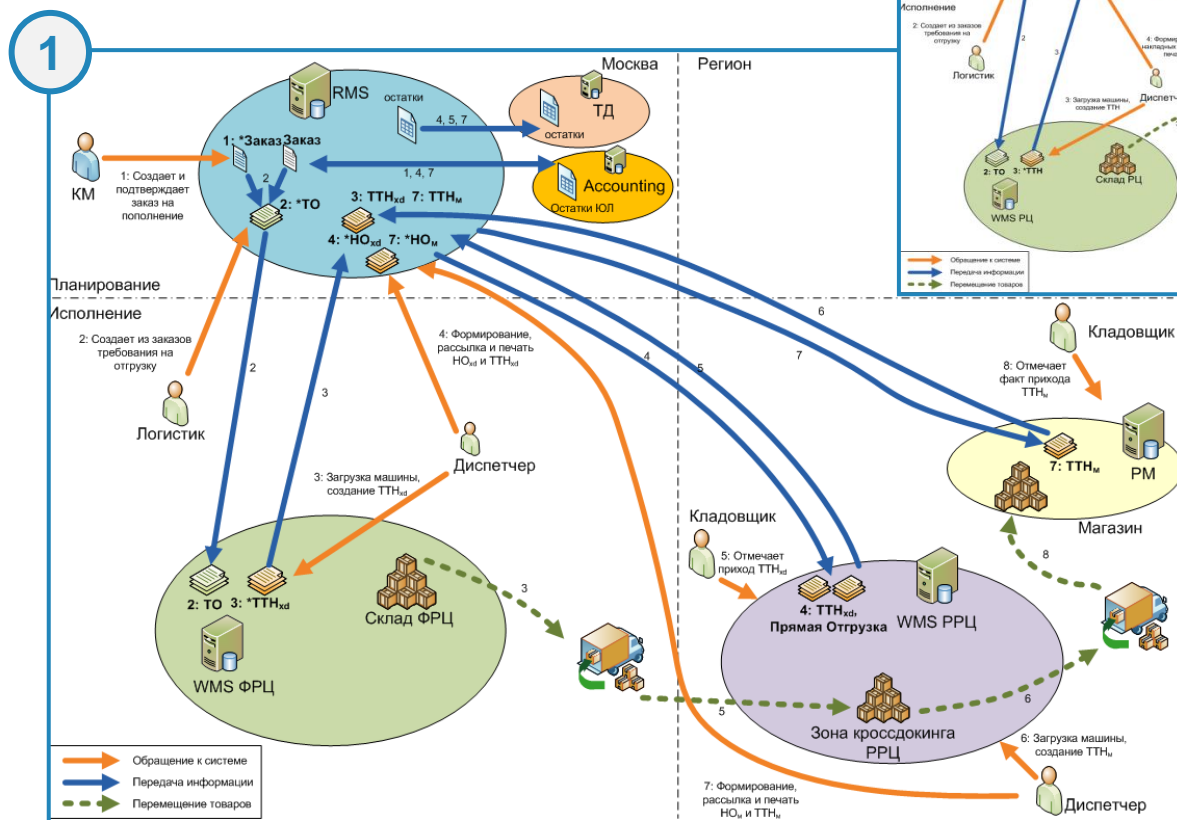
Archimate: концептуальная модель



Архитектура предприятия в целом и бизнес-уровень

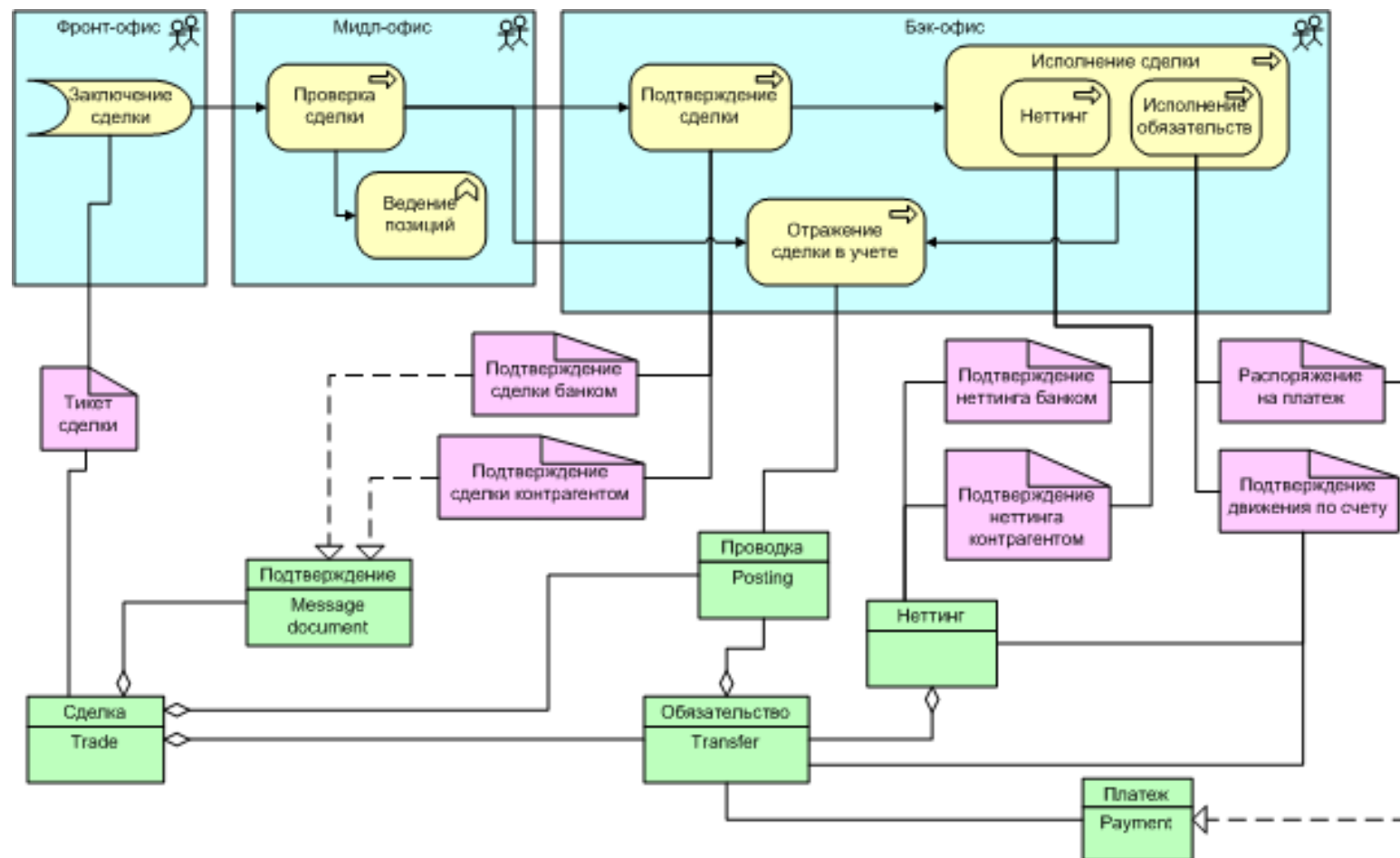
Неформальный бизнес-процесс

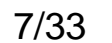
Пример:
снабжение магазинов



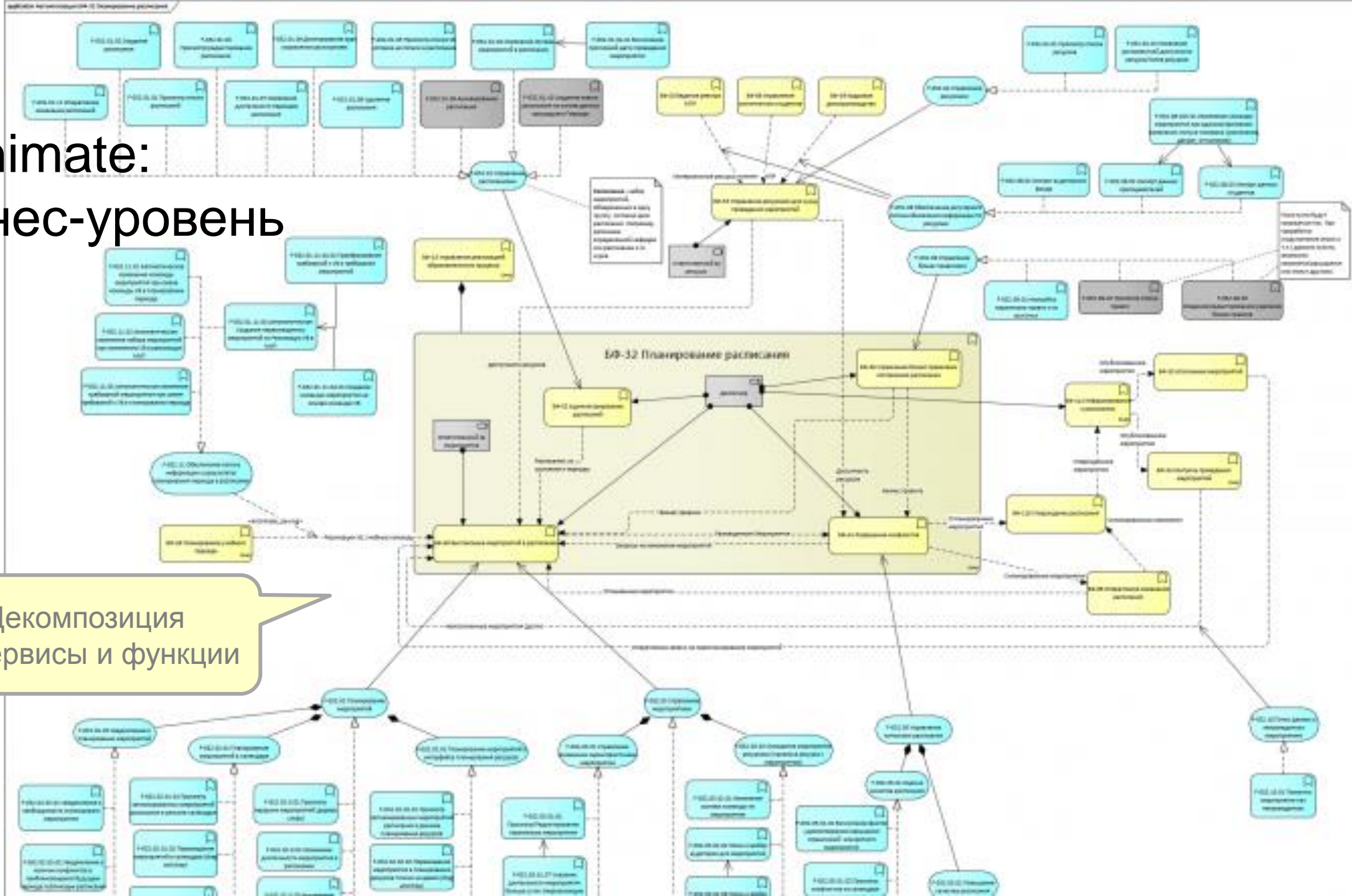
Комплексная схема
3 уровней Archimate,
но в вольной нотации
она понимается **лучше**

Связь бизнес-процессов и объектов приложения





Archimate: бизнес-уровень



Декомпозиция
на сервисы и функции

Метафора системы

Что такое метафора системы

- Метафора – лаконичная концепция, которая много говорит о системе
- Иными словами, проекция (viewpoint), максимально отражающая суть
- Визуальный образ – хорошая метафора
 - Дает цельное представление о системе
 - Обеспечивает взаимопонимание
- Поиск метафоры начинается с vision-проекта
- Визуальная основа – стандартна или придумана

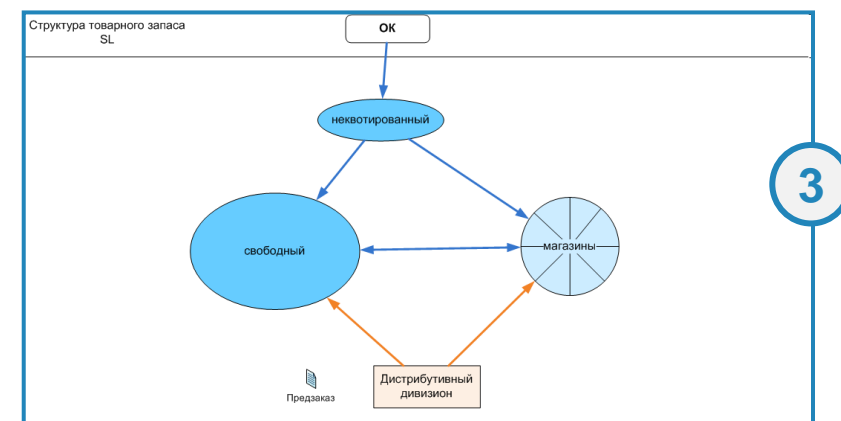
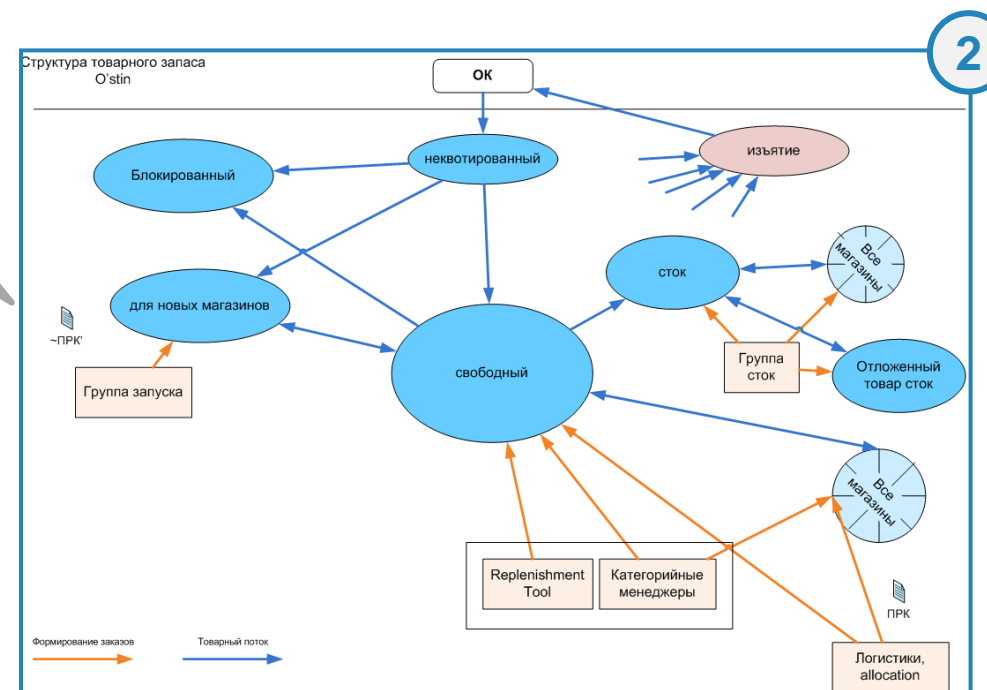
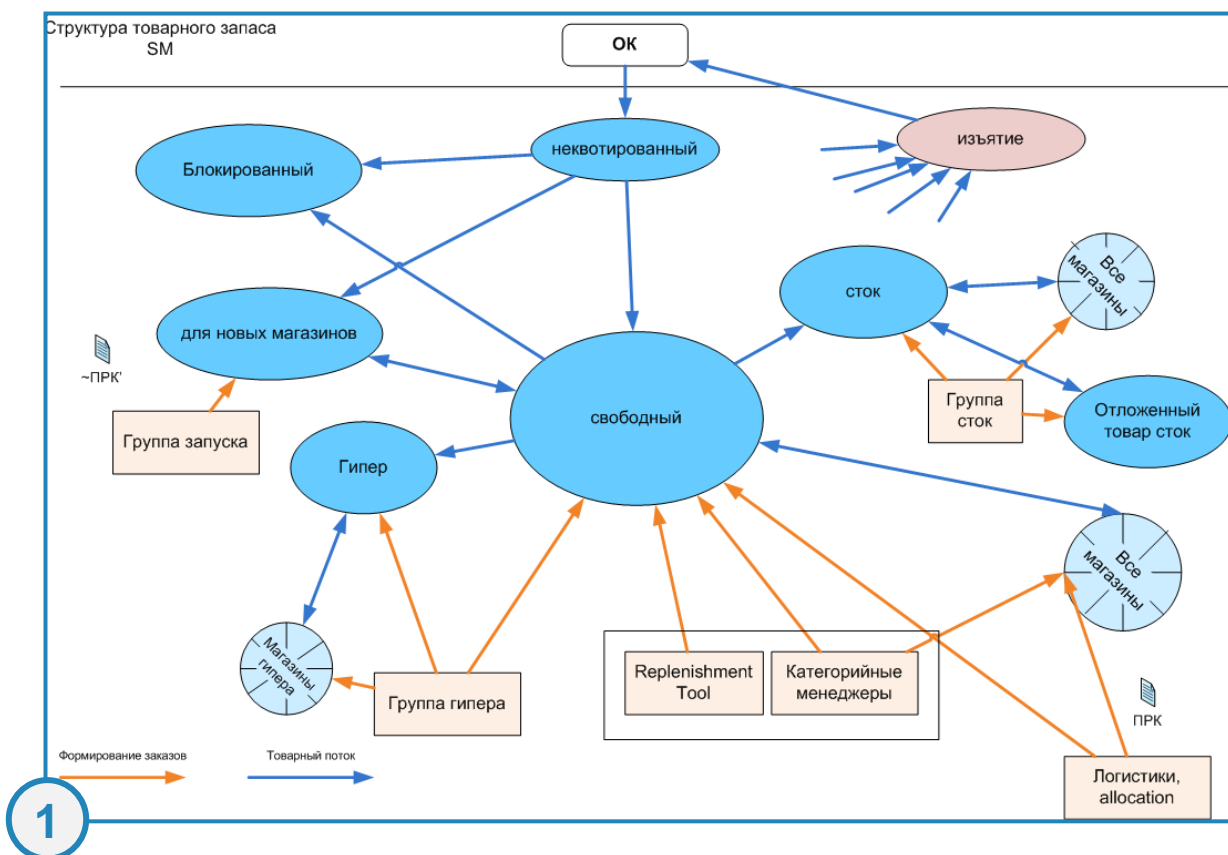


Найти получается не всегда

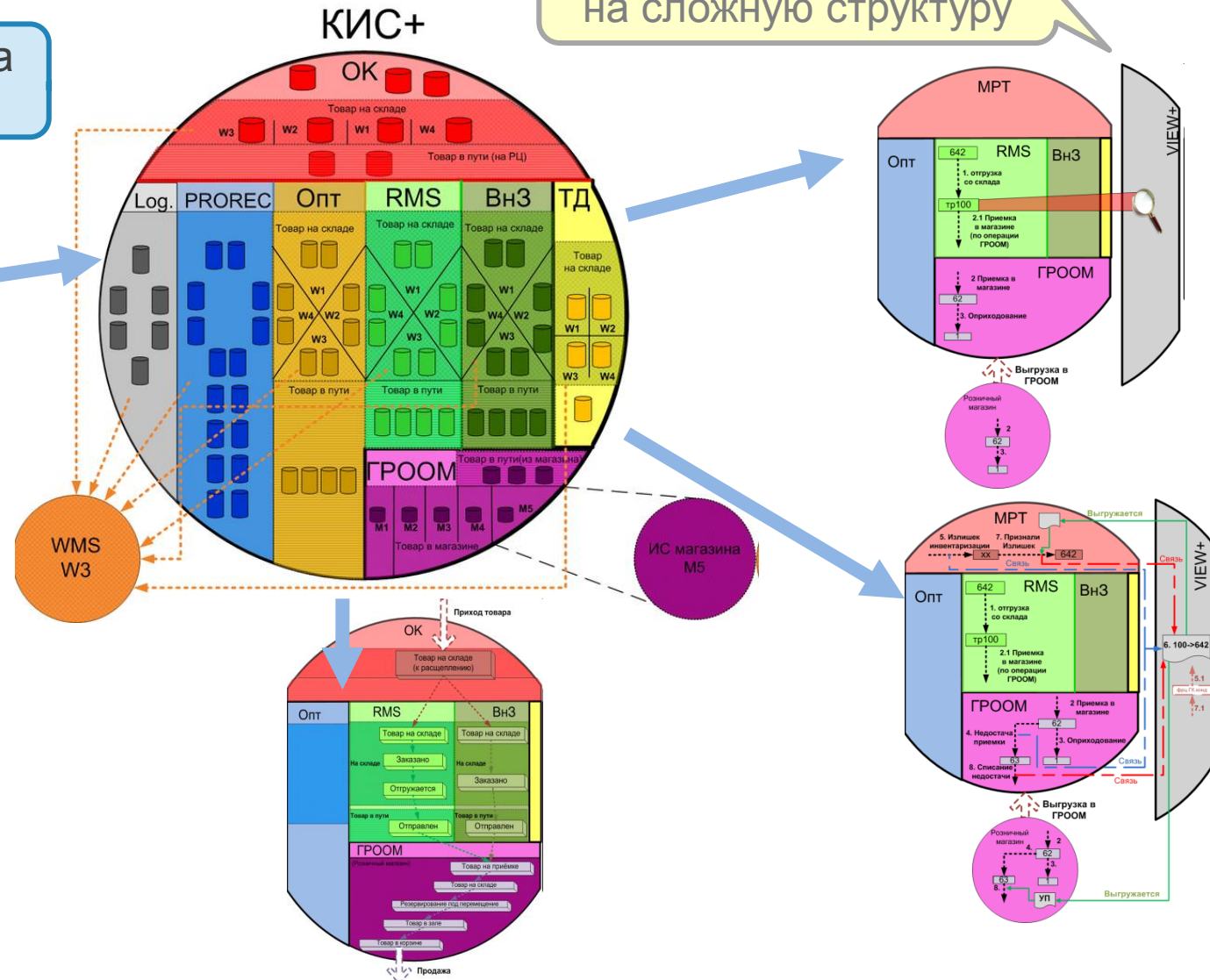
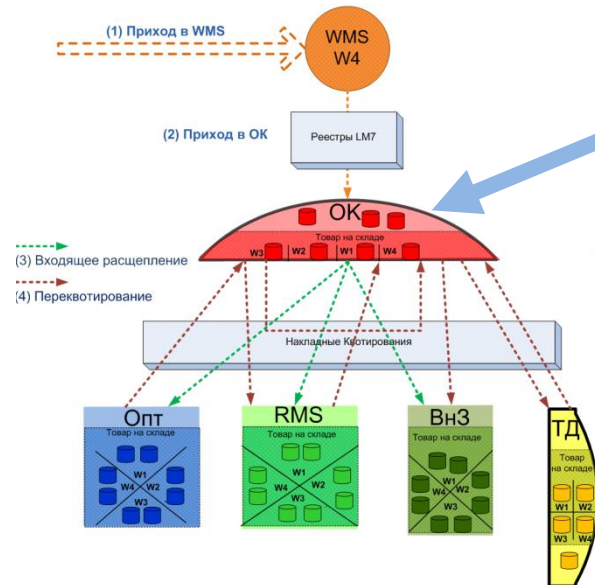
Используем типовые образы

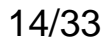
Пример:
деление товара

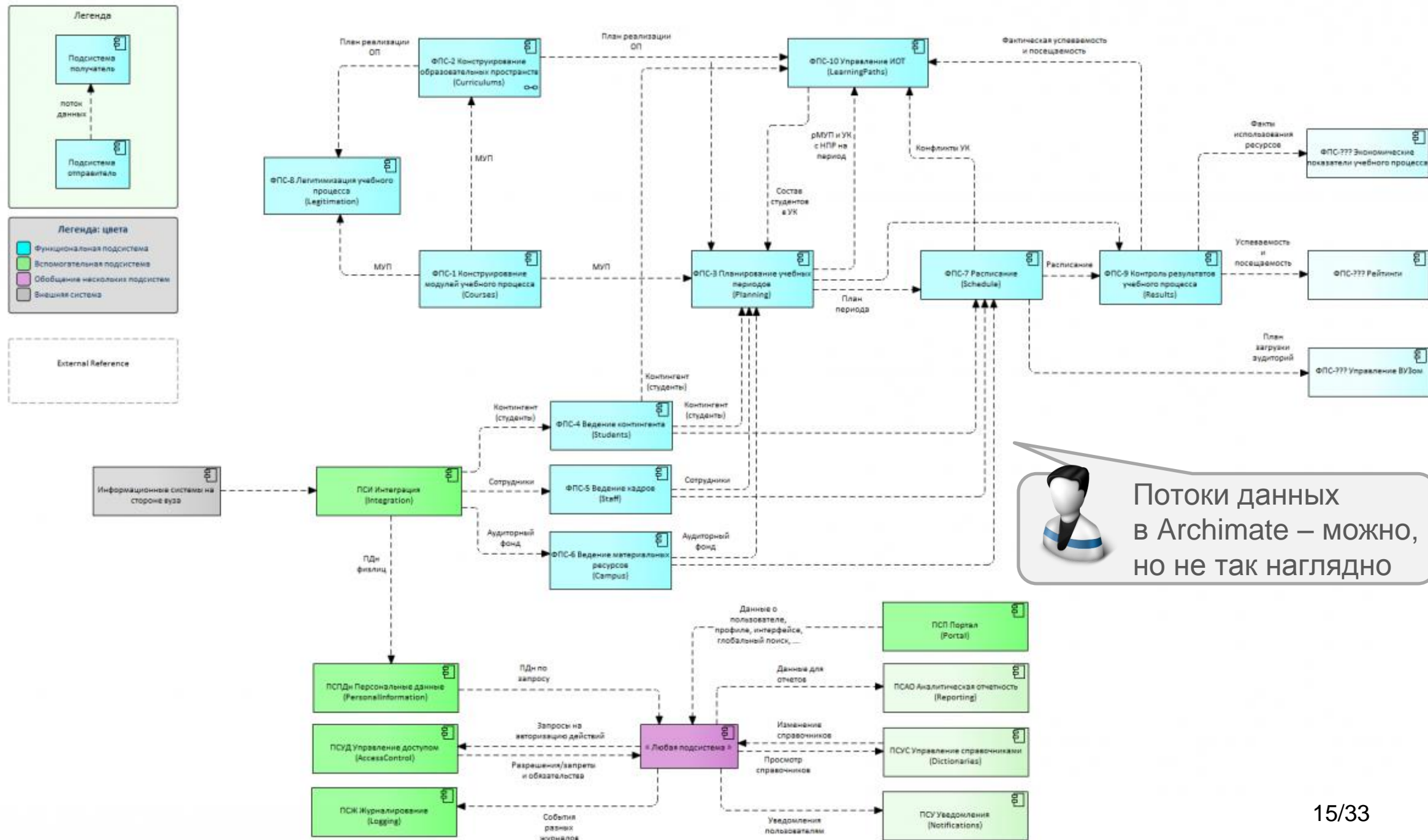
Варианты процессов
в простой нотации,
схемы понимаются на лету



Лупа: смотрим
на сложную структуру



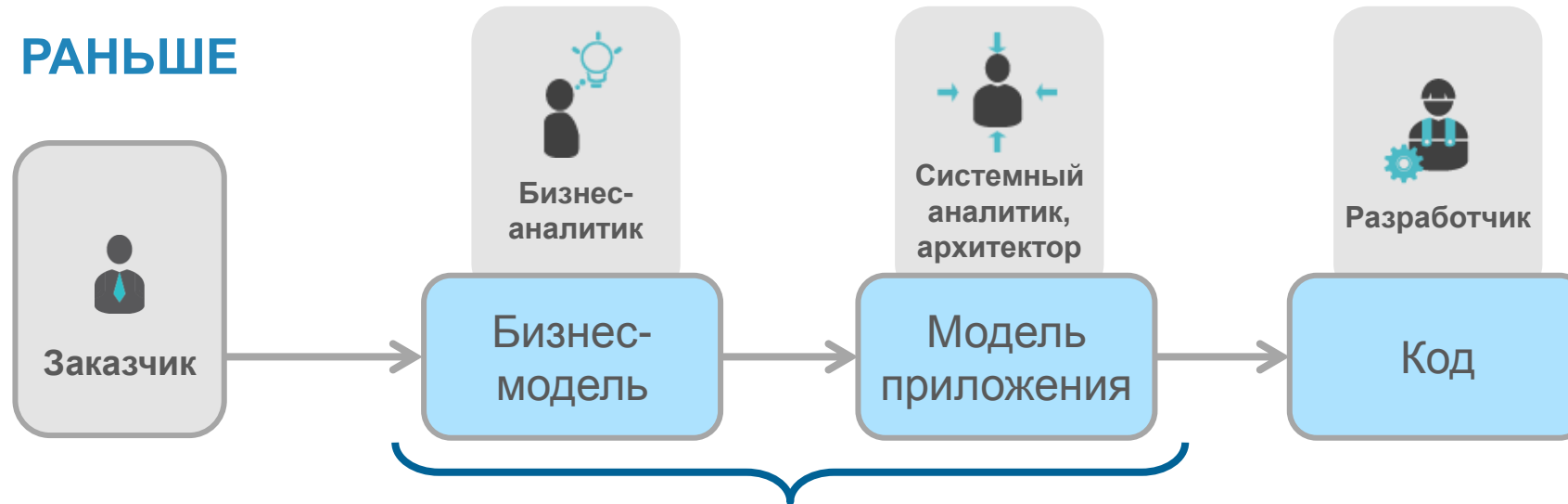




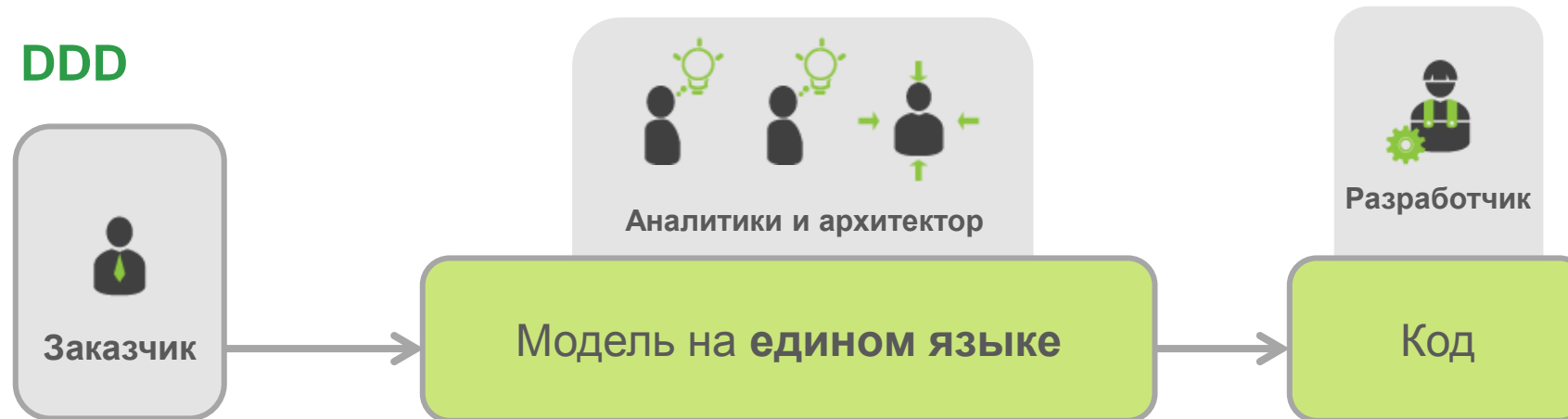
DDD для корпоративного приложения – шаблон Учетной машины

Основная идея DDD

РАНЬШЕ



DDD



Единый язык (ubiquitous language)

- Построен на основе терминов предметной области
- Его понимают ИТ-специалисты и эксперты бизнеса
- На нем описана модель ИТ-системы и ее место в бизнес-процессах



Понятия единого языка: **клиент, накладная, платеж, долг**, – из предметной области

Единая модель

- Аналитик собирает требования и строит модель: сначала – предметной области, затем – системы
- Артефакты модели описывают и систему, и ее использование в бизнес-процессах предприятия
- Разработчик реализует модель
- Артефакты модели можно проследить в коде

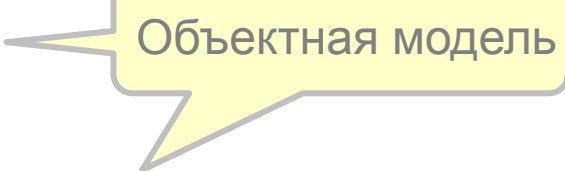
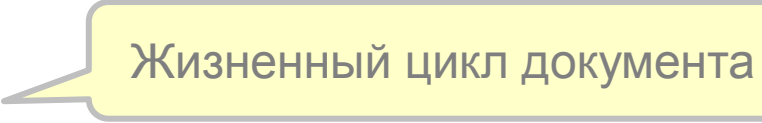



Модель предметной области
становится моделью системы

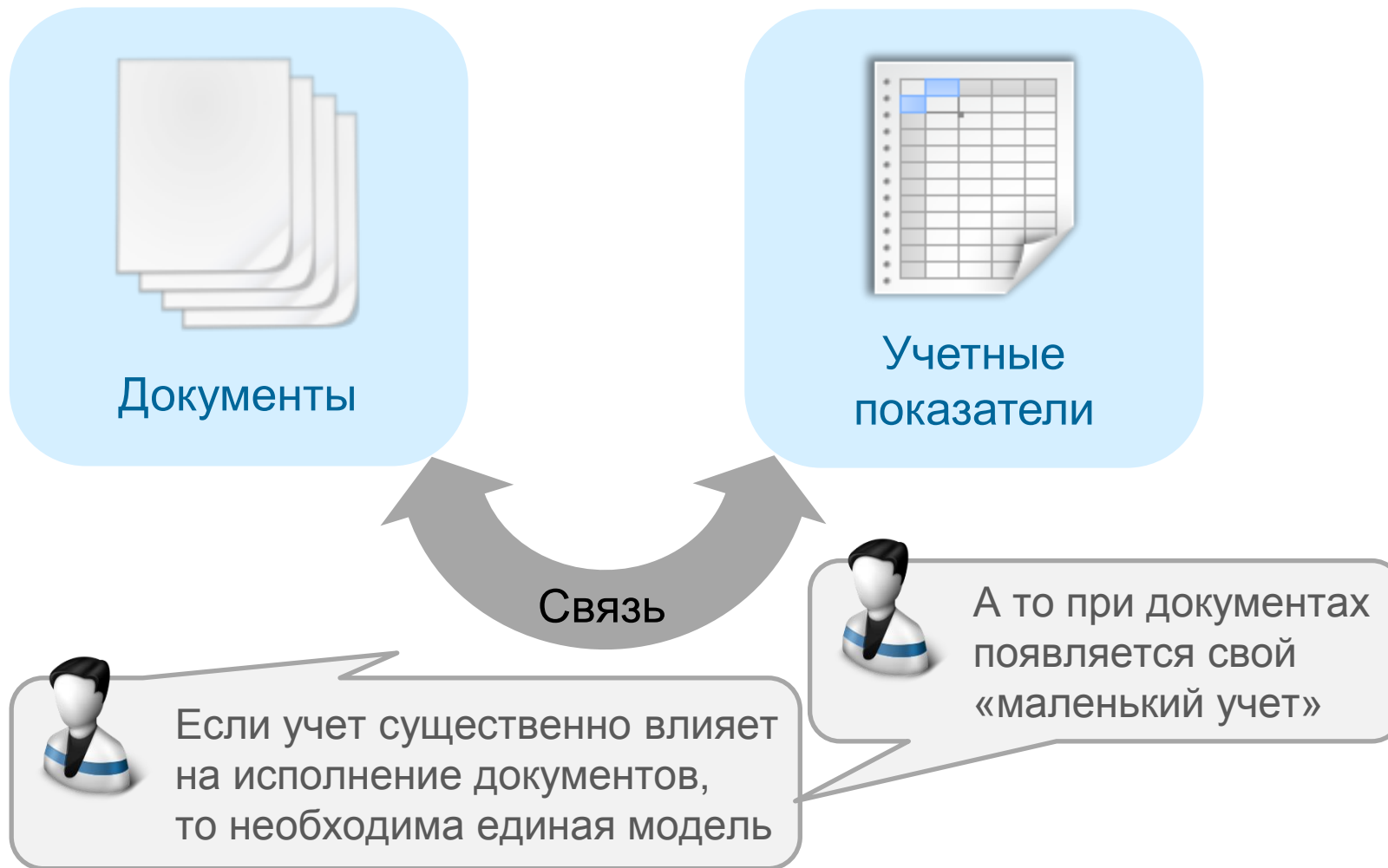
Плюсы единой модели

- 😊 Верификация постановок бизнес-специалистами
- 😊 Общее понимание требований на стороне бизнеса
- 😊 Обсуждение модели бизнесом и ИТ, поиск баланса в сложных решениях
- 😊 Перенос моделей из других предметных областей
- 😊 Бизнес представляет потенциальные возможности системы и сложность различных доработок
- 😊 На этапе эксплуатации – эффективное общение бизнес-пользователей и ИТ без квалифицированных переводчиков-аналитиков

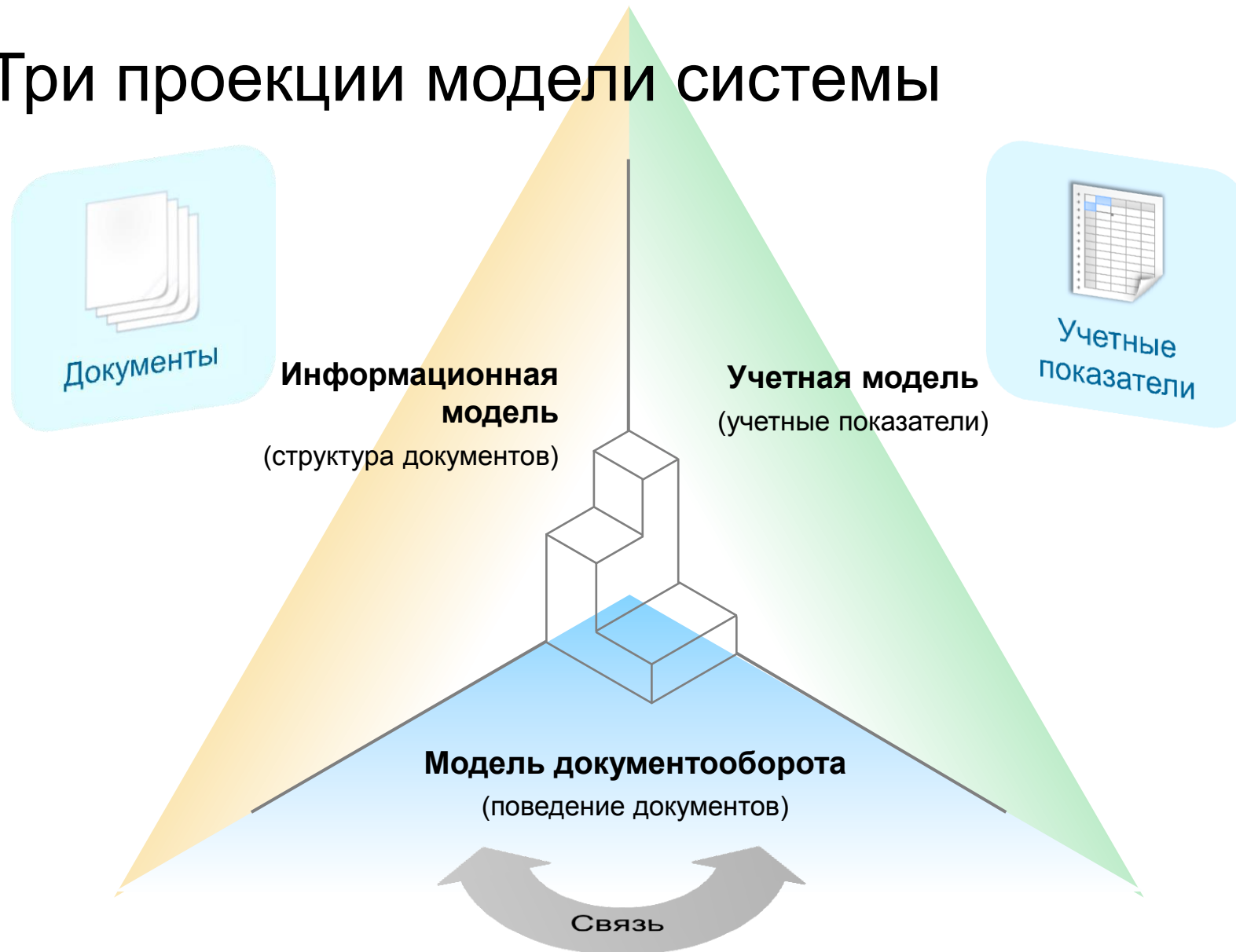
Типичное корпоративное приложение

- Пользователи создают документы  Объектная модель
- По необходимости заполняют справочники
- Потом документы исполняют  Жизненный цикл документа
- При этом меняются учетные данные
- Которые влияют на исполнение документов
- И отражаются в отчетах  Учет –
не объектная модель

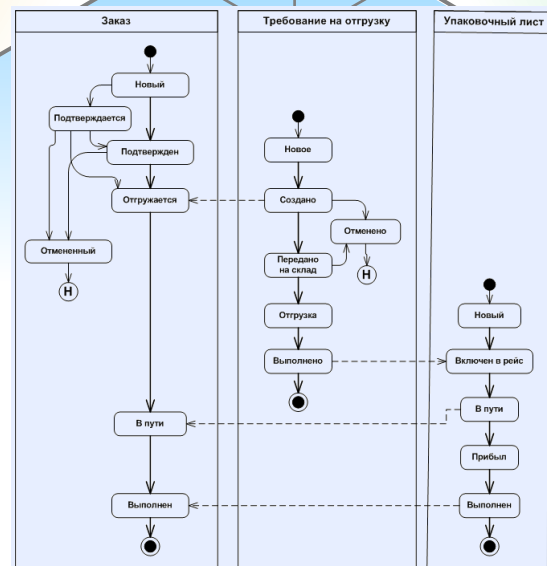
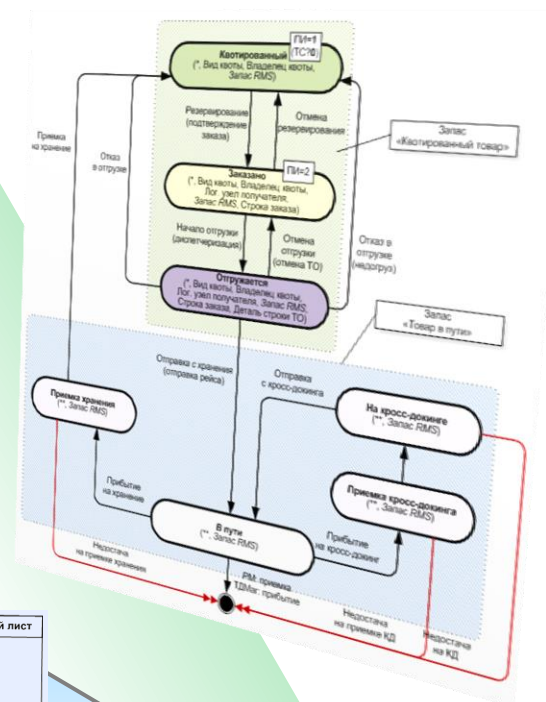
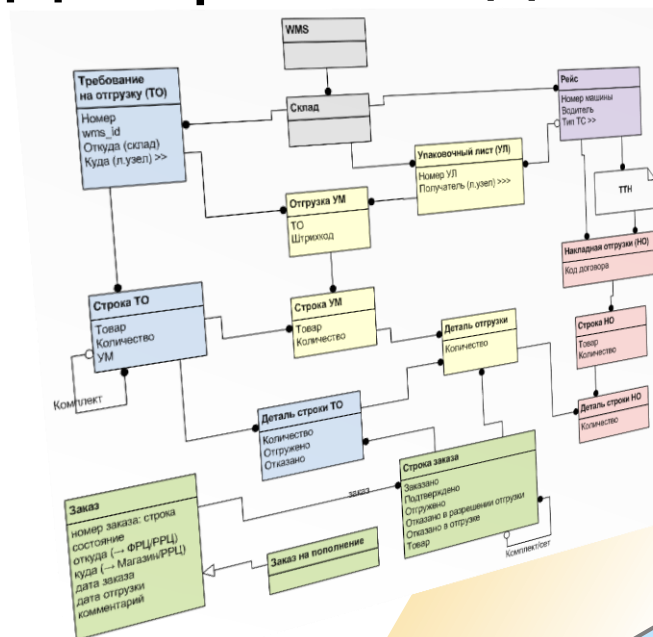
Одна модель или несколько?



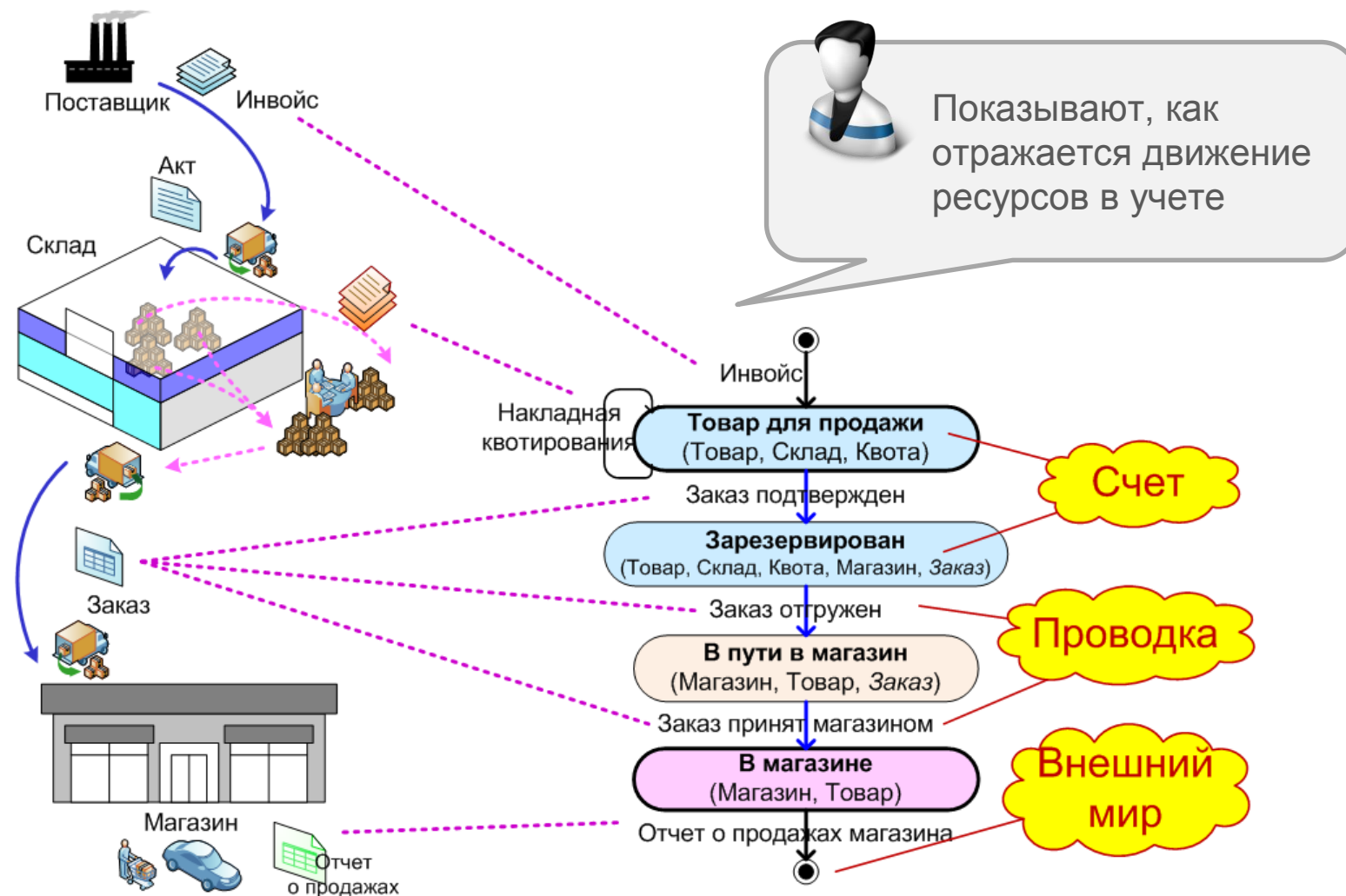
Три проекции модели системы



Диаграммы для проекций

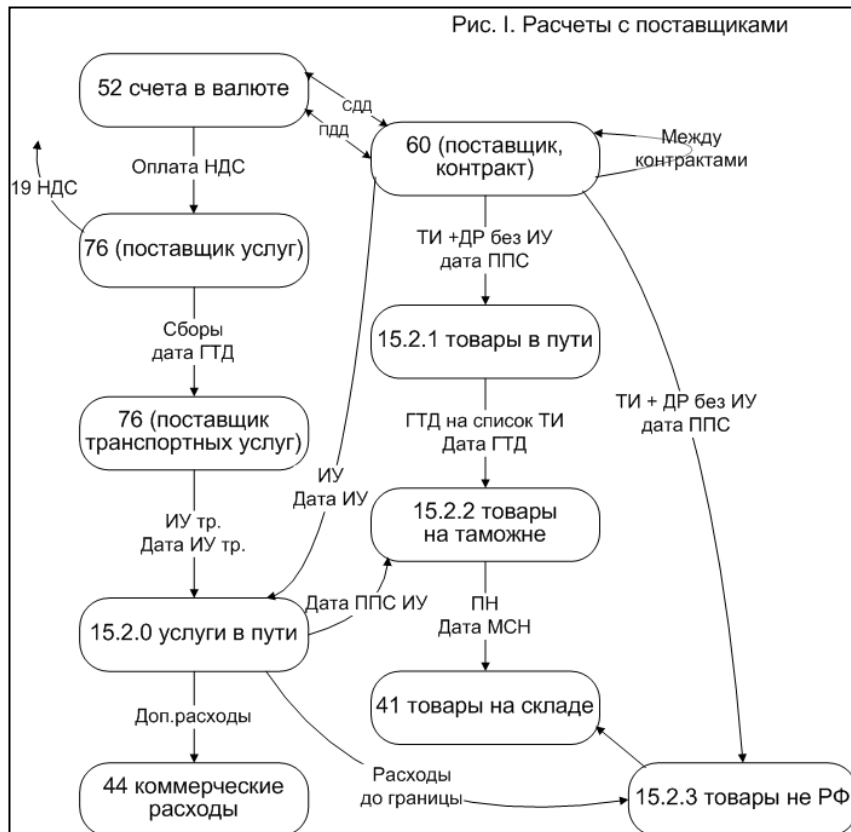


Диаграммы учета



Статья [«Диаграммы учета: мост между бухгалтером и разработчиком»](#)
(журнал «Бухгалтер и компьютер», №5-2011)

Они нагляднее,
чем Excel

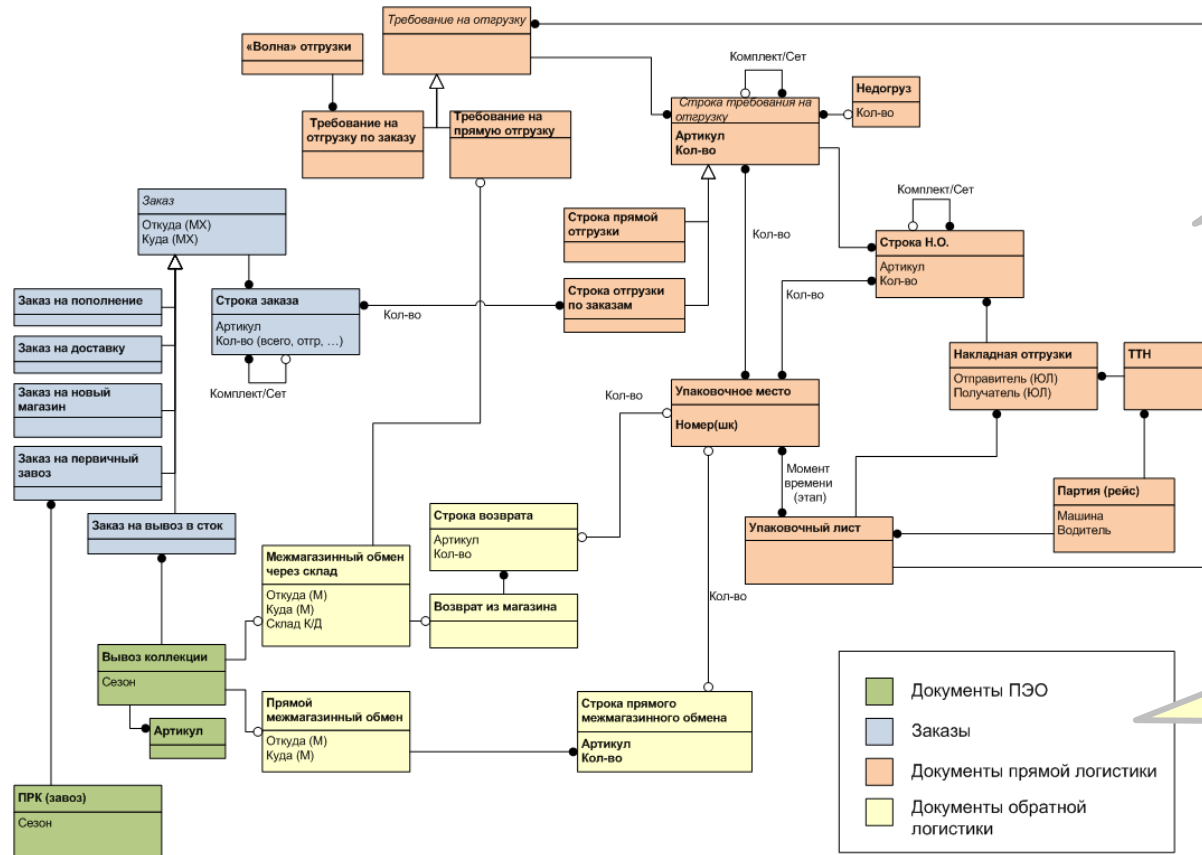
[illegible]

Документооборот – объектная модель

- Объекты с атрибутами и методами – элементы единого языка для предметной области и способ их соединения в сложные конструкции
- Для отражения документооборота используем **состояние документа**
- **Диаграмма классов и диаграмма состояний UML** – визуальный образ для наглядного представления
- Объекты в программе – способ отражения модели в реализацию

Модель должен понимать заказчик

Классы соответствуют бизнес-объектам –
заказчик видит знакомые названия



Представляем
диаграммой
классов

Используем
цветовые выделения

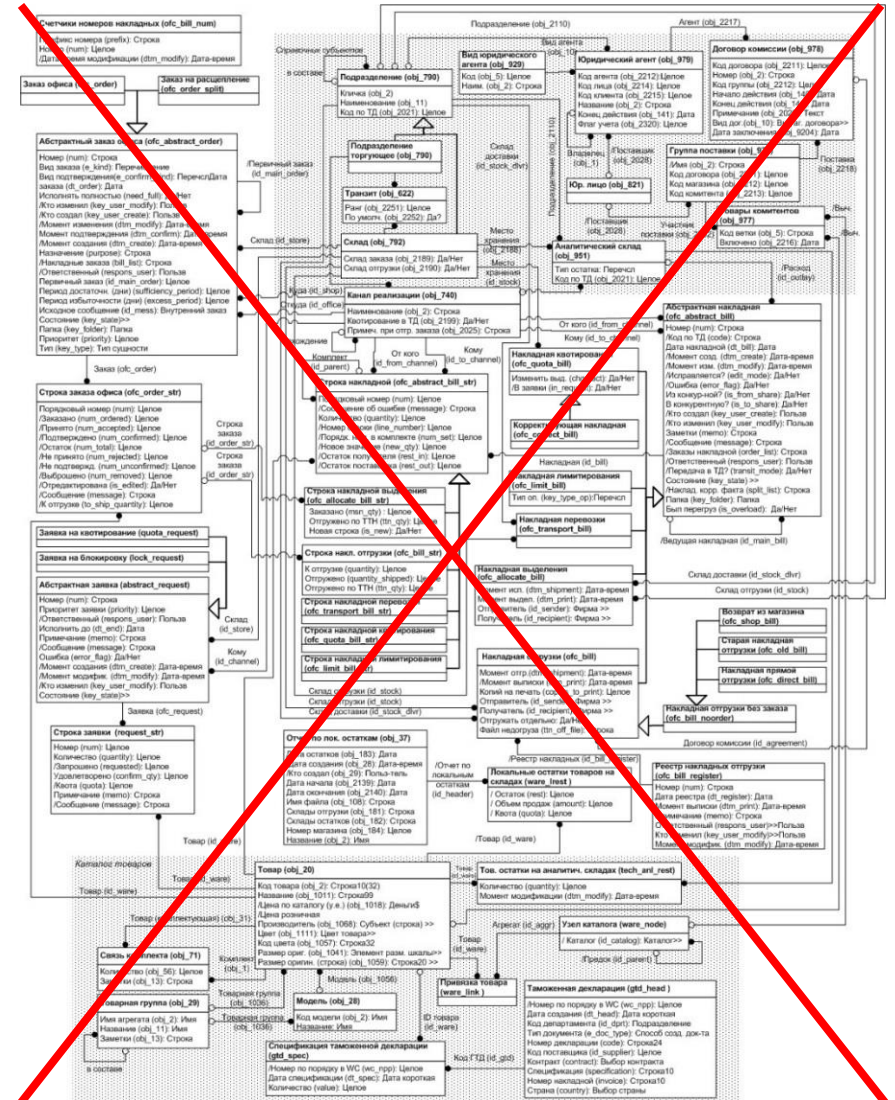


Не нужно усложнять диаграмму

 Не нужно

- Стараться изобразить все классы на одной диаграмме
- Отображать вспомогательные атрибуты
- Использовать технические коды
- Применять сложную нотацию

😊 Диаграммы должны понимать заказчики,
а не только разработчики

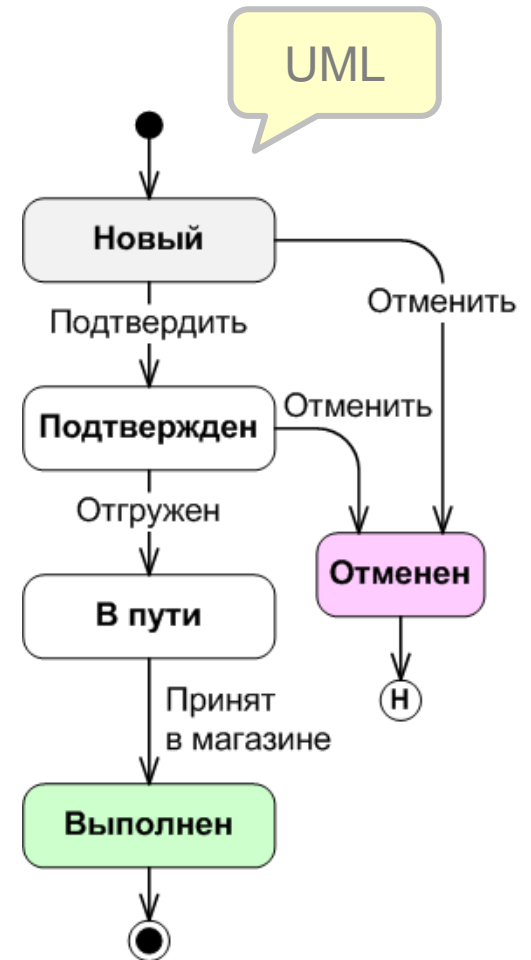


Модель документооборота

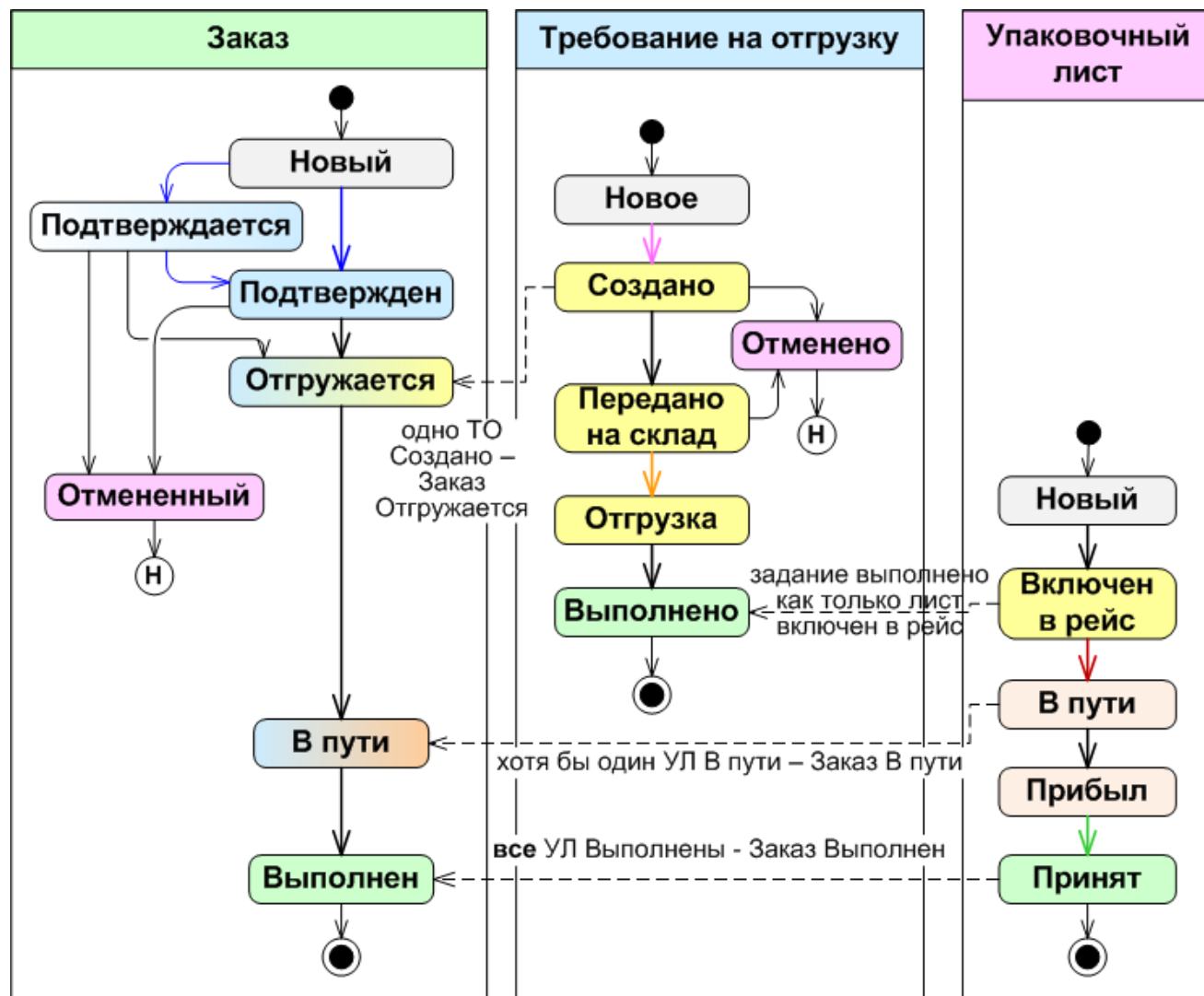
- Документу приписываем **состояние**, оно определяет этап жизненного цикла
 - Какие действия можно совершать и кому
 - Кто отвечает за обработку документа
- Возможные изменения состояний документа образуют **граф переходов** – State machine diagram



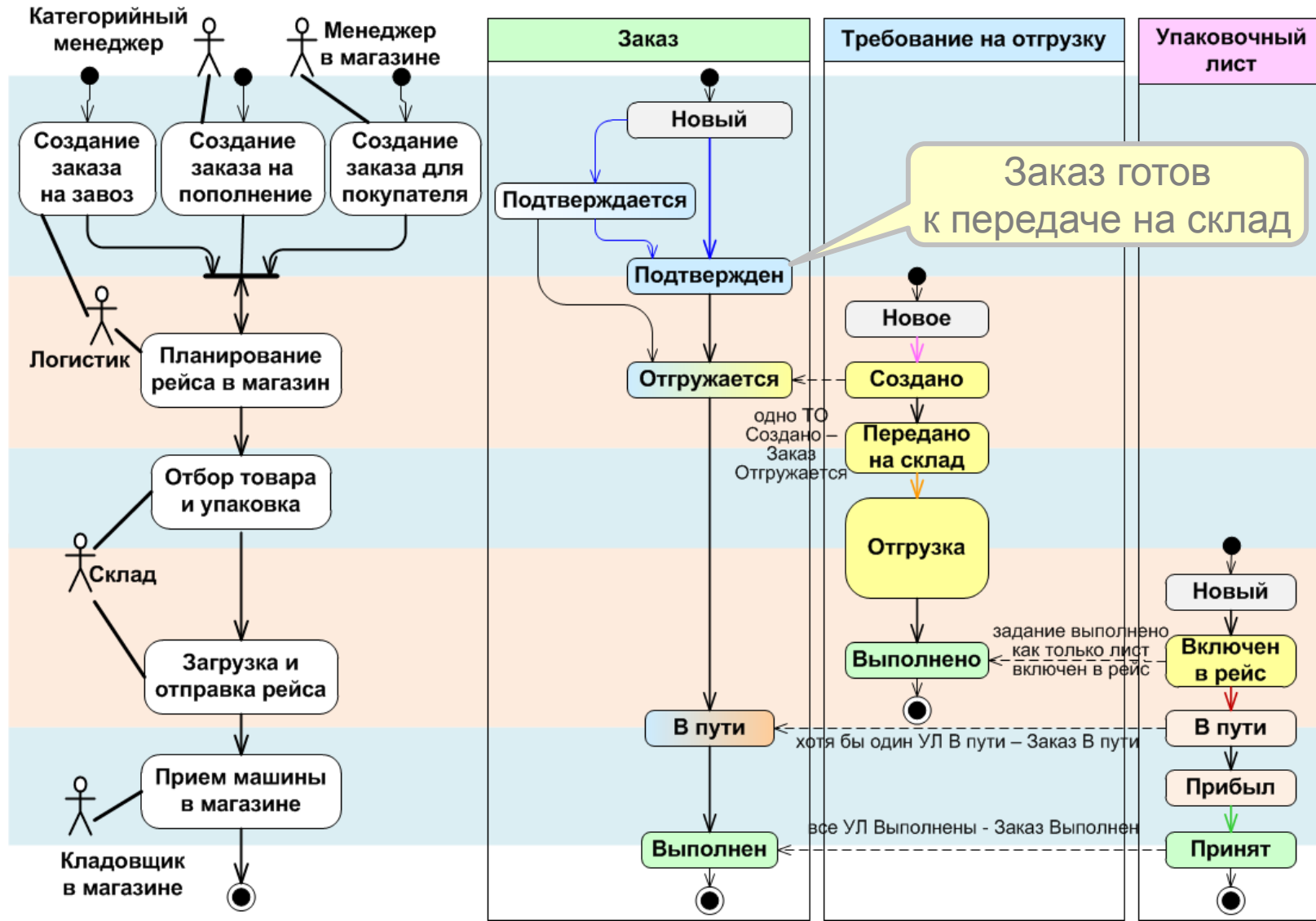
Язык ООП «с расширениями».
Названия состояний и переходов –
на языке бизнеса



Наглядно представляет сложный документооборот



Переходы документов и бизнес-функции



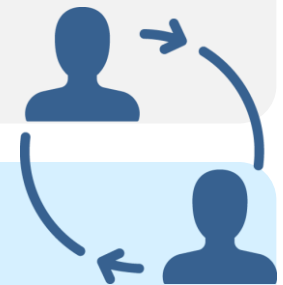
Переходы фиксируют выполнение бизнес-функции, а состояния соответствуют передаче по этапам обработки

И в заключение...

- Нет схем правильных и неправильных – есть **облегчающие коммуникацию** и бесполезные для нее
- Нотация должна пониматься всеми, а не только автором
- Визуальное описание эффективнее текста, но не заменяет его

Вопросы? Обращайтесь!

Максим Цепков
mtsepkov.org



Вакансии архитекторов и аналитиков

Пишите на hr@custis.ru, подходите с вопросами