Современные технологии разработки ПО

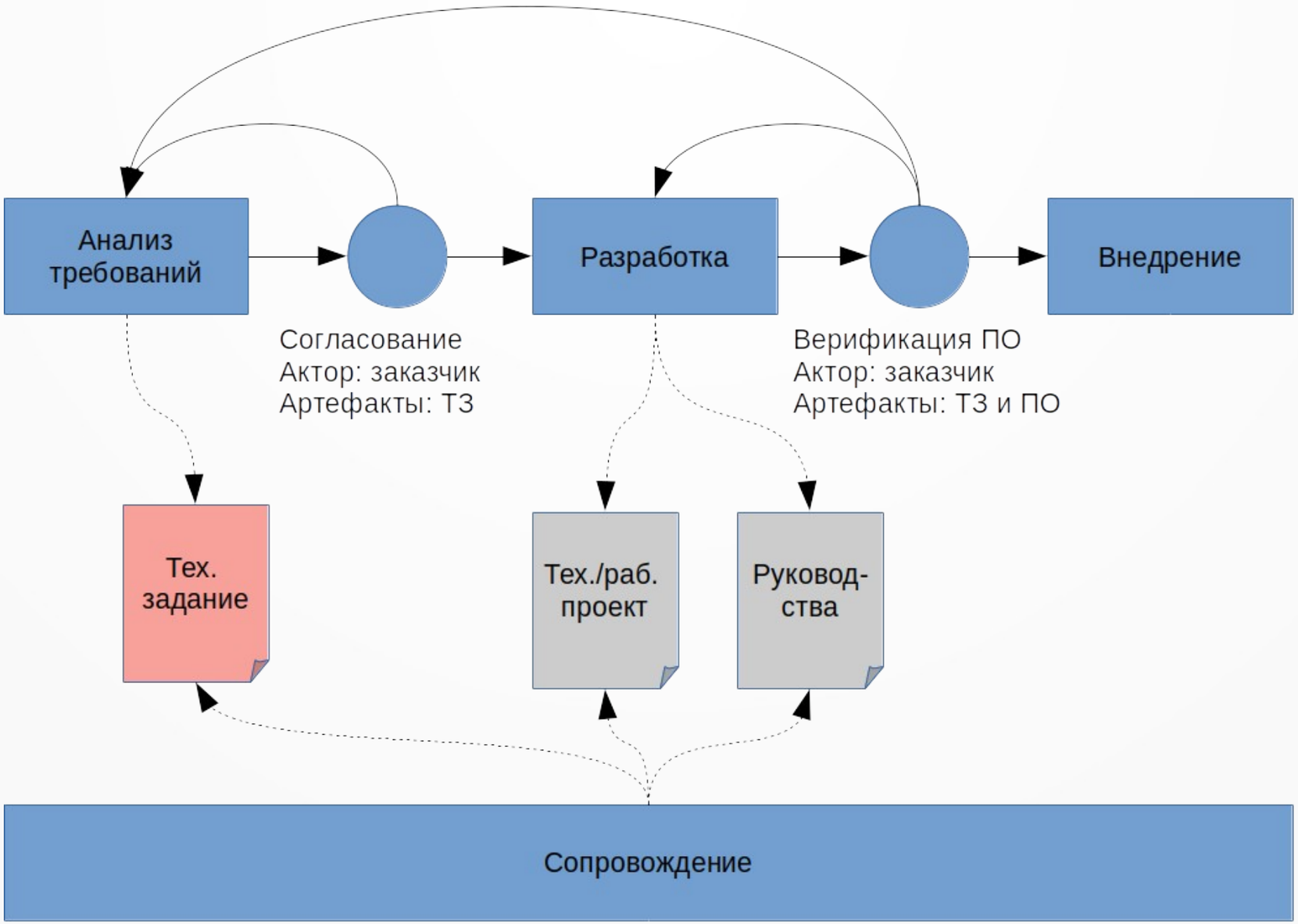
Предметно-ориентированное проектирование.
Изоляция предметной области.

Кафедра ИС, Петров А.А., petrov.a@kubsau.ru

Процесс разработки ПО
Основные этапы*

- Анализ требований
- Разработка
  - проектирование,
  - кодирование,
  - тестирование,
  - верификация.
- Внедрение
- Сопровождение

Процесс разработки ПО
Взаимодействие этапов*



Процесс разработки ПО
Модели*

- Последовательная (каскадная, водопадная)
- **Итерационная**
- И другие

Итерационная модель Методологии*

- Классические (Enterprise):
  - Rational Unified Process (RUP),
  - Microsoft Solutions Framework (MSF),
  - и другие.
- Гибкие (Agile):
  - Extreme Programming (XP),
  - Scrum,
  - Agile Unified Process (AUP),
  - и другие.

Основы гибкой разработки*

- Манифест гибкой разработки
  http://agilemanifesto.org
  - Ценности и принципы
- Экстремальная итеративность разработки

Ценности Agile*

1. Люди и взаимодействие важнее процессов и инструментов
2. Работающий продукт важнее исчерпывающей документации
3. Сотрудничество с заказчиком важнее согласования условий контракта
4. Готовность к изменениям важнее следования первоначальному плану

Резюме по гибкой разработке*

- Постоянное уточнение требований, а значит изменение кода — приветствуются
- Минимизация проектной и конструкторской документации — приветствуется
- «Грязное» проектирование — в порядке вещей
- Итеративный подход, постоянное усложнение и принцип Ready to show с самого начала проекта — обязательно!
- →
- **Процесс разработки должен быть изначально приспособлен к изменениям!**

Адаптивный процесс разработки
Инструменты реализации*

- Язык программирования
  - с поддержкой ООП
  - безопасный код
- Приёмы структуризации кода
  - шаблоны проектирования
  - рефакторинг кода
- Адаптивная модель предметной области
  - моделирование предметной области
  - рефакторинг модели предметной области
- Адаптивная архитектура:
  - изоляция предметной области
  - рефакторинг архитектуры

Предметно-ориентированное проектирование*

- Domain Driven Design (DDD)
- Набор принципов и схем, направленных на создание оптимальных систем объектов
- Сводится к созданию программных абстракций, которые называются моделями предметных областей

Преимущества DDD*

- Позволяет автоматизировать незнакомые разработчикам предметные области
- Позволяет вести разработку итерационно, постепенно усложняя ПО
- Позволяет значительно ускорить разработку сложного ПО

Недостатки DDD*

- Требует лояльности и гибкости заказчика (требование от Agile)
  - готовность заказчика участвовать в разработке
  - гибкость структуры заказчика по работе с контрагентами
- Требует сплочённой и профессиональной команды (требование от Agile)
  - умение использовать современные техники разработки ПО
  - готовность участвовать в изучении предметной области
- Но
  - некоторые приёмы DDD могут с пользой применяться и в неблагоприятных условиях (с осторожностью)

Сложность предметной области

- В процессе разработки ПО хватает всевозможных трудностей. Главное — это естественная сложность предметной области, к которой относится решаемая задача. Всякий раз, когда при разработке ПО возникает необходимость автоматизировать созданные человеком сложные системы, избежать этой сложности нельзя — ею можно только «овладеть».

Мартин Фаулер

Основные определения*

- **Область (Domain)** — предметная область, к которой применяется разрабатываемое программное обеспечение
- **Язык описания** — используется для единого описания модели предметной области
- **Модель (Model)** — описывает конкретную предметную область или её часть, является базой для автоматизации

Коммуникации*

- В чём проблема коммуникации:
  - жаргонизмы предметной области (противоречия и двусмысленности)
  - неумение задать правильный вопрос заказчику
  - общение внутри команды на другом языке затрудняет понимание предметной области
- Аналитик в качестве посредника между предметной областью и командой разработки — возможно, лишнее звено

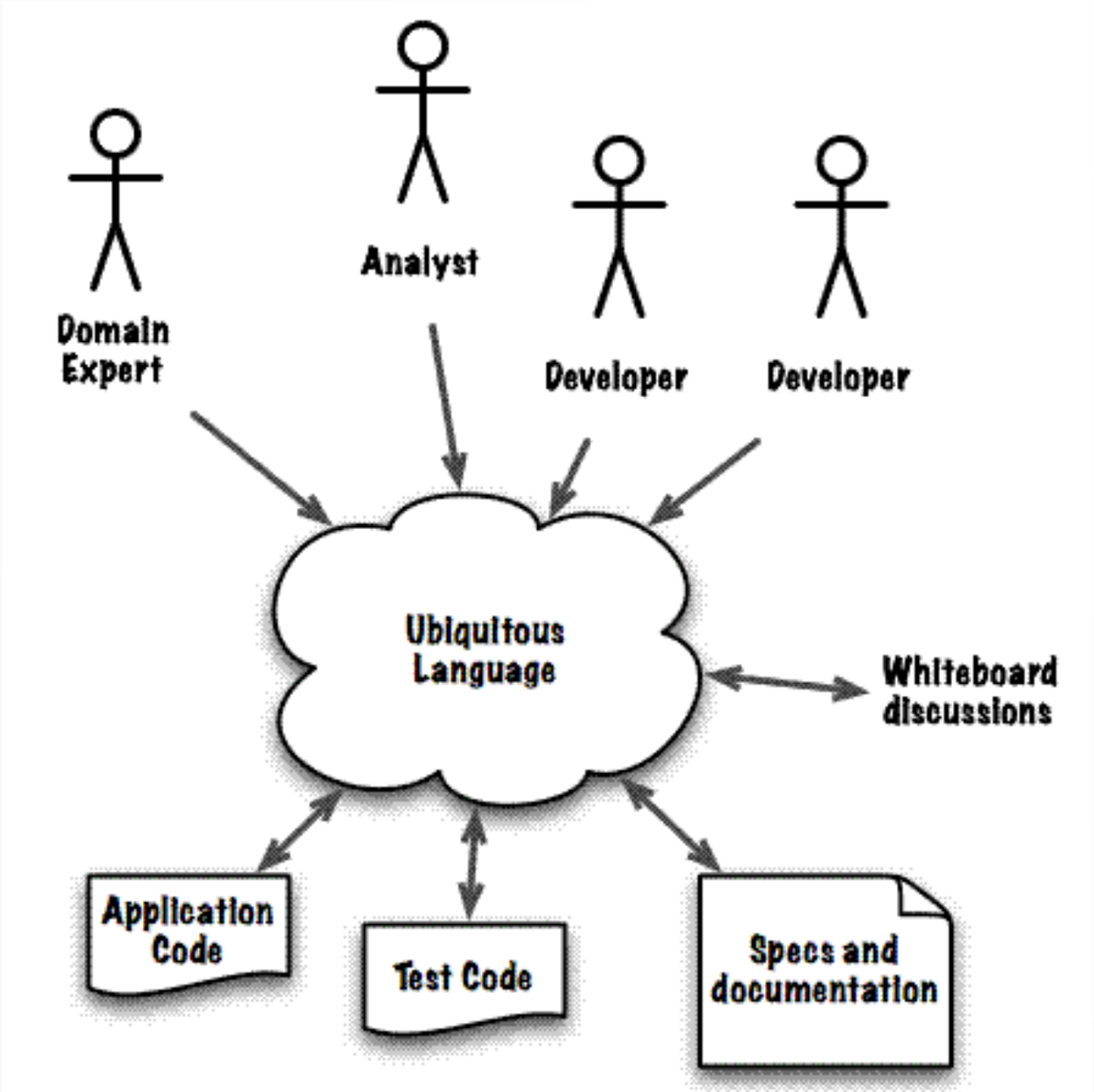
Единый язык коммуникации*

- Единый язык коммуникации — единый язык описания модели
- Одна команда — один язык
- Диалекты языка в описании моделей (контекстов)

Единый язык коммуникации
На пересечении технических жаргонов*

Рис. 2.3. Единый язык рождается "на пересечении" технических жаргонов

Единый язык коммуникации
Схема*



Alberto Brandolini. Strategic Domain Driven Design with Context Mapping
https://www.infoq.com/articles/ddd-contextmapping/

Средства коммуникации

- Словари предметной области
- Случаи использования
  - истории пользователя (Scrum)
- Диаграммы (UML и другие)
- Кодирование
  - наименование объектов кода,
  - документирование кода
- Карты контекстов
- Другое

Словарь предметной области*

- Набор ключевых абстракций в терминах предметной области
- Может быть сразу представлен в виде отношений между терминами (диаграммы классов)

Модель предметной области*

- Система абстракций, которая описывает избранные аспекты предметной области и может использоваться для решения задач, относящихся к этой области

Модель предметной области Особенности*

- Модель и архитектура ПО взаимно определяют друг друга
- Модель лежит в основе языка на котором говорят все члены группы разработки
- Модель — это дистиллированное знание
  - абстракция
- Априорная незавершенность модели
  - адаптивность модели
- Множество моделей

Модель предметной области
Множество моделей

- Необходимо выделять несколько моделей
- Предметная область «Перевозка грузов»
  - Модель маршрутов
    - зависимая
  - Модель передачи ответственности
    - основная
- Взаимодействие моделей
  - Планы моделей
  - Ограниченный контекст
  - И др.

Изоляция предметной области*

- Отделение модели предметной области от архитектурных решений
- Необходима для:
  - замены архитектурных решений без изменения кода модели предметной области;
  - распространение понятия ограниченного контекста на архитектурные решения.
- Реализуется как применение ШП Functional design или SRP
- Для реализации как правило используется «многоуровневая архитектура» (но не только она)

Многоуровневая архитектура
Зачем*

- Размазывание кода предметной области приводит к неадаптивности проекта:
  - бизнес-логика в скриптах страницы,
  - бизнес-логика в хранимых процедурах БД;
  - и т. п.
- Важно заранее предусмотреть:
  - замену компонентов проекта (СУБД, UI и т. д.);
  - масштабируемость проекта;
  - перенос на другие платформы;
  - раздельное тестирование и автотесты.
- Программу разделённую на уровни гораздо проще поддерживать, т. к. они имеют тенденцию развиваться разными темпами и обслуживать разные потребности

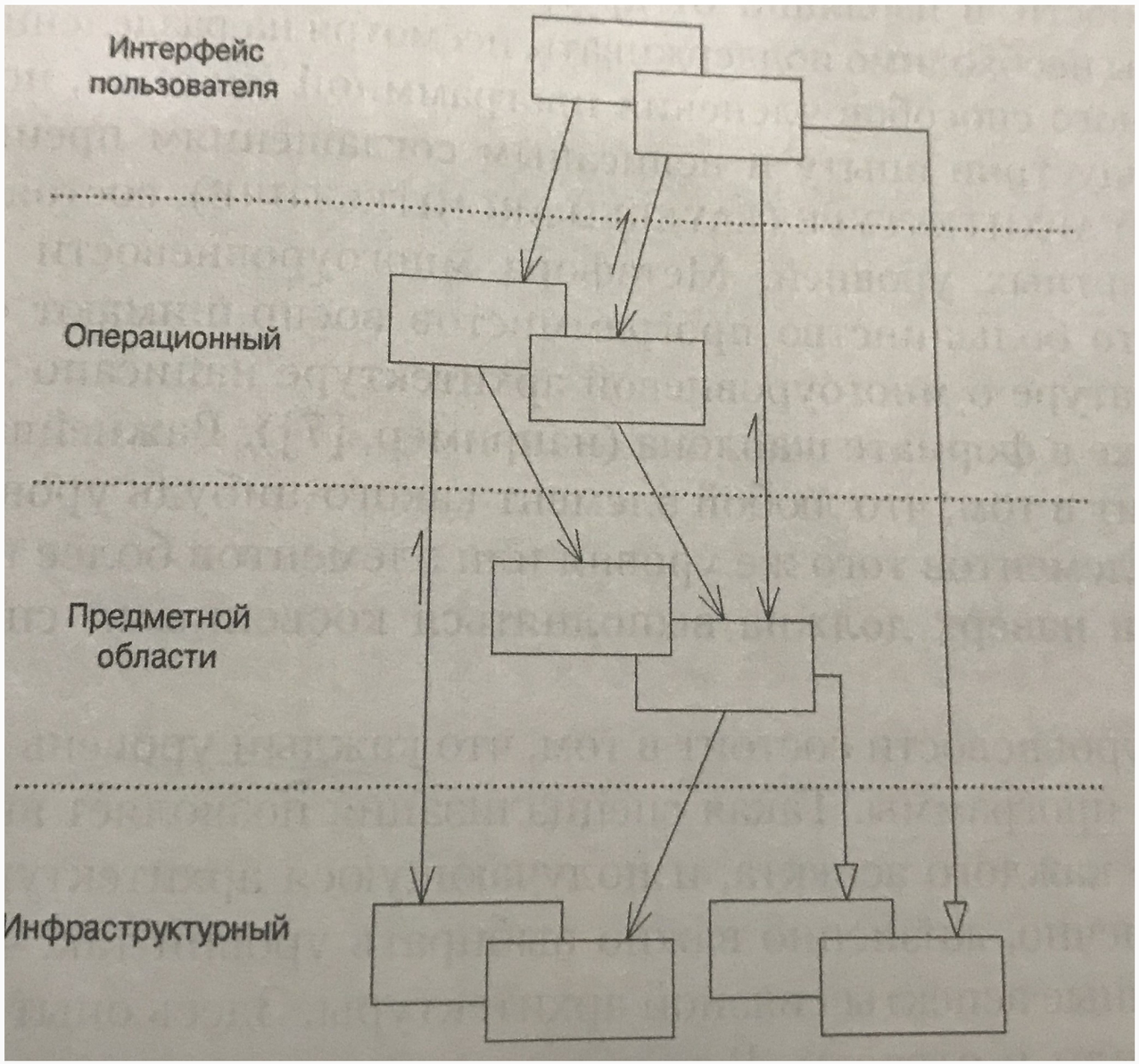
Многоуровневая архитектура
Уровни*

- **Интерфейс пользователя** (уровень представления) – UI
- **Операционный уровень** (уровень прикладных операций, уровень приложения) – AL
- **Уровень предметной области** (уровень модели, уровень бизнес-логики) – DL
- **Инфраструктурный уровень** (уровень доступа к данным) – IL

Многоуровневая архитектура
Принцип зависимости уровней*

- Каждый уровень:
  - зависит только от нижележащего слоя,
  - может существовать без вышерасположенных слоёв.
- Для связи с верхними уровнями используются:
  - обратные вызовы (callback);
  - ШП Observer;
  - стандартные архитектурные шаблоны:
    - Model-View-Controller (MVC),
    - Model-View-Presenter,
    - Naked objects,
    - и др.

Многоуровневая архитектура
Условная схема*



Интерфейс пользователя*

- Отвечает за:
  - вывод информации пользователю,
  - интерпретацию команд пользователя.
- Внешним действующим субъектом может быть не человек, а другая программа

Операционный уровень*

- Определяет задачи, связанные с конкретным действием в UI (команда пользователя или потребность в информации) и распределяет их между объектами предметной области
- Не хранит состояний объектов предметной области
- Может играть интегрирующую роль - взаимодействовать с операционными уровнями других систем
- Наиболее близкий ШП: Fasade

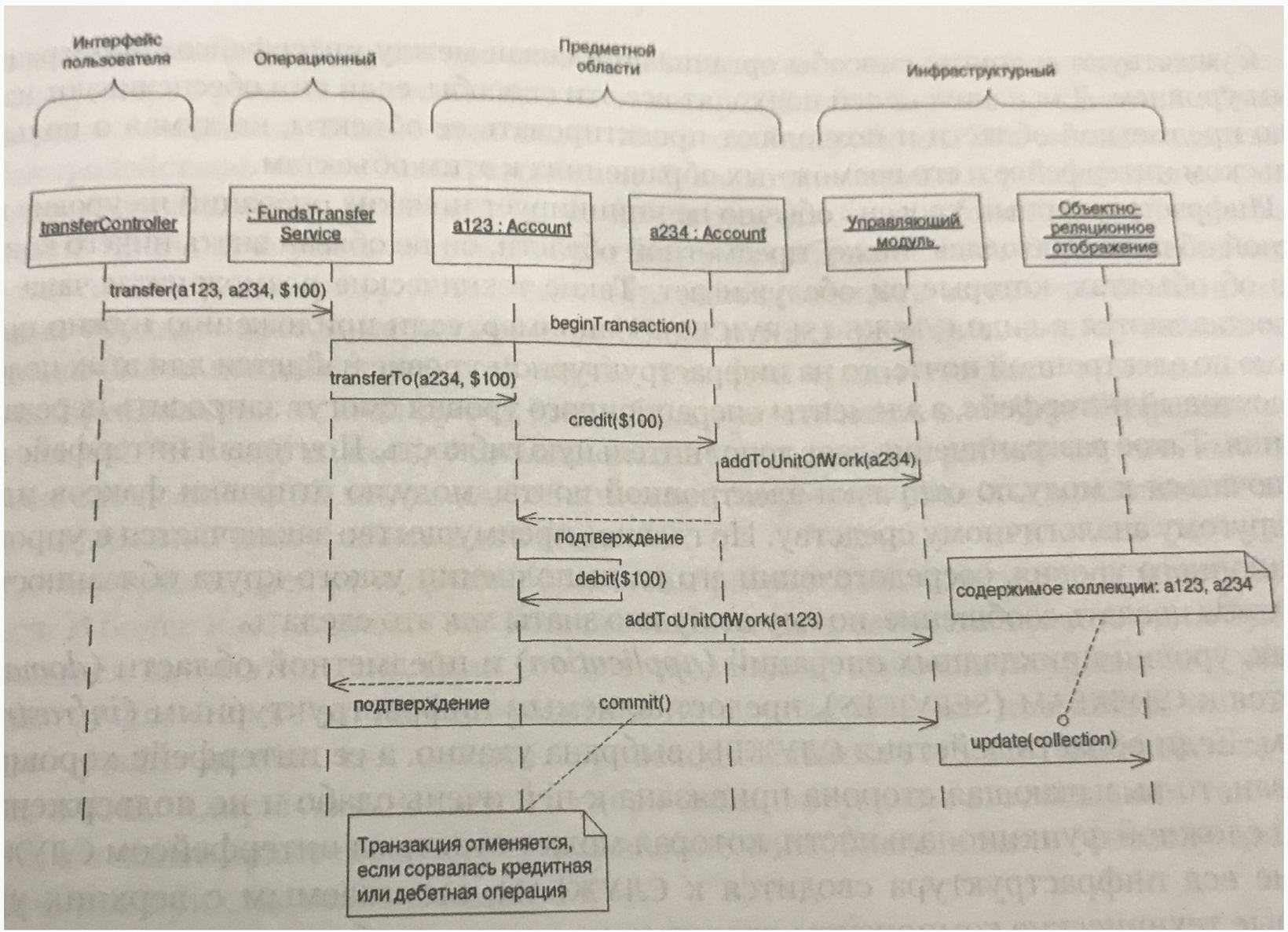
Уровень предметной области*

- Отвечает за:
  - представление понятий прикладной предметной области,
  - рабочие состояния,
  - бизнес-регламенты (поведение модели).
- Этот уровень является главной, алгоритмической частью программы

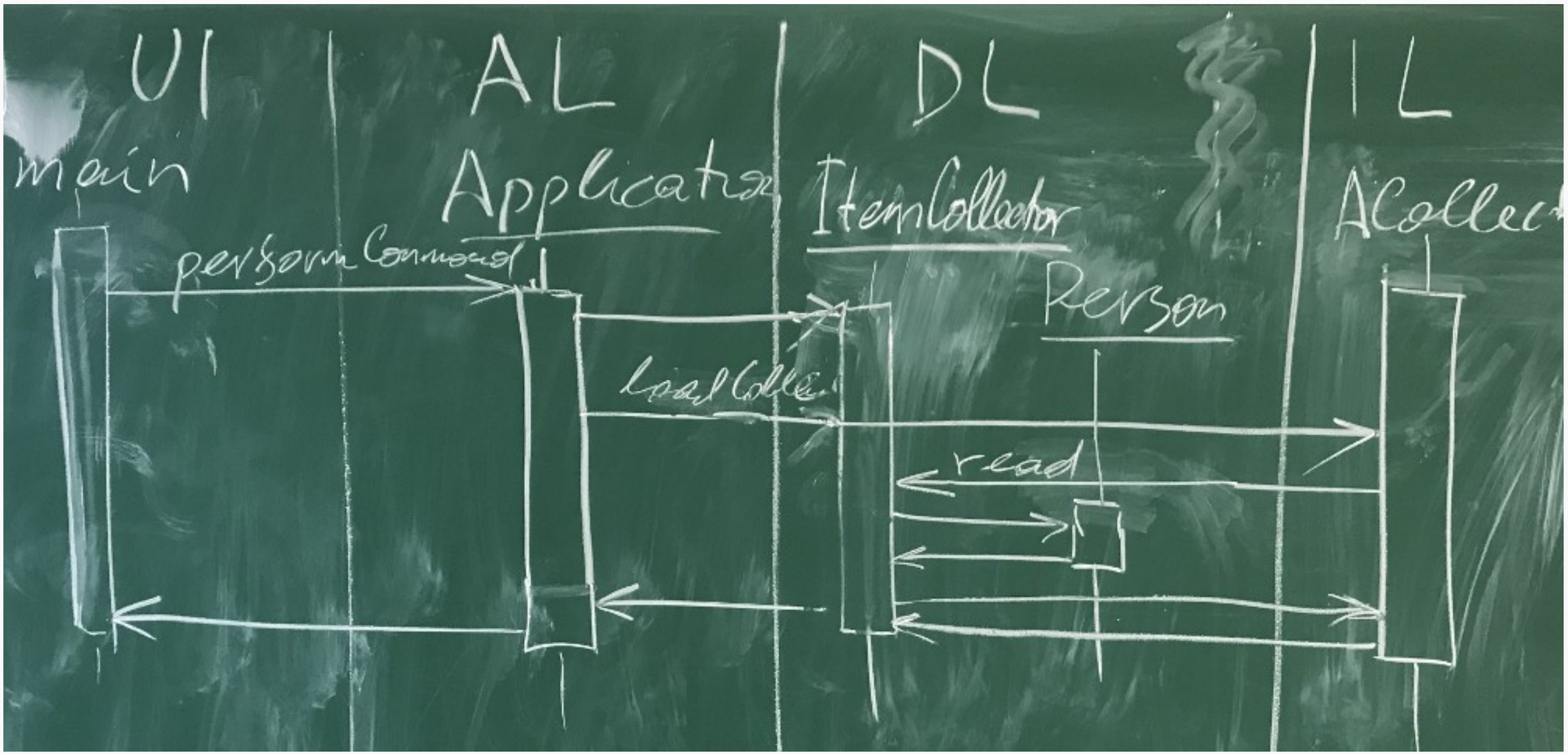
Инфраструктурный уровень*

- Слой технических сервисов
- Обеспечивает техническую поддержку для верхних уровней:
  - передачу сообщений на операционном уровне;
  - непрерывность существования объектов на уровне модели (хранение, транзакционность и т.д.);
  - службы передачи сообщений;
  - почтовые службы;
  - и т.д.

Многоуровневая архитектура
Пример диаграммы последовательности*



Многоуровневая архитектура
Пример диаграммы последовательности*



Архитектурные среды (фреймворки)*

- Набор наиболее востребованных заготовок для разработки приложения на определённой платформе (чаще всего Web)
- Примеры:
  - Ruby on Rails (RoR),
  - Jakarta EE (ранее Java 2 Enterprise Edition или J2EE),
  - Node.js
  - SIMODO
  - и т. д.
- Выбор фреймворка — выбор ограничений, которые вам необходимо будет учитывать при разработке ПО

Интеллектуальный интерфейс

- Противостоит DDD/MDD
- Плюсы:
  - быстрый результат в простых приложениях;
  - низкие требования к квалификации разработчиков.
- Минусы:
  - серьёзные проблемы при усложнении приложения (дублирование кода, код-спагетти и т.д.);
  - проблемы с масштабируемостью;
  - сложнее выполнить интеграцию приложений (только через общую БД);
  - невозможно сменить среду или платформу (нужно всё переписывать);
  - невозможно изменить методику, например перейти на DDD/MDD (нужно всё переписывать).

Изоляция предметной области
Пример

Вопросы?