Введение в проектирование по предметной области (DDD)

Бакулева Екатерина. Хозяшев Павел. Вылегжанин Михаил.

В начале было….

Новый проект:

- Длительный жизненный цикл
- Сложная предметная область
- Высококвалифицированные разработчики
- Не надо всё реализовывать «вчера».

Определение DDD

DDD (Domain Driven Design) - предметно-ориентированная парадигма проектирования. В данной парадигме ключевым понятием является логика (бизнес-логика) вокруг которой строится приложение.

Плюсы и минусы, а везде ли использовать?

Плюсы:

- Сокращает издержки на сопровождение программного продукта;
- Сокращает издержки на документирование;
- Легкая адаптация приложения к изменениям в предметной области;
- Разработчики понимают предметную область.

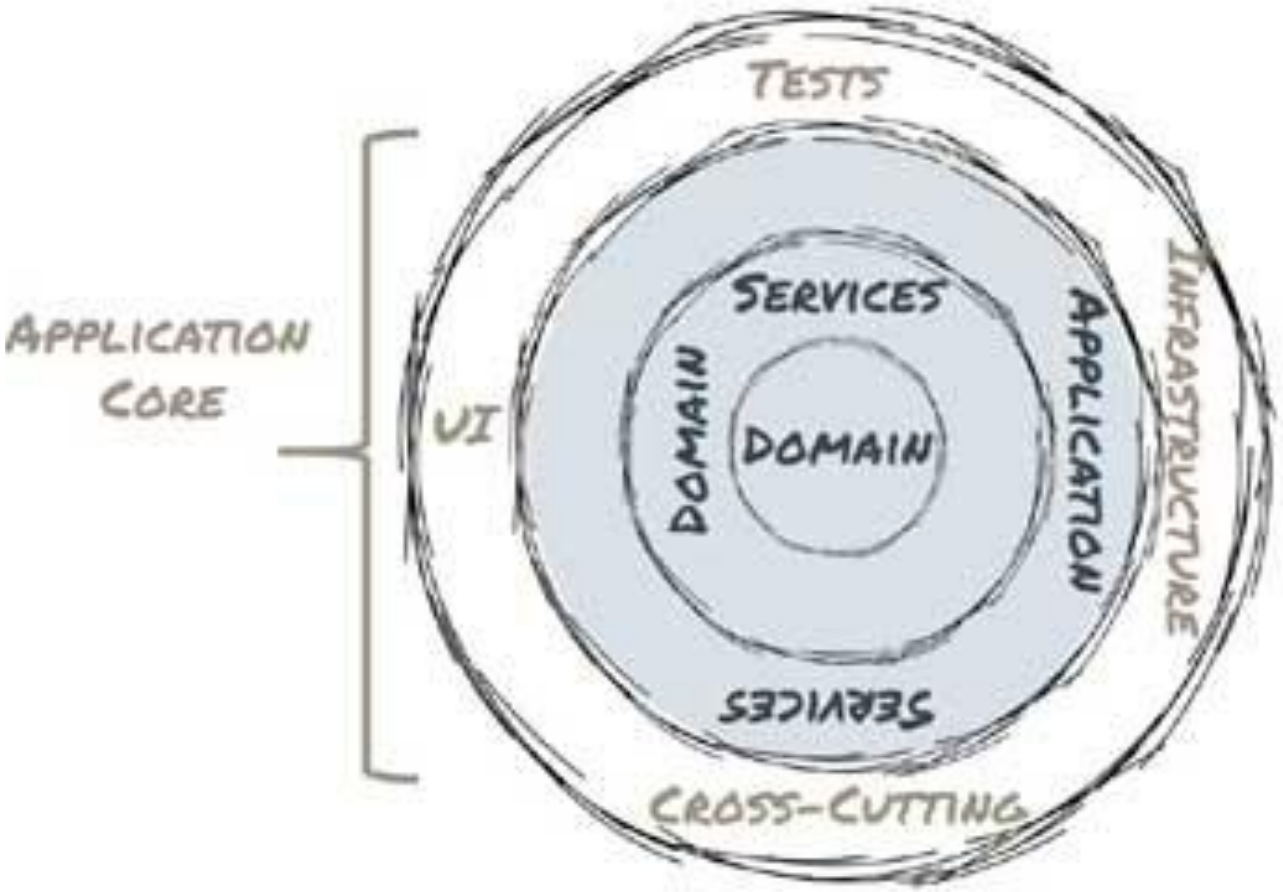
Минусы:

- Требует высокой квалификации разработчиков;
- Требует много времени на анализ информации и построение модели;
- В большинстве случае требует совместной работы разработчика и специалиста.

DDD. Основные понятия

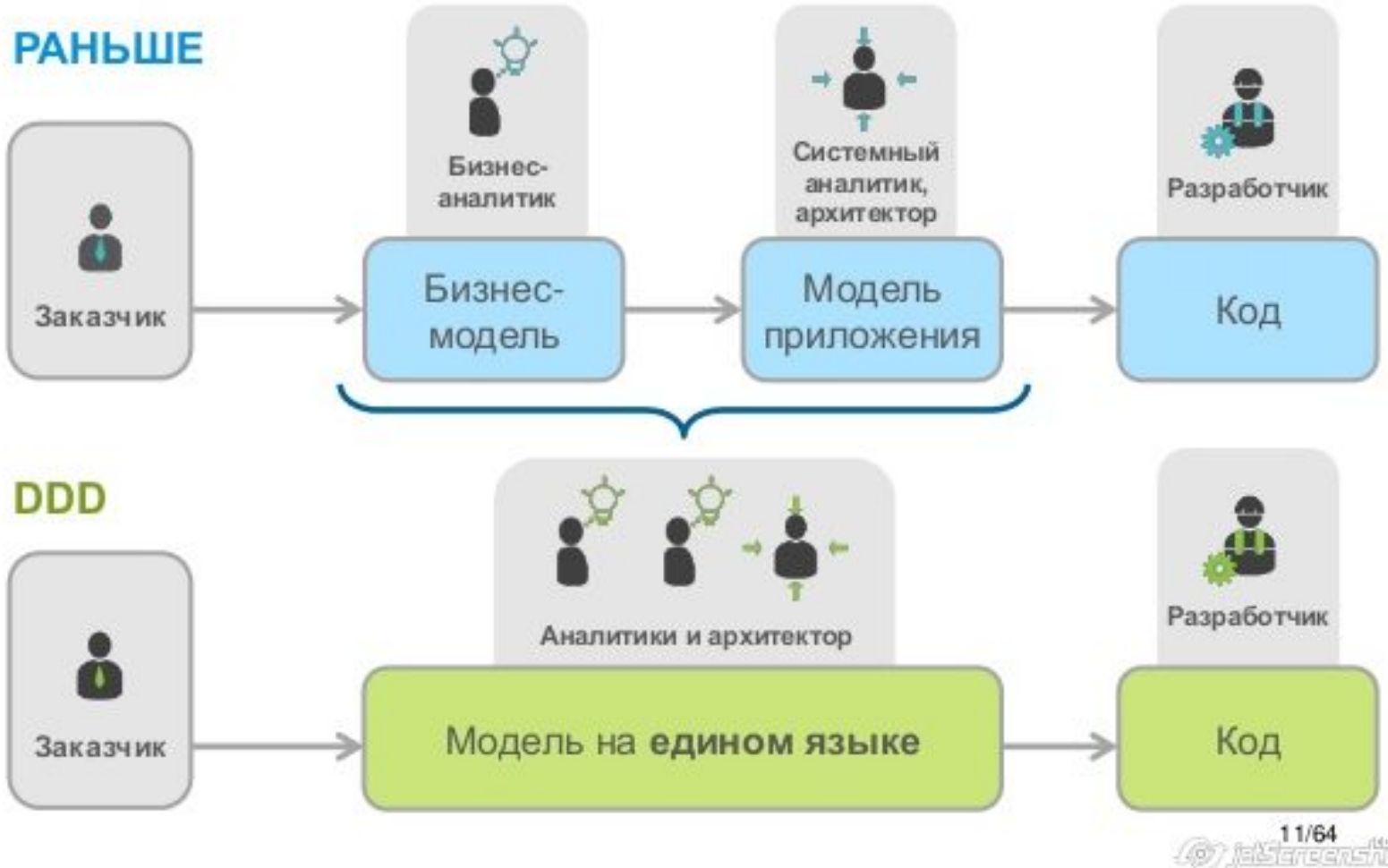
Domain Model (Модель)- это структурированные знания, которые связаны с определенной предметной областью.

Core domain (смысловое ядро) - часть домена, имеющая первостепенное значение для выполнения главной задачи.



CUSTIS

Основная идея DDD



Единый язык

Единый язык - общий и понятный всем, как специалистам так и разработчикам, язык общения. Является хранилищем переработанных знаний о предметной области и включает:

- название принципов высокоуровневой организации;
- имена классов, основных операций, шаблонов, модулей;
- метафорический образ системы;
- любые другие термины\словари используемые в модели.

Переработка знаний

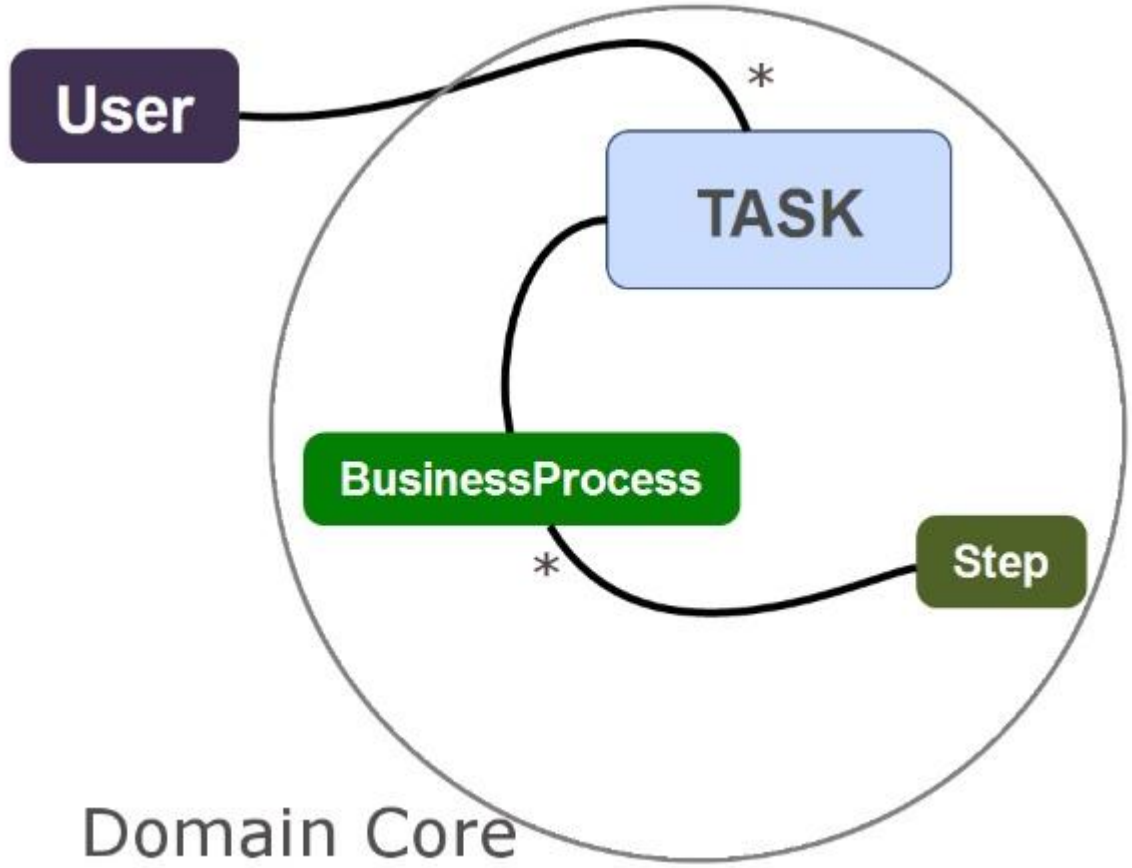
Начиная писать программу, невозможно знать достаточно.

В основе создания модели лежит переработка знаний.

Переработка знаний - получение информации о предметной области в удобной, понятной и структурированной форме.

Непрерывное обучение, в процессе итерационной разработки – постоянное изучение предметной области в процессе формирования модели и общения со специалистами\пользователями.

Task Tracker: первое приближение



Единый язык:

- Task Tracker
- User
- Task
- BussinesProcess
- Step
- Domain Core

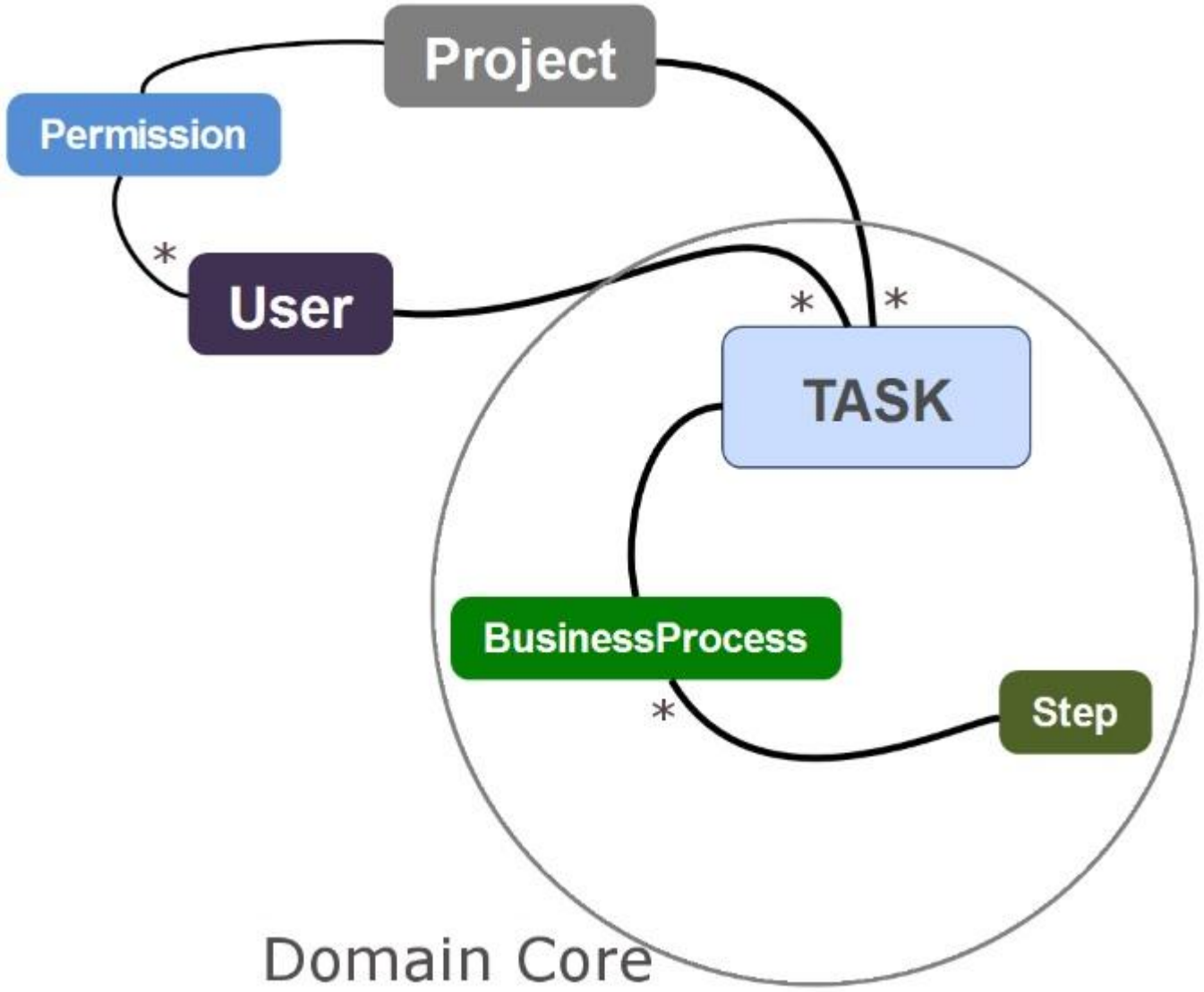
Углубленная модель

Углубленная модель – модель, которая наиболее точно описывает предметную область.

Инициализация:

- ответ на проблему в коде;
- пропущено важное понятие;
- модель не отражает предметную область;
- реализацию и модель не получается синхронизировать;
- Кризис\тупиковая ситуация при разработке.

Task Tracker : Углубленная модель



Единый язык:

- Task Tracker
- User
- Task
- BussinesProcess
- Step
- Domain Core
- Project
- Permission

Составляющие эффективного моделирования

- **Единый язык -** язык, основанный на модели;
- **Информоемкая модель -** модель максимально точно отражает предметную область;
- **Проектирование по модели (MDD)** – реализация полностью отражает модель;

- Спроектированная модель придерживается принятых в среде разработки парадигм программирования;
- **Дистилляция -** модель не содержит знаний не из предметной области;
- **Эксперименты и мозговые штурмы.**

DDD в крупномасштабных проектах

- Ограниченные контексты;
  - Карта контекстов;
- Взаимосвязи между контекстами:
  - Общее ядро;
  - Заказчик-поставщик;
  - Конформист;
  - Предохранительный уровень;
  - Отдельное существование;
  - Службы с открытым протоколом.
- Непрерывная интеграция;
- Разделение модели по уровням;
- Дистилляция:
  - Выделение ядра модели;
  - Выбор целей рефакторинга;
  - Декларативное программирование.

Заключение

Объекты модели

- Сущность
- Объект-значение
- Сервис
- Модуль
- Агрегат
- Фабрика
- Хранилище
- ???

DDD не говорит тебе, как писать код

- Абстракция. Инкапсуляция. Наследование. Полиморфизм.
- SOLID
- KISS
- GoF
- ORM
- …

Пример DDD : реализация биллинга телефонного оператора.

Кратко про биллинг:

Комплекс процессов и решений, ответственных за сбор информации об использовании телекоммуникационных услуг, их тарификацию, выставление счетов абонентам, обработку платежей (wiki)

Основные функции:

Тарификация услуг(звонков), хранение истории

Выставление счетов

Обработка платежей

Тарификация ~~услуг~~ звонков

Клиент может звонить(и звонки тарифицируются в соответствии с тарифом клиента), менять тариф, пополнять баланс

Клиент может звонить :

Клиент звонит на номер, после разговора у него с баланса списываются деньги в соответствии с его тарифом. Если баланс отрицательный, клиент не может позвонить – выдается сообщение об ошибке.

Объекты модели:

- Клиент
- Тариф
- Баланс
- Звонок? Номер?
- Операции:
- Звонить?
- Списание средств с баланса (Тарификация)?

Звонок изнутри, со стороны оператора:

С сим-карты поступает звонок на номер. Если на сим карте недостаточно денег, выдается сообщение об ошибке. Иначе звонок совершается. По окончанию звонка с баланса сим карты списываются деньги в соответствии с правилом тарификации по тарифу

Объекты модели:

- Сим карта
  - Клиент (! Оставили, потому что …)
  - Тариф
    - Правило тарификации
  - Баланс
- Звонок

Операции:

- Звонить => Сервис, который инициирует звонок
- Списание средств с баланса => Сервис тарификации

Посмотрим код.

Выставление счетов

В зависимости от тарифа ежемесячно клиенту выставляется общий счет, суммируется абонентская плата и подключенные к симке услуги, с учетом времени пользования. Если услугу подключили и отключили в один день, то в счет попадает сумма за весь день пользования

?????

Я: Абонентская плата – это признак тарифа?

Я: Подключенные услуги – что услуги, подключенные к тарифу?

В: Да, у тарифа может быть абонентская плата, или её может не быть.

В: Нет, услуги подключаются на сим карту, при смене тарифа должны оставаться

Новые объекты:

- Тариф
  - Абонентская плата
- Сим-карта
  - Услуга
- Общий счет

Всё хорошо, но нет

Тариф

Услуга

Клиент?

- Как оно работает?

- Откуда оно взялось?
- Что с ним будет дальше?
- Если мы это поменяем здесь и сейчас, кто должен увидеть эти изменения?

В: А давай ещё АРМ добавим и систему информирования? У нас ведь всё написано, это ж быстро

Я: Нет.

Список использованной литературы

- Эрик Эванс: Предметно-ориентированное проектирование (DDD): структуризация сложных программных систем;
- Флойд Маринеску и Эйбел Аврам: Domain-Driven Design Quickly;
- Презентация Tados’а по DDD: Предметно-ориентированное проектирование.
- https://habrahabr.ru/post/158277/ - Парадигмы программирования. Data Driven vs Domain Driven
- https://habrahabr.ru/post/258693/ - Снова о разработке на основе предметной области (Domain-Driven Design, DDD)
- https://habrahabr.ru/post/173893/ - Проектируем по DDD. Часть 1: Domain & Application

Первая версия примера

Непонятно, но прикольно получилось.