

CUSTIS

Системное мышление — нужно ли оно в ИТ и зачем?



Максим Цепков

Главный архитектор решений CUSTIS

Навигатор по миру Agile, бирюзовых организаций и спиральной динамики

<http://mtsepkov.org>

27–28 ноября 2024, Москва



TeamLead++
Conf **2024**

Что такое системное мышление?

Как обычно, есть много *разных* мнений

- Это систематизированный подход к решению задач и проблем, обеспечивающий высокую вероятность их решения
- Это восприятие мира как совокупности взаимодействующих систем
- Это мышление, позволяющее создавать и изменять сложные системы

В определениях нет принципиальной разницы, отличие — в сложности решаемых задач, и в докладе будут все варианты



Задача ИТ — создание и эксплуатация сложной системы — софта, и системное мышление даёт **комплексный взгляд** на решение этой задачи

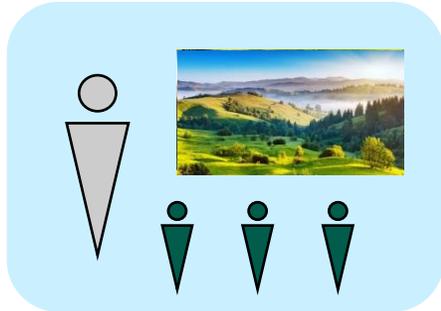
В чём проблема с системным мышлением?

- Методы проектирования и разработки софта, такие как ООП, опираются на системное мышление: авторы его знали и предполагали у разработчиков
- Нынешний тренд и ситуация с образованием — не копать в глубину, а действовать по аналогии, на основе примеров и простых моделей
- В результате в ходе проекта возникают провалы: таких моделей, как c4 model, Archimate или DDD оказывается недостаточно
- В докладе я дам представление о системном мышлении, покажу, где на него опираются конкретные методы и ошибки в случае его отсутствия
- Это не обучение, но доклад позволит замечать пробелы у себя и у других, разбираться в их причинах и работать с командой над их устранением

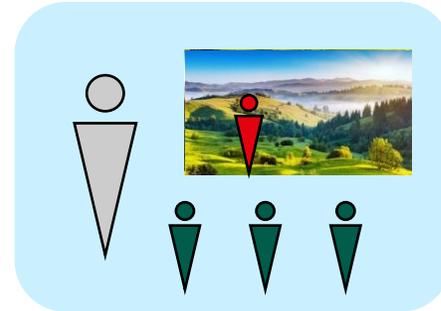


Лучшее, на мой взгляд, обучение методам рационального и системного мышления — в [школе системного менеджмента Анатолия Левенчука](#) (это не реклама)

Не просто слушать, а думать о применении



Рассказ даёт карту местности
и возможные варианты движения.
Местность активно меняется



Ваша задача — увидеть на этой
карте свой будущий путь —
применение системного мышления



Путешествие не является необходимым, это зависит от вашей ситуации. Но пока ты не представишь себя идущим, мой рассказ будет мёртвой теорией

Системный взгляд на ИТ-проект

Весь мир состоит из систем

- Система — выделенный набор взаимодействующих объектов, для которых внутренние связи сильнее, чем внешние
- Система может образовывать целостность, **большую, чем сумма составляющих**: есть синергетический эффект, эмерджентность, и в этом случае описания частей недостаточно для описания целого
- Первоначально систему понимали как объективное устройство объекта с физическими границами, как вскрытие чёрного ящика
- Применение в биологии и социальных науках привело к понятию **soft system с нечёткими границами**: экосистема, система безопасности — от устройства мира перешли к способу мышления о мире
- Выделение системы зависит от цели: для одних целей система «человек» ограничена его телом, а для других включает личные вещи, обеспечивающие функционирование, например, смартфон

Нечёткие границы систем

Пример: система доставки интернет-магазина

- Понятный процесс доставки: от склада до получателя
- Внешние курьерские компании: это отдельные системы или материал для элементов нашей системы доставки, а SLA задаёт требуемую морфологию материала?
- А собственная служба доставки, она подобна внешним, она точно внутри?
- Формирование условий доставки при заказе клиента — в системе доставки или магазине?
- Если интернет-магазин в составе большой розничной сети, то у него отдельная служба доставки (система)? Или удобнее рассматривать всю службу логистики как одну систему?
- Может ли автономная собственная служба начинать выполнять чужие заказы?
- Что изменится, если наш интернет-магазин начинает продавать чужие заказы?

Все эти вопросы не только про бизнес, но и про гибкость ИТ-решений

Система и её окружение

Целевая система — та, над которой мы работаем в настоящий момент

- Различаем описание системы и её саму в реальном мире

Целевая система встроена в **надсистему**

- Выполняет определённые функции в ней
- Отвечает на интересы агентов надсистемы

Эксплуатирующая (обеспечивающая) система обеспечивает работу целевой, обычно включает **команду** и другие части, живые и неживые

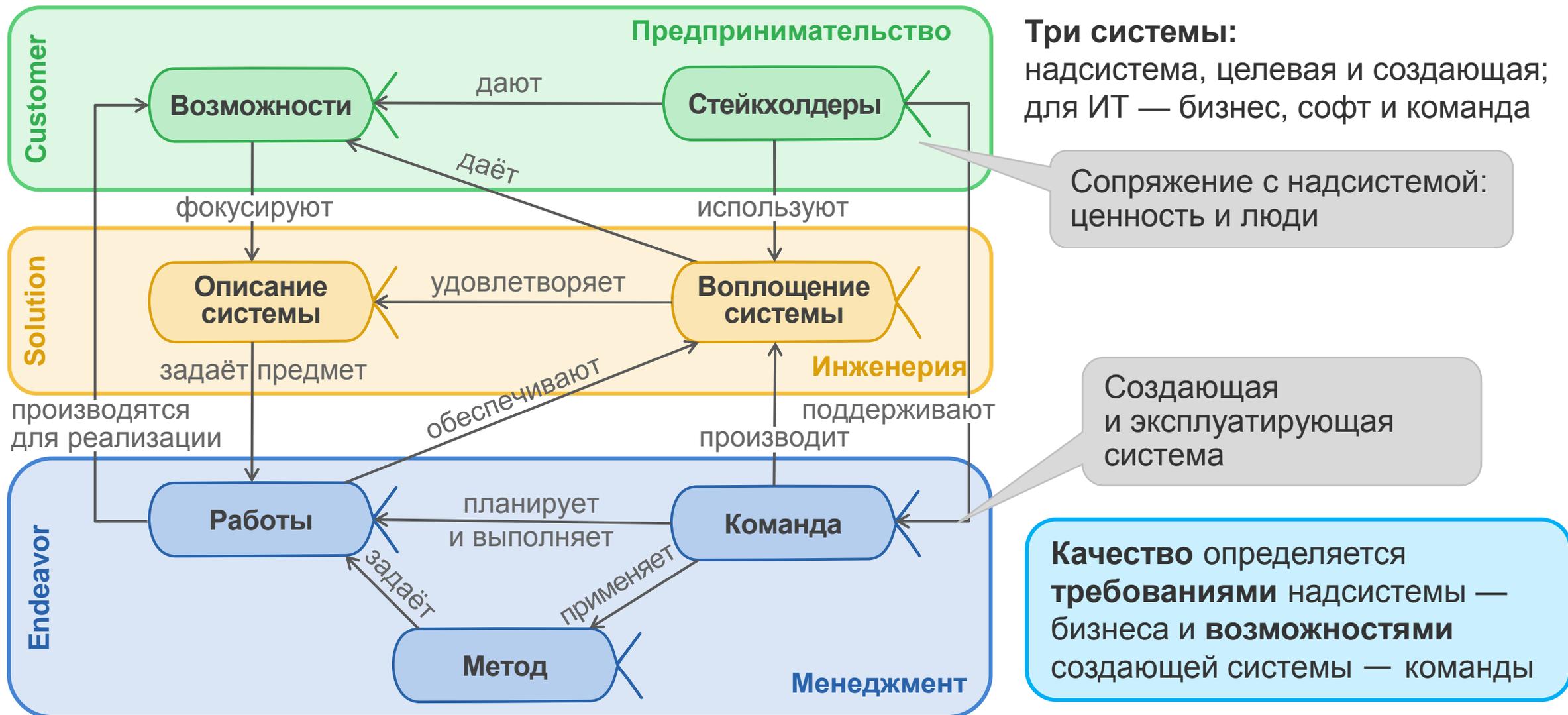
Создающая система — та, что создаёт и развивает целевую

Окружение системы — смежные системы, с которыми целевая взаимодействует в процессе своей работы



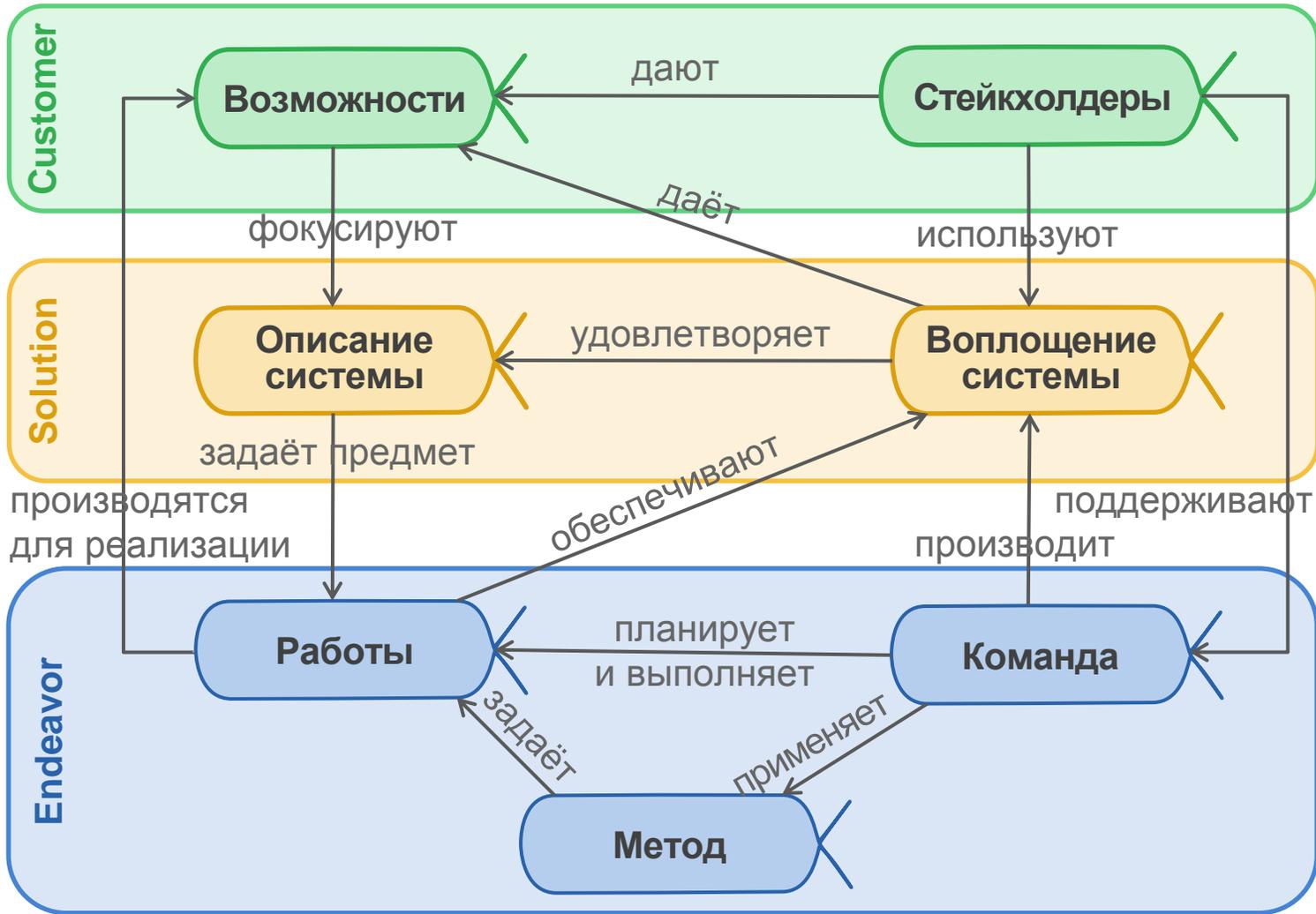
Нетривиальные связи — [закон Конвея](#): отражение в структуре создаваемой ИТ-системы оргструктуры команд создающей системы

OMG Essence — обобщённое описание проекта

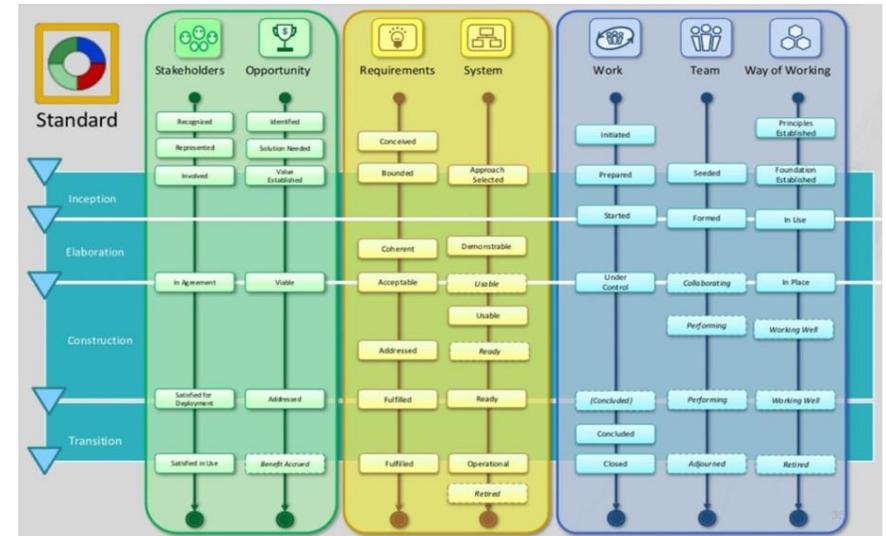


Стандарт создан для ИТ и принят OMG, адаптирован Анатолием Левенчуком для системной инженерии и общего менеджмента

Нелинейный workflow проекта



Концепция: каждый объект внимания — альфа — живёт своей жизнью



Есть описания всех методов разработки и сопровождения в ИТ в этом формализме: в [стандарте](#) и [библиотеке Ивара Якобсона](#)

Системный взгляд на команду и человека



Моя книга «Инженерная модель личности»

- Для эффективного сотрудничества с людьми надо **понимать** себя и других, уметь изменять себя и способствовать изменению других
- Этому помогают модели психологии — они показывают способы взаимодействия, возможность и способы изменений для типов людей
- Моделей много, они разрознены и противоречат друг другу, их основания непонятны, и потому их часто отвергают вместо уточнения
- В книге — сборка моделей нейрофизиологии и психологии в архитектурную модель личности



Понимание механизмов мозга помогает работать с собой и другими, как знание устройства базы данных помогает делать эффективное хранение

Методы ведения ИТ-проектов и убеждения стейкхолдеров

Агенты и роли

Деятельность ведут **агенты**: люди, команды, компании, сообщества

Театральная метафора: **агент играет роль**, действуя в системе

- Человек может совмещать роли, например, руководитель и разработчик
- Компания тоже играет роли в надсистемах: на рынке, в государстве, обществе

Роль предполагает конкретное взаимодействие с другими ролями, которое определяется конструкцией системы

Агент имеет **цель**, для которой он встроился в работу системы, и она некоторым образом соотносится с **целью системы как агента**

У агента есть **предпочтения** по поводу исполнения своей роли и **интересы**, касающиеся соучастия в результатах работы системы

Влияние убеждений стейкхолдеров

У стейкхолдеров могут быть принципиальные различия убеждений, влияющие на проект

- Одни выступают за тщательное планирование как залог успешной работы, и нестрашно, если процесс планирования будет долгим
- Другие говорят, что есть много факторов, которые невозможно предусмотреть, и надо быть гибким, быстро реагировать на изменения

Это влияет на организацию тестирования, проведение демонстраций и т. д.

Убеждения связаны с представлениями о культуре ведения проектов, которые стейкхолдеры когда-то приняли и которые полагают правильными



Надо видеть за словами в коммуникации по частным вопросам **убеждения стейкхолдеров** и работать с ними

История культур ИТ-проектов

Рамка проекта

ИТ-система...

обеспечивает
бизнес

делает то,
что нужно

сделана
вовремя

работает

Public web и продуктовый подход:
софт становится основой для бизнеса



Время персоналок и Scrum:
софт для бизнеса, задача
меняется, пока идёт разработка



Эпоха RUP — учимся создавать софт
по проекту в заданные бюджеты и сроки,
как в производстве других отраслей

Провал

Эпоха НИОКР — создаём софт для решения
заданной задачи так, как создают другие
инженерные изделия, например, автотехнику

Большие
компьютеры

1960

1985

2003

2013

О культурах ИТ-проектов у меня есть статья [«Краткая история ИТ-менеджмента»](#), доклад [«Мыслить проектно: история и современность»](#) и ряд других выступлений

Бизнес и софт

Продукт как возможность

Пример. Бизнес требует поддерживать услугу срочной доставки, так как без неё есть риск проиграть конкурентам, у которых она уже появляется

- Вопросы о возможности: кто будет пользоваться, для каких заказов, что значит «срочная», как мы будем проверять гипотезы и какова мощность в динамике?
 - Справится пеший курьер или нужна машина?
- Будет ли отдельная система срочной доставки? Или надо существующую систему доставки дополнить элементами и наделить способностью так доставлять?
- Какова целевая операционная стоимость доставки, какова стоимость в пилоте, каковы допустимые инвестиции, в том числе для создания ИТ-систем?
 - Если курьер будет брать самокат, стоимость останется допустимой?
- Какие возможны лёгкие решения на пилоте, справится ли Excel + мессенджер?
- Как будем реагировать, когда не справляемся, кого из клиентов обижать?
- И так далее...

V-модель — линейный взгляд на проект



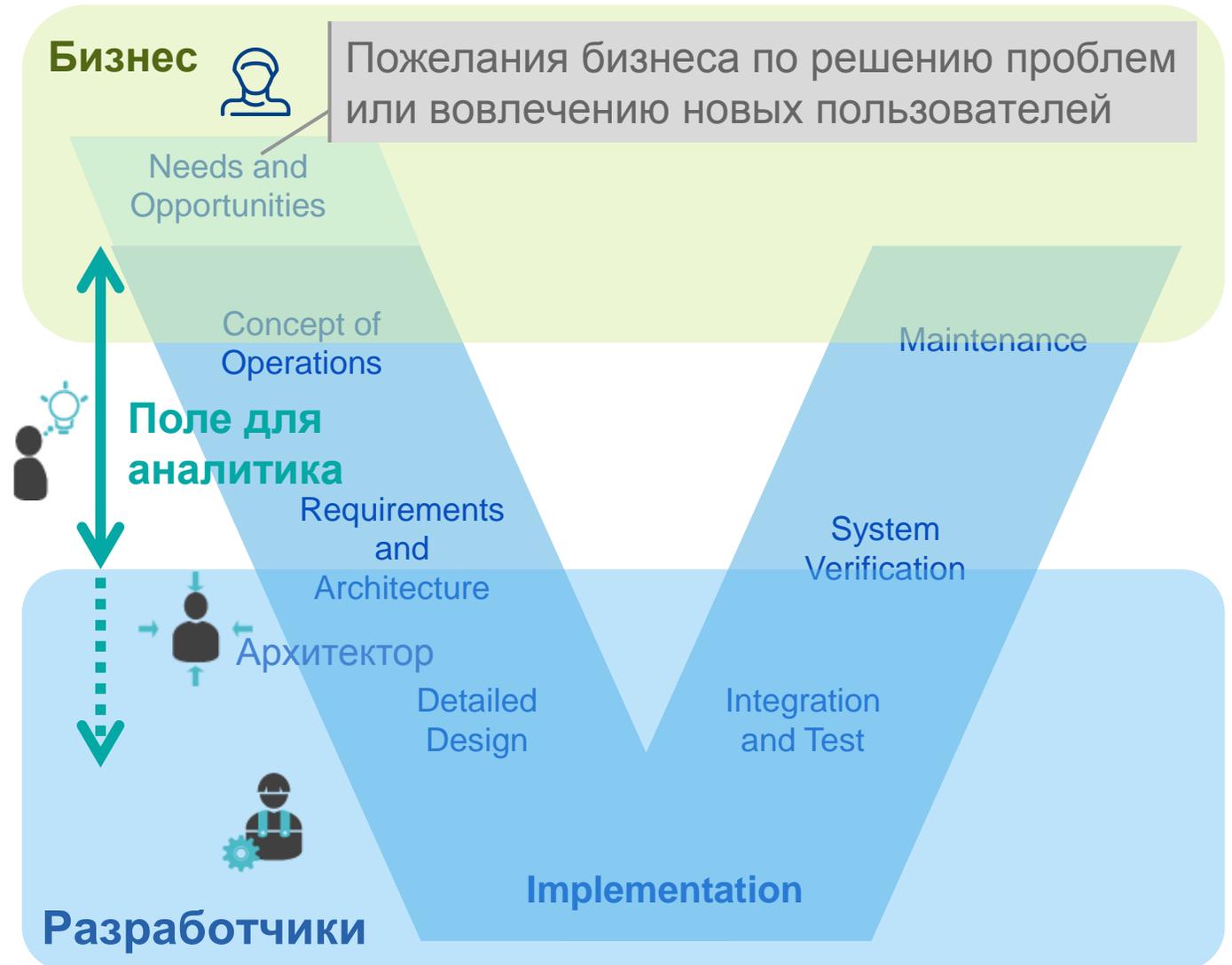
[V-Model \(Wikipedia\)](#)

Разрыв между бизнесом и разработкой

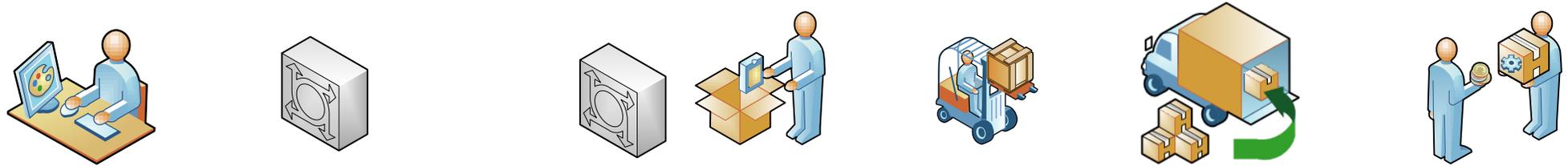
- Разрыв может закрывать разработчик или аналитик
 - Как устроен бизнес?
 - Какое место займёт софт?
 - Какой для этого нужен софт?
 - Как с ним будут работать?
- Надо не только найти ответы, но и согласовать их с бизнесом и разработкой



Мы описываем встройку подсистемы (софта) в надсистему (бизнес), сопрягая различный материал и модели систем



Три уровня представления: обработка интернет-заказа



Деятельность

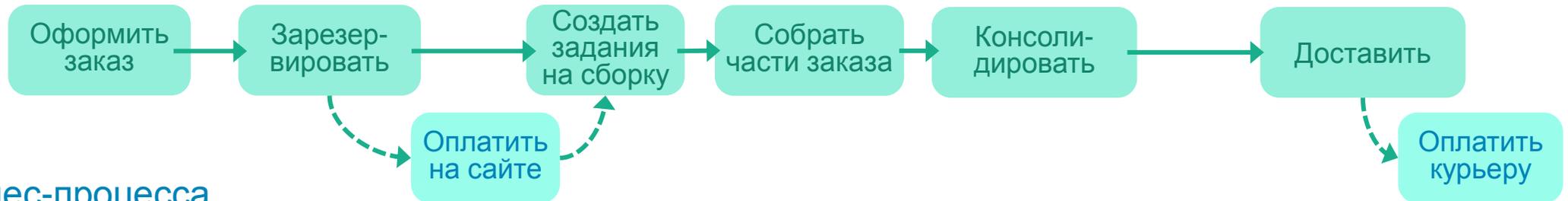


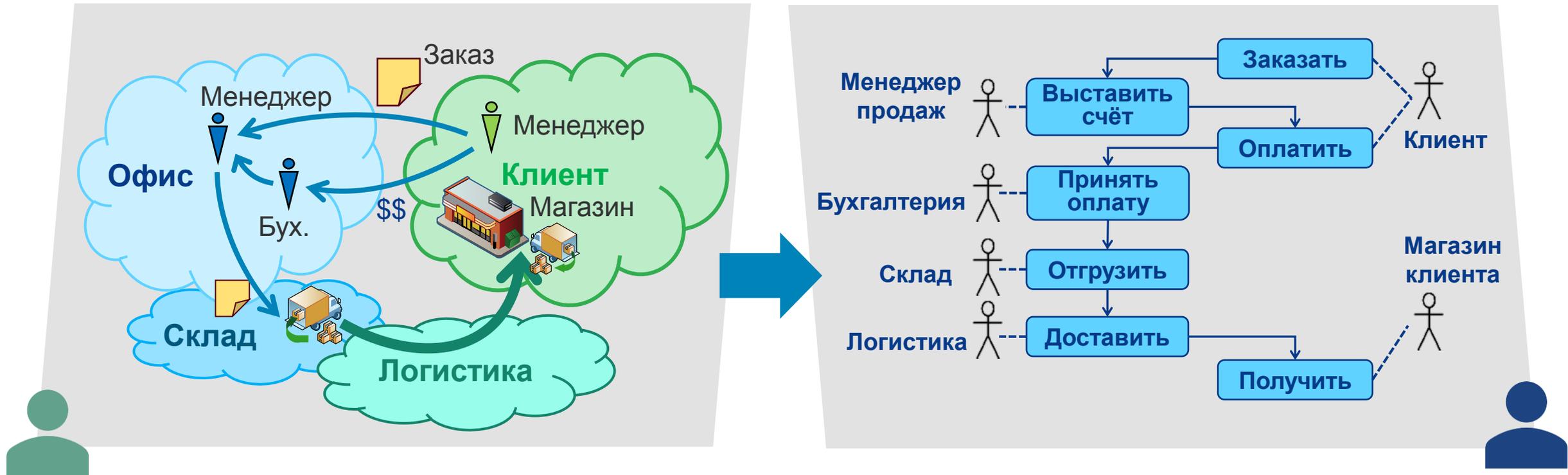
Схема бизнес-процесса



Объекты и их состояния

Формализация бизнес-модели

Оптовые продажи магазинам и торговым сетям



Как осуществляется переход к формальной бизнес-модели от представлений о повседневной деятельности на интервью?

От нарратива к объектам

- В процессе понимания текст размечается: в нём выделяются объекты, которые служат основой моделей
 - **Физический объект** — нечто, выделенное в мире, имеющее идентичность, границы в пространстве и существующее во времени
 - **Ментальный объект** — абстракция, идеальный объект в мышлении, полученный обобщением некоторых объектов, физических или ментальных, или их множеств
- **Два типа моделей**: мозаичная с экземплярами и концептуальная с обобщениями
- Разные типы отношений между объектами
 - Обобщения и абстракции: экземпляр — класс, тип или аспект
 - Иерархия абстракций: класс — подкласс или тип — подтип
 - Часть — целое
 - Агент — роль, интересы и поведение
- Конкретные отношения часто опускают в текстах и путают между собой

Выделение объектов и построение модели

- Основной принцип: существительное — это объект
 - Может не быть названия: «кредит погашается платежами или другими способами»
 - Одно название может скрывать несколько объектов: счёт поставщика и счёт покупателю различаются, и это могут различать в терминах (invoice и счёт), или называть одинаково
 - За одним словом может быть разное содержание: «товарная группа» для закупщиков — товары, закупаемые вместе, у мерчендайзеров — выставляемые рядом для покупателя
- Типизация объекта может быть разнообразна, возможны [типологии Борхеса](#)
 - Виды субъектов по правовым формам и ролям: ЮЛ, ФЛ, ИП, самозанятые, налоговые...
 - Многообразные виды платёжных поручений, к ним добавляются неденежные расчёты...
 - Субъекты с уникальной ролью: «Иван Иванович может разрешить превышение лимита»
- В тексте может употребляться чересчур общий или слишком конкретный тип
- Исключения во взаимоотношениях объектов — норма, как и в русском языке

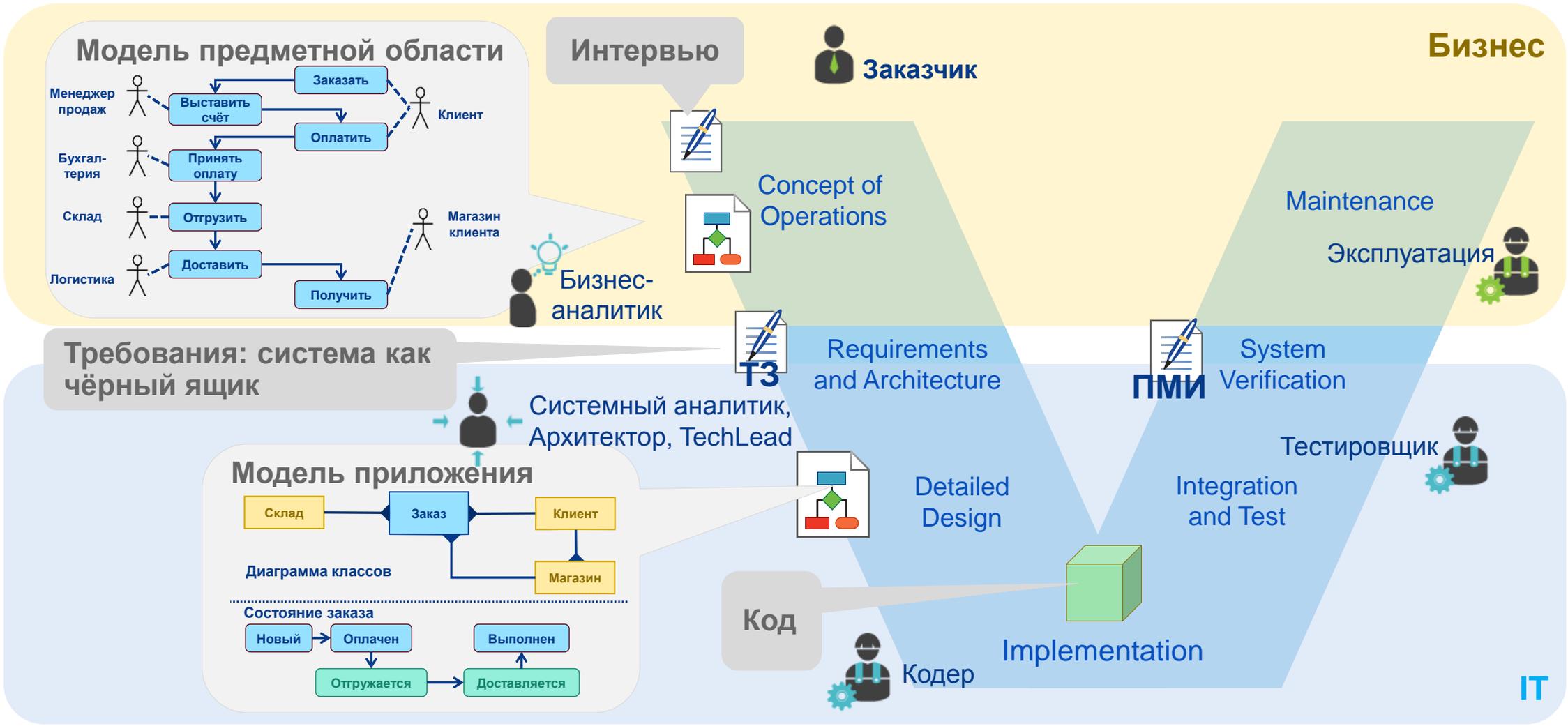


Такие же нечёткие типы проявляются в описании процессов команды и типов людей. Виды программистов в книге Рейнвотера «Как пасти котов» — пример типологии Борхеса.

Схемы для прояснения мысли

- ✓ Тексты для понимания требуют разметки, а схемы уже размечены
- ✗ Схемы — не гарантия разметки, там тоже бывают свои нарративы
- ✓ Нарботан ряд эффективных представлений для разных целей: диаграммы классов и ER-диаграммы, схемы состояний и бизнес-процессов и др.
- ✗ Проблема с представлением больших объёмов на схемах не решена
- ✓ Табличные представления более строгие, позволяют представить больший объём, но воспринимаются значительно хуже
- ✓ Полное представление — композит: схемы, таблицы, текст

Две системы



Опыт показал, что такое разделение приводит к несоответствию ИТ-системы ожиданиям бизнеса. Agile-методы это поменяли, но подход к описаниям остался

Event Storming и бизнес-процессы

BPM — моделирование бизнес-процессов

- Представляем деятельность как последовательность шагов с ветвлениями
- Хорошо подходит для основного потока операций, плохо — для исключений
- Есть работы, которые «состоят из исключений»: выверка отчётов, увязка планов, для них подходит Case Management, а не Process Management

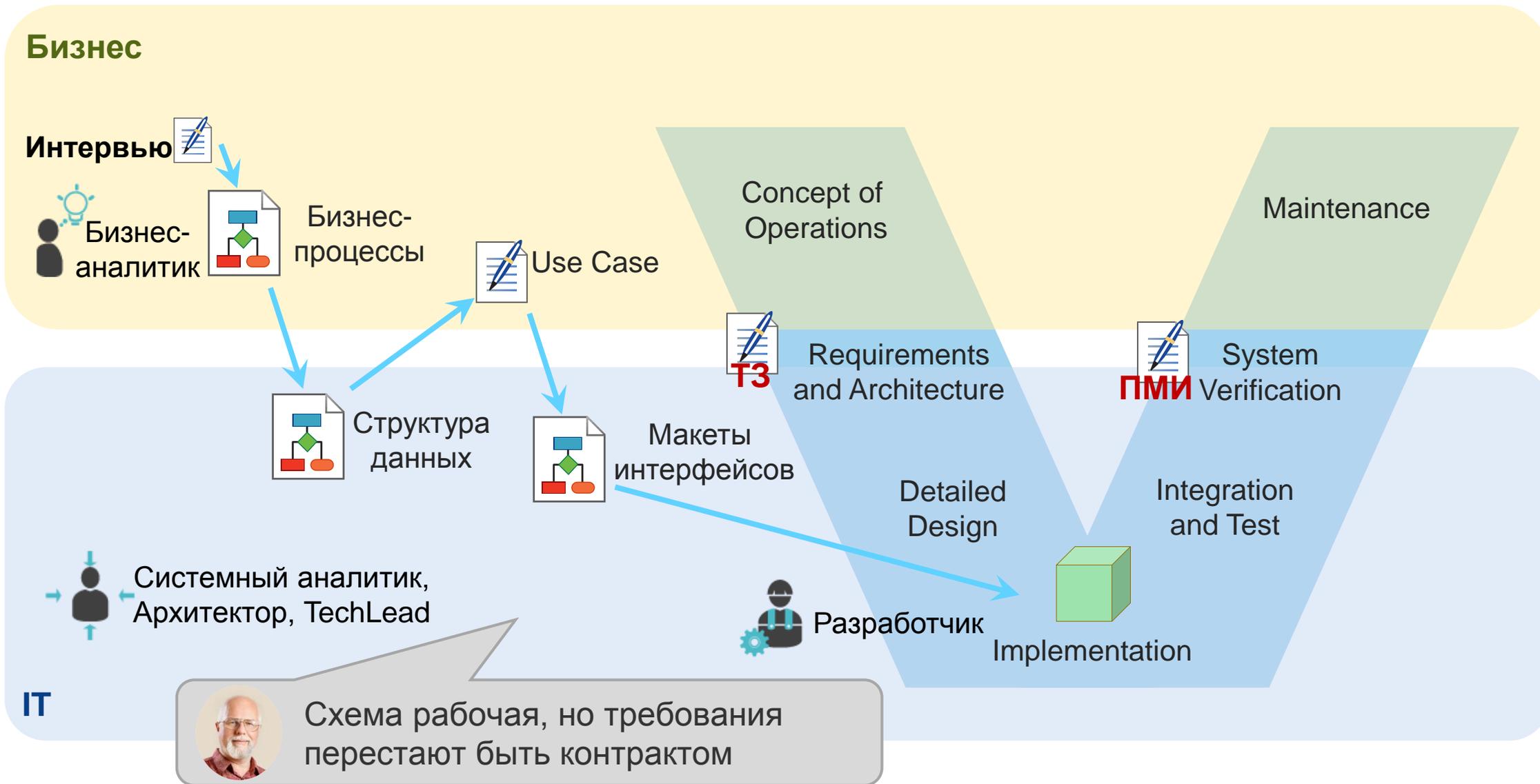
Event Storming

- Выдаёт событийную, а не процессную модель устройства деятельности
- Хорошо подходит для автоматизации «от событий» в сервисной архитектуре
- Помогает разобраться в бизнесе, но не даёт целостного представления
- Восстановить целостное представление, связав разные события, можно по-разному: через схему бизнес-процессов, через систему целей или через учёт



Различные парадигмы представления бизнеса отражаются и на методах, применяемых на последующих стадиях проекта

Нужны два прохода: данные и интерфейсы



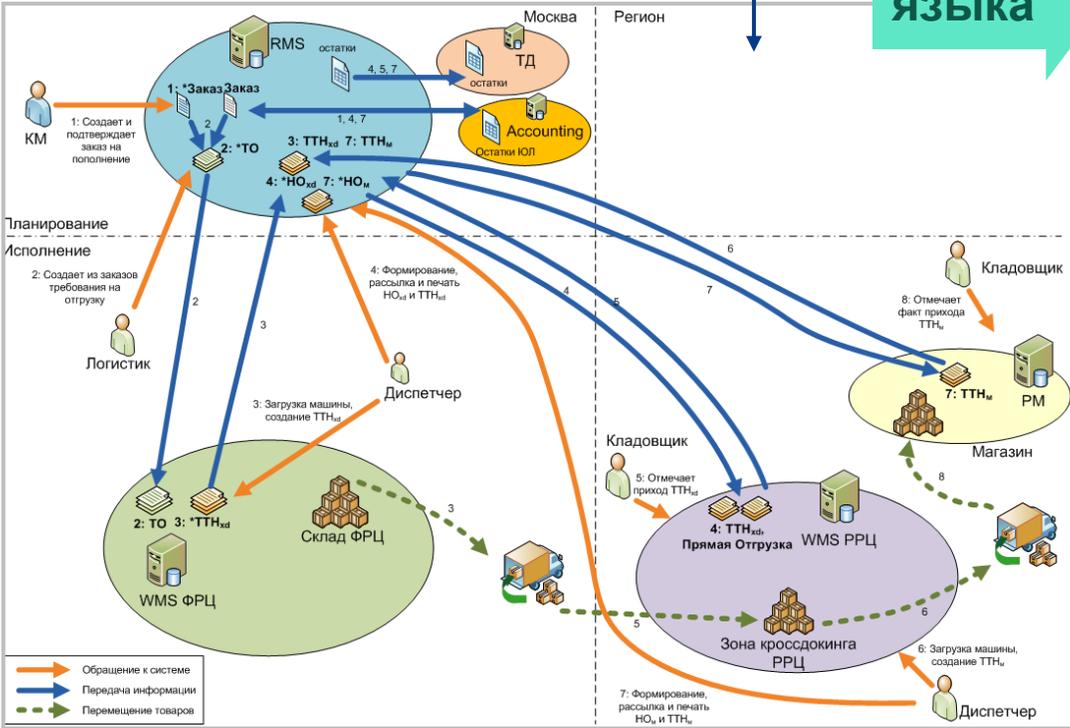
Формализация снабжения магазинов

Состояние документов — State Diagram

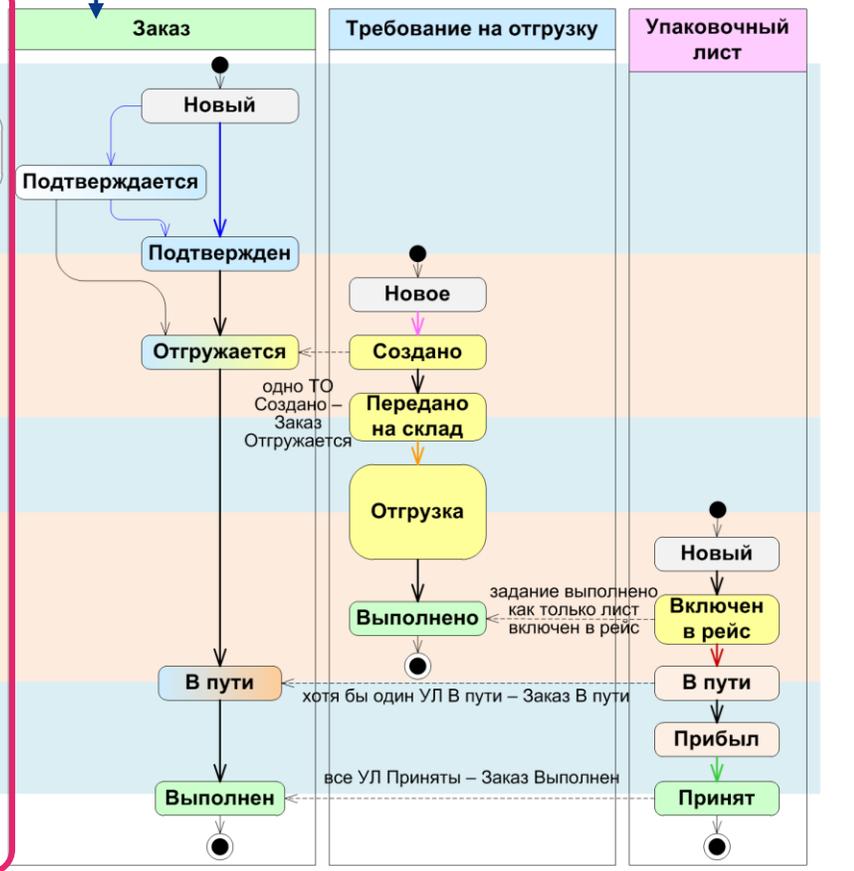
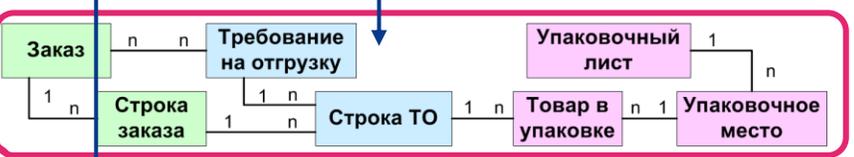
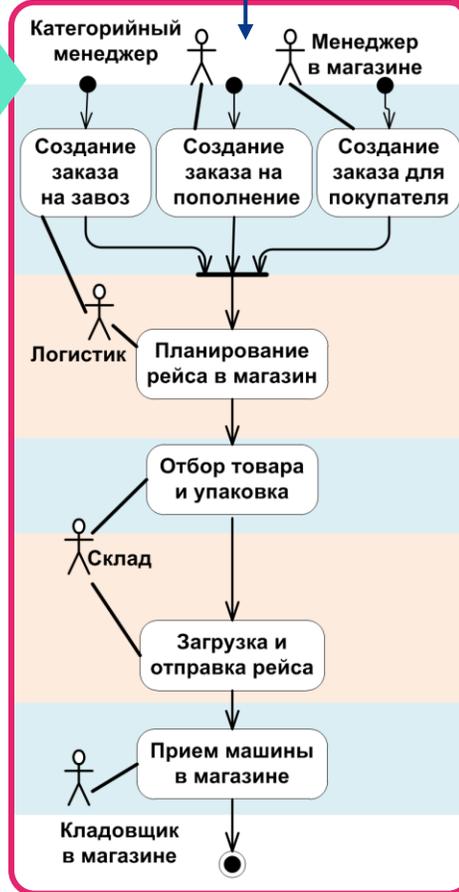
Объекты — Class Diagram

Неформальная схема деятельности и её отражение в существующих системах

Смена языка



Бизнес-процесс — Activity Diagram



Многоплановый взгляд на системы

Система представляется через множество планов, viewpoint, которые имеют своего адресата и назначение

- **Два взгляда на систему:** снаружи из надсистемы и устройство внутри
- **Три схемы устройства системы:** функциональная, модульная и размещения, исполнение функции может быть локализовано в конкретных модулях или распределено в виде аспекта
- **Выделение системных уровней:** архитектура бизнеса, приложений и техническая в архитектуре приложений, или UI, бизнес-логика и хранение, или фронт и бэк

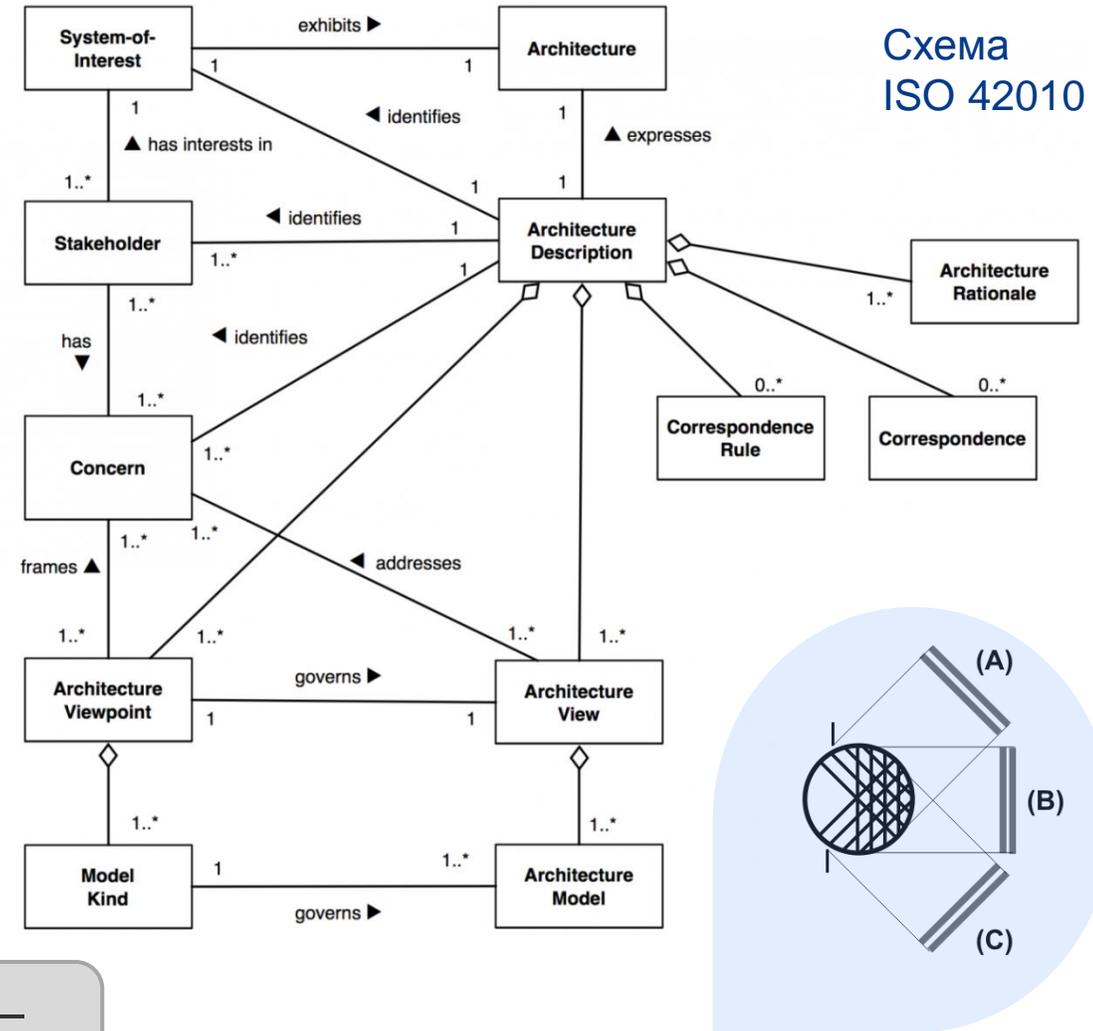
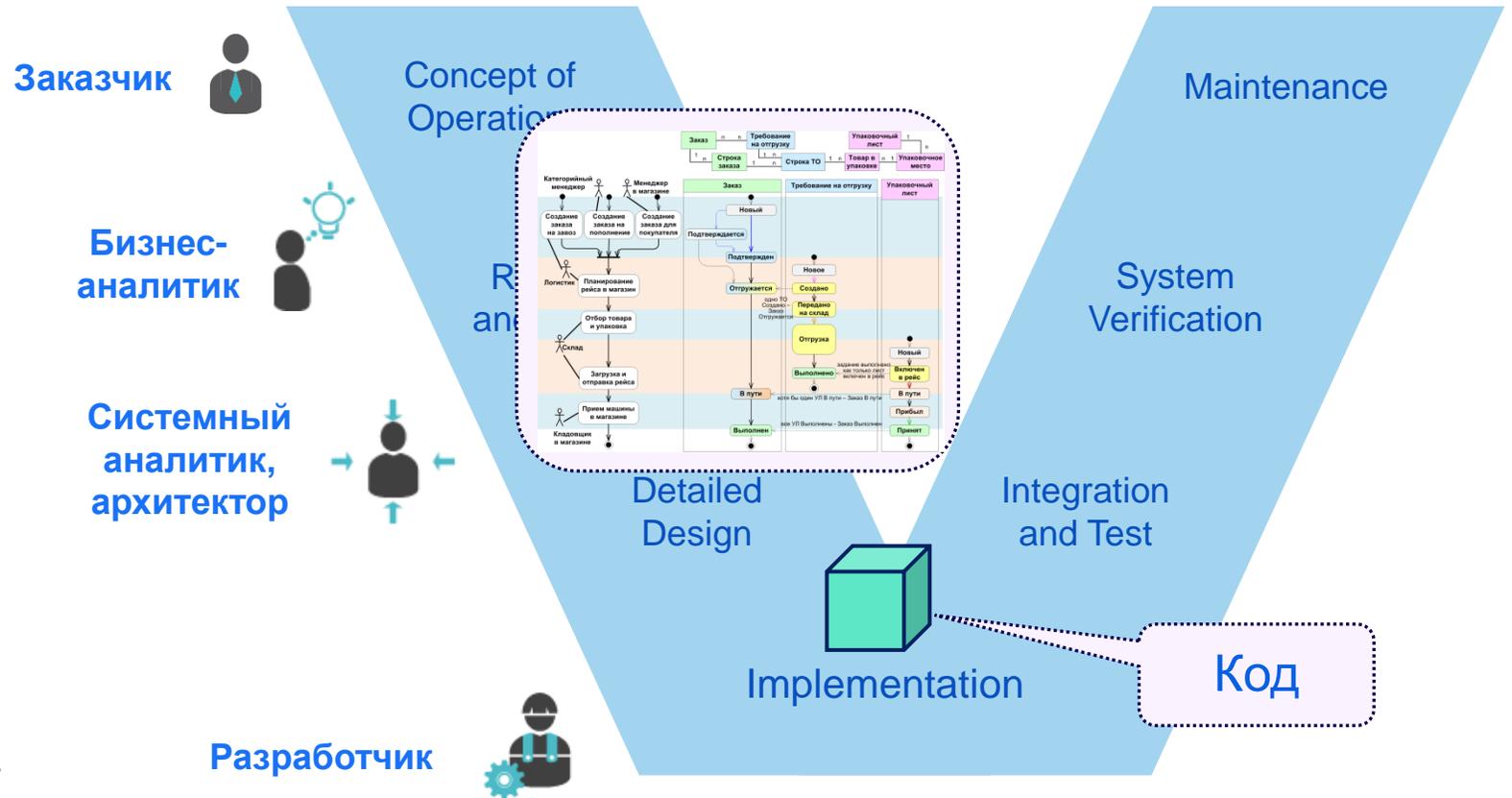


Схема ISO 42010

 ISO 42010 представление через viewpoint — раскрытие схемы многих знаний

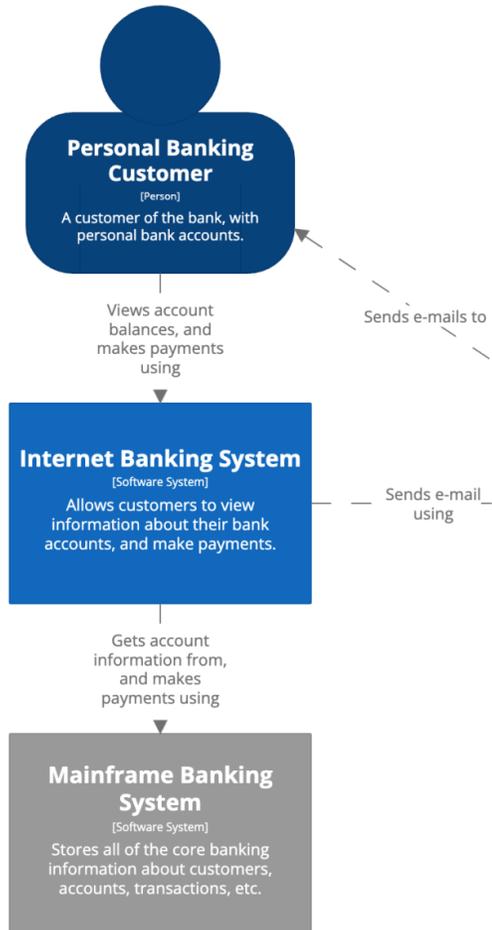
Domain Driven Design — единая система

- Единый язык
 - На основе терминов предметной области
 - Понятен всем участникам проекта
- Единая модель, приложения и его встройки в бизнес
- Прозрачное отражение модели в код



У меня есть много докладов о DDD, последний «[DDD: модели вместо требований 9 лет спустя \(ЛАФ-2023\)](#)»

Контекстная диаграмма C4 Model — система в окружении

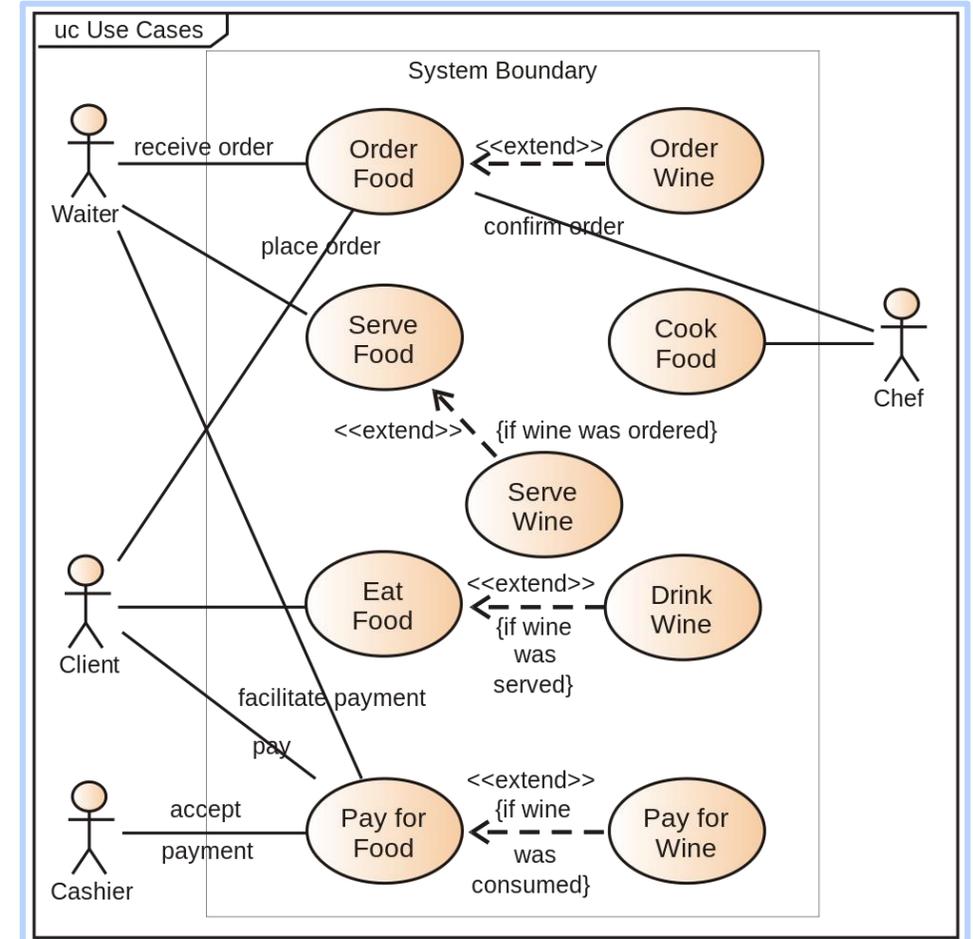


В центре — наша система, вокруг — пользователи и взаимодействующие системы

Стрелки — акты взаимодействия

Множество внешних систем — смена парадигмы на сервисы

По сути, это свёрнутая use case-диаграмма: вместо овалов — стрелки с подписями



Проектирование софта — место системного мышления

В каких точках нужен системный взгляд?

- Обработка особых ситуаций
 - У покупателя дорогого товара не хватает денег на одной карте или наличных
 - Покупателю надо доставить заказ в дом, до которого Яндекс-карты не знают дороги
 - При доставке привезли другой товар, покупатель согласен на замену — надо оформить
 - В точке получения не работает мобильный интернет, а приложению он необходим
- Сложности выделения единицы конструкции
 - Бизнес-объекты велики: заказ надо уметь собрать, оплатить, отгрузить, доставить, напечатать документы с реквизитами покупателя, в результате тестирование мелких правок печатной формы накладной требует настройки лимитов и проведения оплаты
 - Выделение абстракций, например, формы печати накладной, связывает типы документов, используемых для разных целей; исправление одного ломает другое
- Мир изменяется, нужно воспроизводить старое, например, напечатать договор или отчёт с реквизитами и в форме, как было вчера или год назад



Системное мышление помогает удерживать разрыв между бизнесом и его отражением в софте и целостность архитектуры софта при реализации частных решений

Гибкость структуры объектов и интеграции

- В любом объекте полезен произвольный комментарий пользователя
- К любому объекту может потребоваться добавить атрибут, и не один, и хорошо, когда такое дополнение требует доработки только там, где новый атрибут реально используется в бизнес-логике или UI
- Списки являются расширяемыми и превращаются в справочники
- Лучше, когда новые атрибуты подхватывает интерфейс — *naked objects*
- Интеграция передаёт любые сообщения и объекты, при этом в интерфейсах просмотра сообщений можно развернуть их внутреннюю структуру



Системное мышление нужно, чтобы заложить в архитектуру гибкость моделей, возможность будущей адаптации и развития софта за развитием бизнеса

Резюме из доклада «[Проектирование для многообразия — конструктор и DSL вместо жёсткой реализации требований](#)» на WIAD-2019

Выделение ячейки проектирования

- Методики проектирования склонны выделять элементарную ячейку с единственной ответственностью: объект или (микро) сервис
- Реальный мир
 - Мы мелко дробим объекты и получаем накладные расходы на взаимодействие
 - Или, наоборот, собираем всё вместе — и появляются BigObject или микролит сервисов
- Системное мышление учит работать со сложными системами
 - Различать функциональную и модульную структуру
 - Различать логические уровни систем, границы которых могут отличаться от физических
 - Сопрягать логические структуры с особенностями физического материала
- Ясной логической структуры можно достичь на любом материале — в любом фреймворке или языке реализации

Микросервисы: зачем нужны и что меняют

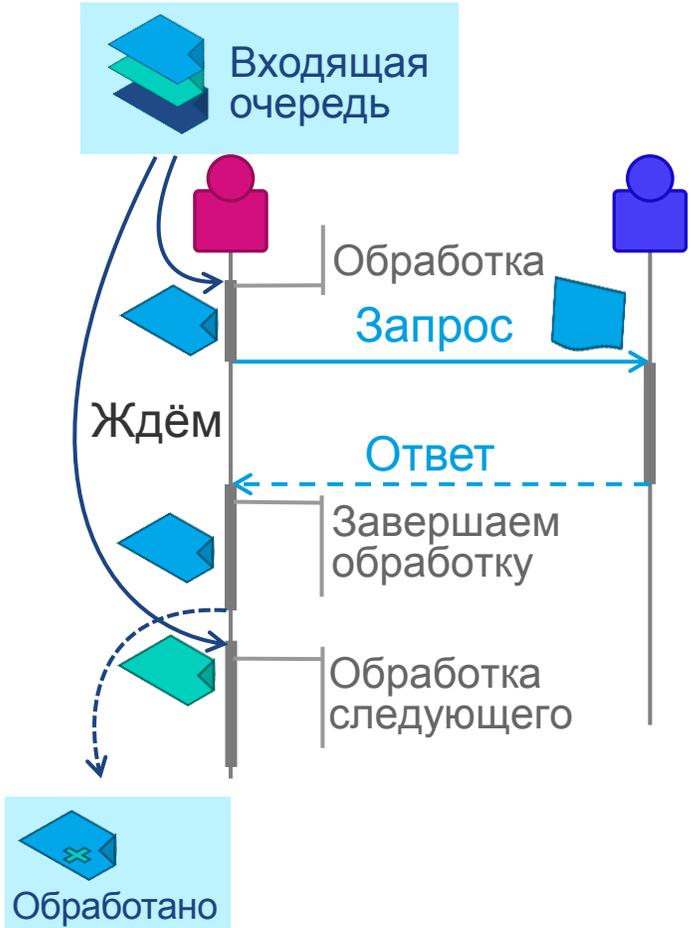
- Исторически микросервисы сделаны для горизонтального масштабирования за счёт увеличения числа экземпляров каждого сервиса
- Для этого усложнили приложение, и это было необходимое зло:
 - Асинхронное взаимодействие, транзакционность и консистентность в приложении, а не в базе данных, много данных в одном приложении
 - Сервисы и узлы кластеров падают, при восстановлении данные неконсистентны
 - Бизнес-запрос обрабатывает много сервисов, сообщения остаются при отмене запроса
- Когда архитектуру освоили, увидели дополнительный профит: жёсткая изоляция микросервисов, ограниченный объём и возможность переписать
 - Это кажется привлекательным, чтобы избавиться от легаси-монолитов
 - Однако увеличивает сложность и нагрузку и чревато появлением микролитов



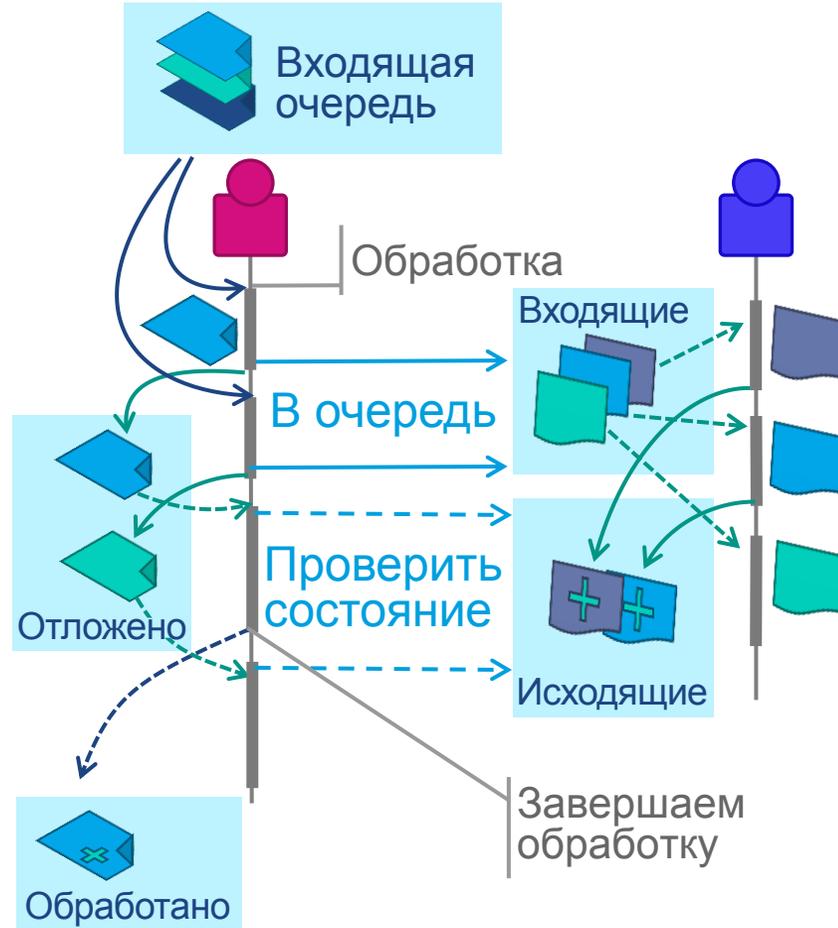
Системное мышление помогает в этом разобраться, построить модели

Взаимодействие сервисов

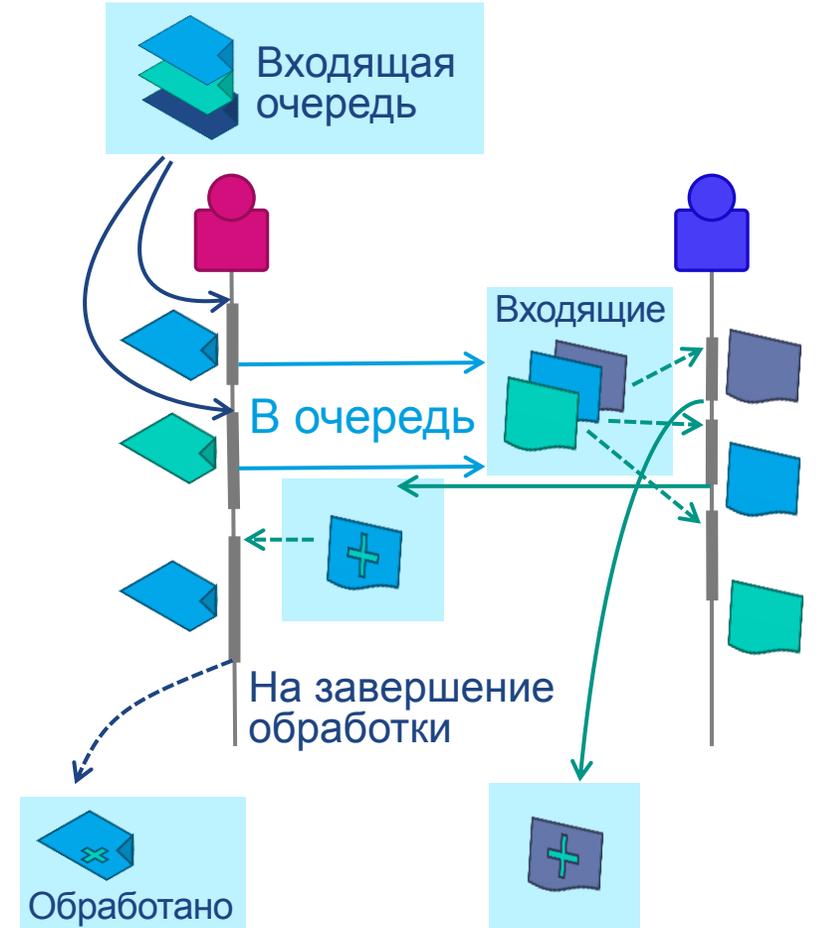
Синхронное



Асинхронное

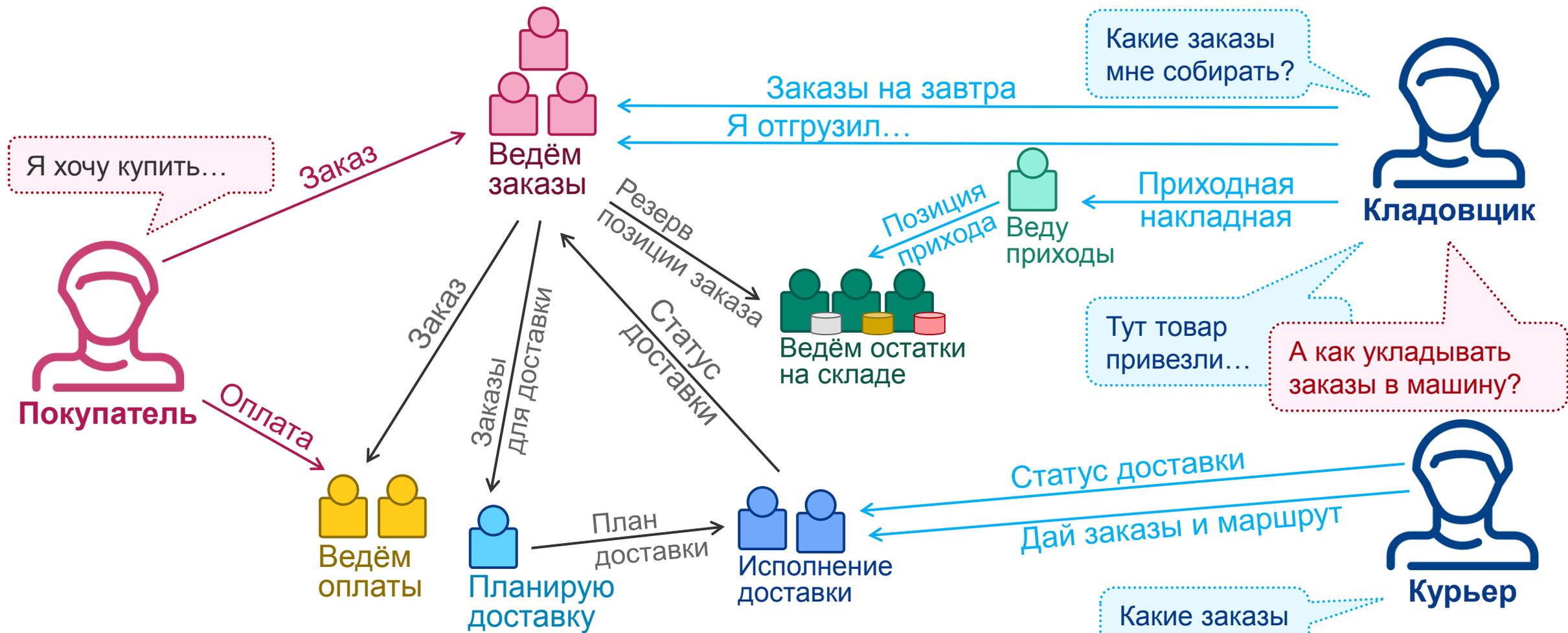


Реактивное



Это — для пары сервисов. А надо представить всю структуру обработки каждого запроса, возможные сбои и поведение при росте нагрузки

Сервисная архитектура — модель гномиков



Смотри доклады [«Модели приложения для разных парадигм программирования»](#) (SQA Days #27, осень 2020) и [«Визуальное проектирование масштабируемых приложений»](#) (TechLead-2021)

Пробки, я не успеваю. Что делать?

Модель гномиков: в чём системное мышление?

Модели гномиков не было, были частные задачи

- Покажите, как система будет работать, если отвалится один дата-центр?
- Как сделать, чтобы время отклика на двух дата-центрах было приемлемым?
- Фискальные регистраторы часто падают, особенно с новой прошивкой. Как уменьшить количество случаев, требующих ручного разбора?
- И другие подобные...

В каждом случае надо было рисовать понятные диаграммы, проигрывать на них процессы, ставить релевантные эксперименты. Системное мышление помогало этому

А модель гномиков — результат обобщения частных моделей



Подводя итоги

Компетенции мышления

1

- Уметь мыслить абстракциями и обобщениями, а также наборами фактов
- Размечать при восприятии тексты и картинки, соотнося их со своими моделями мира и дорабатывая свои модели
- Создавать новые ментальные объекты — абстракции, концепты и модели
- Использовать известные наработанные способы мышления

2

- Структурированно и понятно описывать свои модели для других
- Сопоставлять абстракции и модели с реальным миром, понимать зазор

3

- Фокусироваться на задаче
- Рефлектировать ход своего мышления и работать над ошибками
- Знать типовые проблемы мышления — когнитивные искажения, влияние эмоций и уметь мыслить с их учётом
- Использовать экзокортекс: компьютер, бумагу, таблицы, схемы



Мышление тренируемо, но путь развития надо проектировать: всем и сразу овладеть невозможно

Итоги

Системное мышление помогает

- Разобраться со сложными конструкциями, построить модель
- Видеть за структурной схемой динамику работы
- Увидеть за моделью реальный мир, оценивать разрыв между моделью и миром и учитывать его, принимая решения
- Видеть устройство бизнеса и места софта в нём, представить пользователя
- Разбираться в том, как софт устроен внутри, как реально работает

Надеюсь, рассказ вас убедил, и вы решите познакомиться глубже!



Оценить доклад



Максим Цепков



<http://mtsepkov.org>



[@MaximTsepkov](https://t.me/MaximTsepkov)

На сайте много материалов [по анализу и архитектуре](#), [Agile](#) и [менеджменту самоуправления](#), [моделям soft skill](#), мои [доклады](#), [статьи](#) и [конспекты книг](#)



Вакансии

Пишите на hr@custis.ru,
подходите с вопросами!