Современные средства разработки ПО

Методологии. Введение в DDD (предметно-ориентированное проектирование).

Фетисов Михаил Вячеславович
fetisov.michael@bmstu.ru

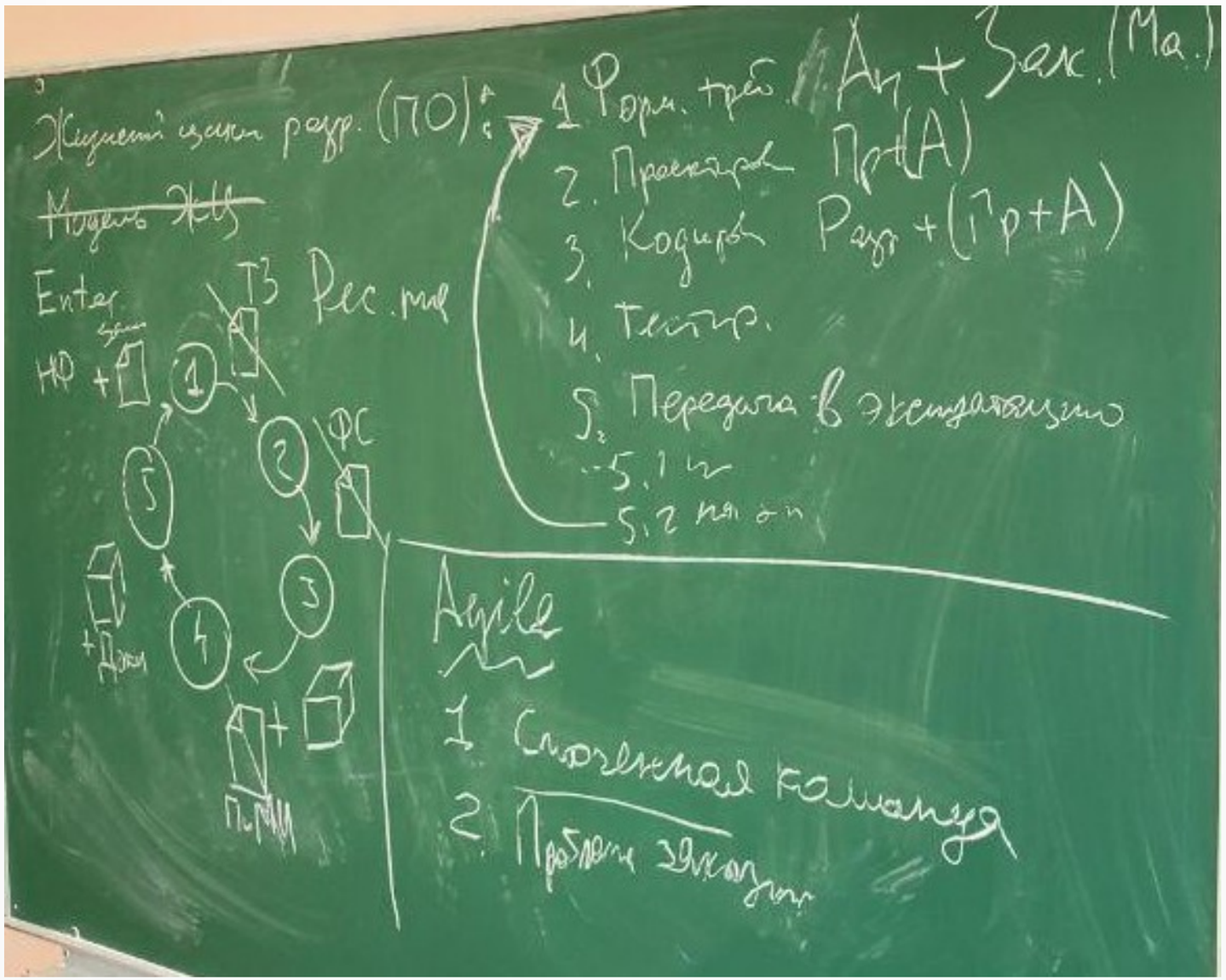
Итерационная модель Методологии

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

Enterprise
Особенности методологий

- Разработка разделяется на итерации
- Каждая итерация разделяется на этапы
- Каждый этап выполняется своим набором ролей
- Каждый этап не может начаться без наличия артефакта (документа или продукта)
- Каждый этап заканчивается формированием артефакта
- Артефакты используются для верификации
- Такая схема позволяет достичь **хорошего качества** при **низкой лояльности заказчика** и **среднем уровне квалификации сотрудников**
- Проблема в больших накладных расходах на поддержку жизненного цикла, а значит, в длительности итераций. В итоге, **стоимость разработки значительна**.

Схема разработки по Enterprise методологии



Гибкая разработка
Определение

- **Гибкая методология разработки (agile)** — обобщающий термин для целого ряда подходов и практик, основанных на ценностях Манифеста гибкой разработки программного обеспечения и 12 принципах, лежащих в его основе

Гибкая разработка
Минифест

- Agile-манифест разработки программного обеспечения: http://agilemanifesto.org/iso/ru/manifesto.html
  - Люди и взаимодействие важнее процессов и инструментов
  - Работающий продукт важнее исчерпывающей документации
  - Сотрудничество с заказчиком важнее согласования условий контракта
  - Готовность к изменениям важнее следования первоначальному плану

Гибкая разработка
Особенности

- Переворачивает разработку в методологиях Enterprise, **исключая документацию** как необходимый связующий элемент **верификации** и **связи** этапов жизненного цикла.
- Позволяет радикально быстро реагировать на изменение требований. Более того, изменение требований принимается естественным и включено в процесс разработки, который становится **более итерационным**.
- Верификация выполняется с использованием продукта в конце каждой итерации.
- В результате может быть достигнута **высокая производительность** при автоматизации сложных, и особенно, **плохо формализуемых** предметных областей.
- Но каким образом это достигается? Что для этого необходимо в первую очередь? Новые методологии или что-то другое?

Предметно-ориентированное проектирование Определение

- **Предметно-ориентированное проектирование** (Domain Driven Design, DDD) — Набор принципов и схем, направленных на создание оптимальных структур объектов, устойчивых к изменениям.

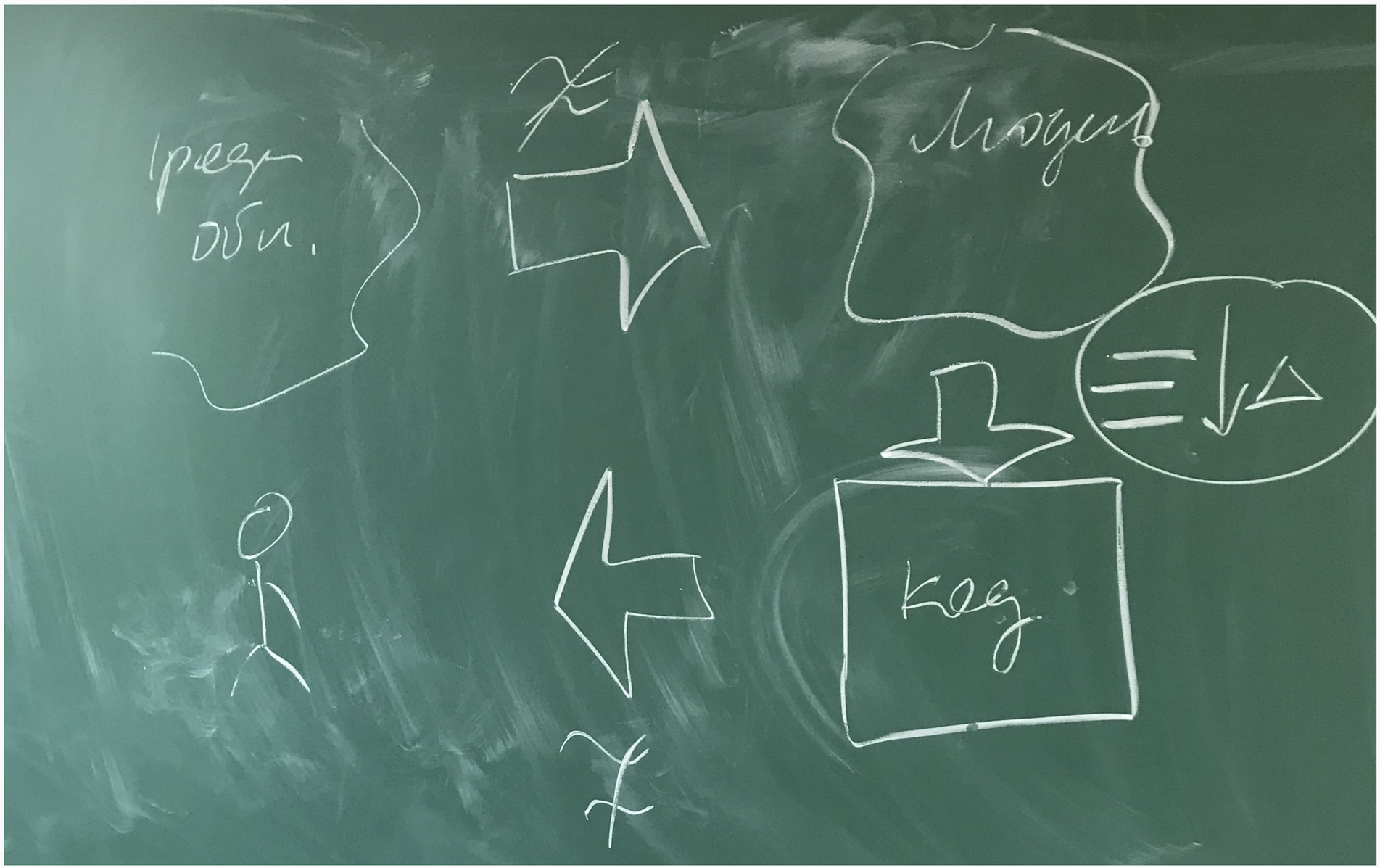
Предметно-ориентированное проектирование Определение 2

- **Предметно-ориентированное проектирование** (Domain Driven Design, DDD) — это подход к разработке ПО, в котором мы:
  - сосредотачиваемся на основной предметной области;
  - исследуем модели предметной области в творческом сотрудничестве с теми, кто имеет в ней опыт, и теми, кто разрабатывает ПО;
  - разговариваем на одном языке (предметной области) в рамках чётко ограниченного контекста.

Предметно-ориентированное проектирование Особенности

- Техника проектирования и разработки, которая хорошо подходит для гибкой разработки.
- Хорошо сочетается с микросервисной архитектурой.

Модель жизненного цикла при проектировании по модели в DDD



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)

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

Понятия и словарь предметной области

- Термины модели предметной области — основные понятия
- Не рекомендуется создавать новые понятия, сначала нужно убедиться, что их нет в модели и оно действительно необходимо

Изоляция предметной области Определение

- Отделение модели предметной области от инфраструктурных и других решений
- Необходима для:
  - замены архитектурных элементов без изменения кода модели предметной области;
  - распространение понятия ограниченного контекста на архитектурные решения.

Изоляция предметной области
Описание

- Реализуется как применение ШП Functional design или SRP
- Для реализации используются различные архитектуры:
  - MVC,
  - многоуровневая архитектура,
  - «чистая архитектура»,
  - гексогональная архитектура,
  - и тд.

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

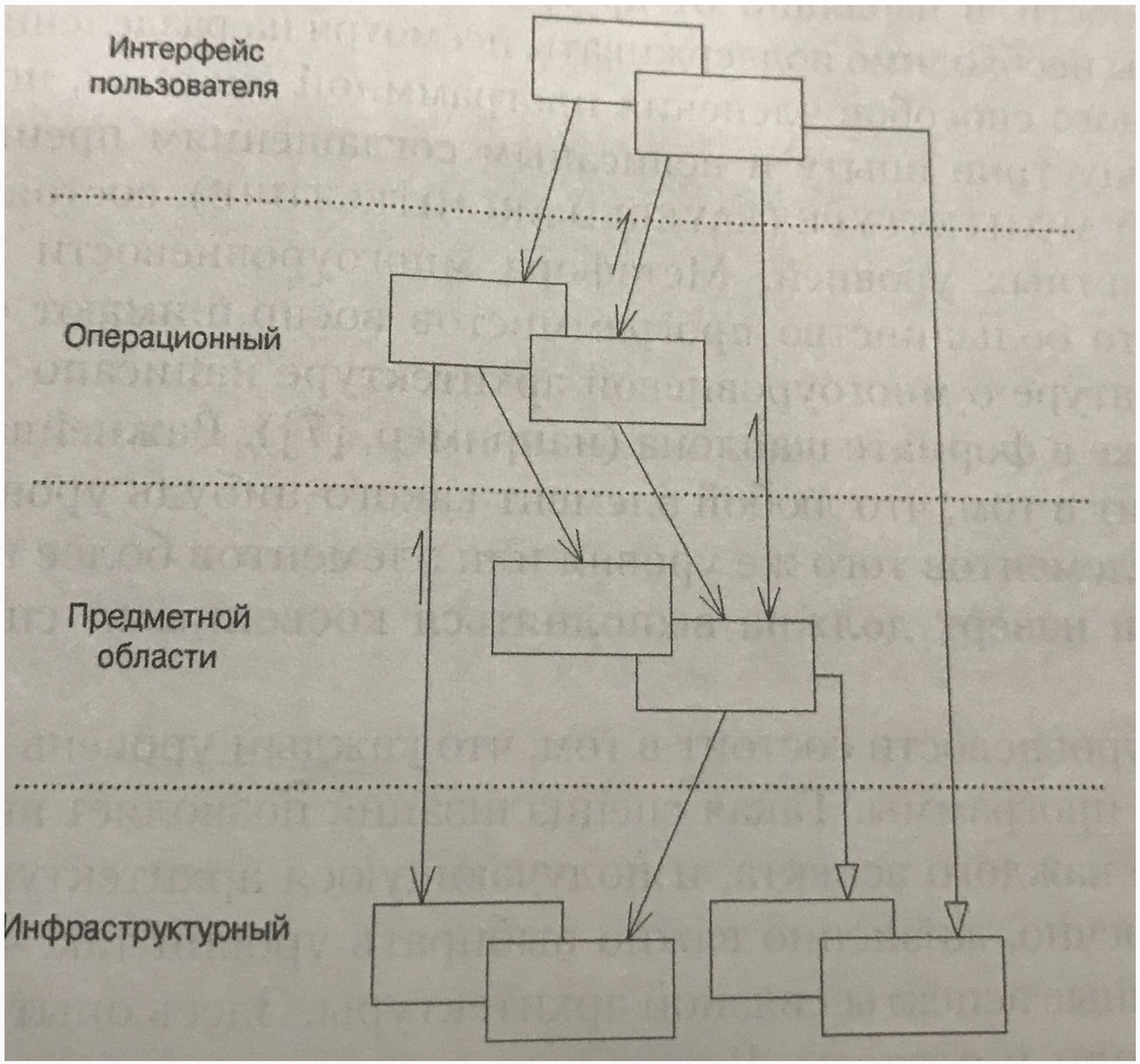
Многоуровневая архитектура
Уровни

- **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

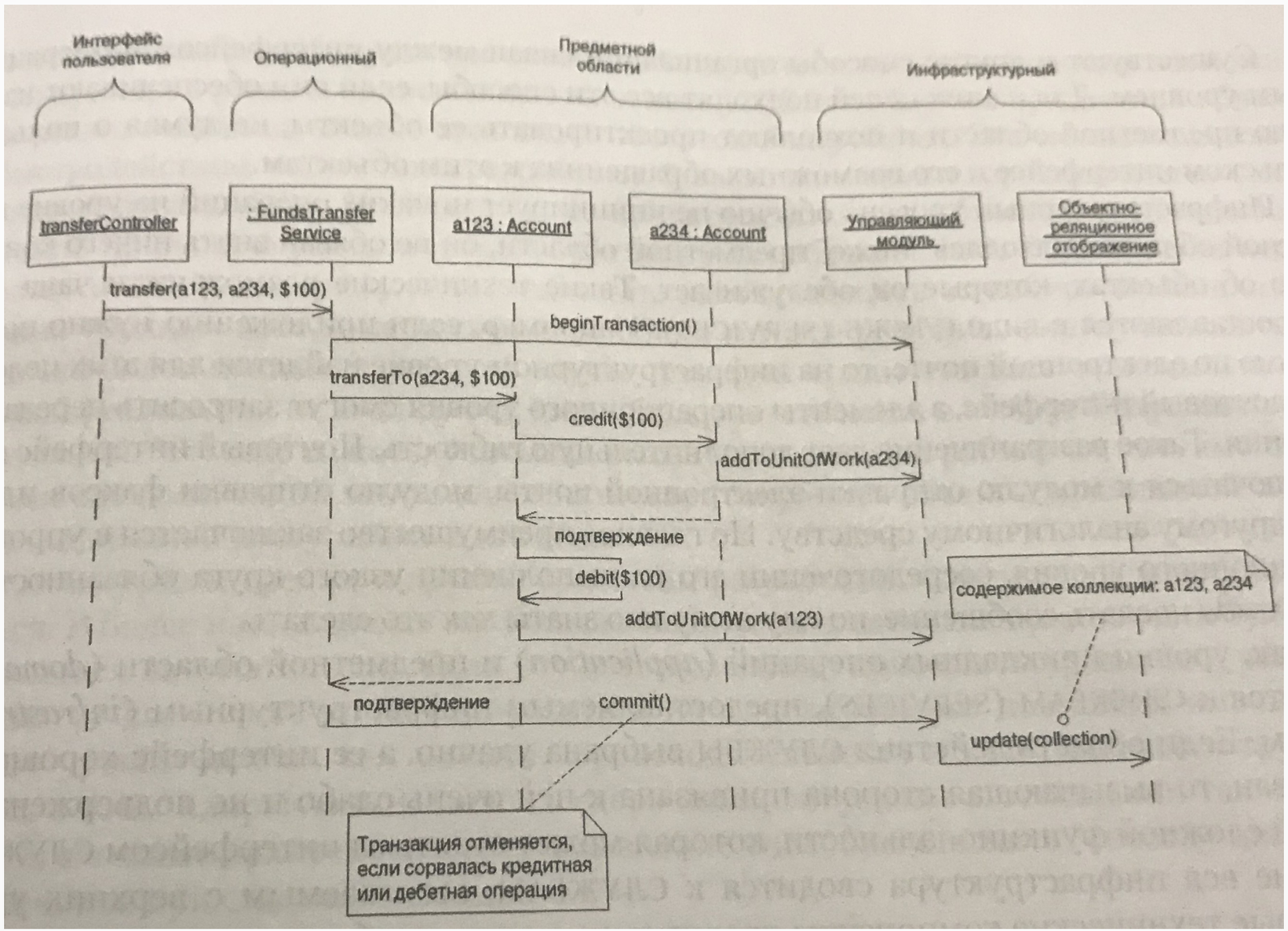
Уровень предметной области

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

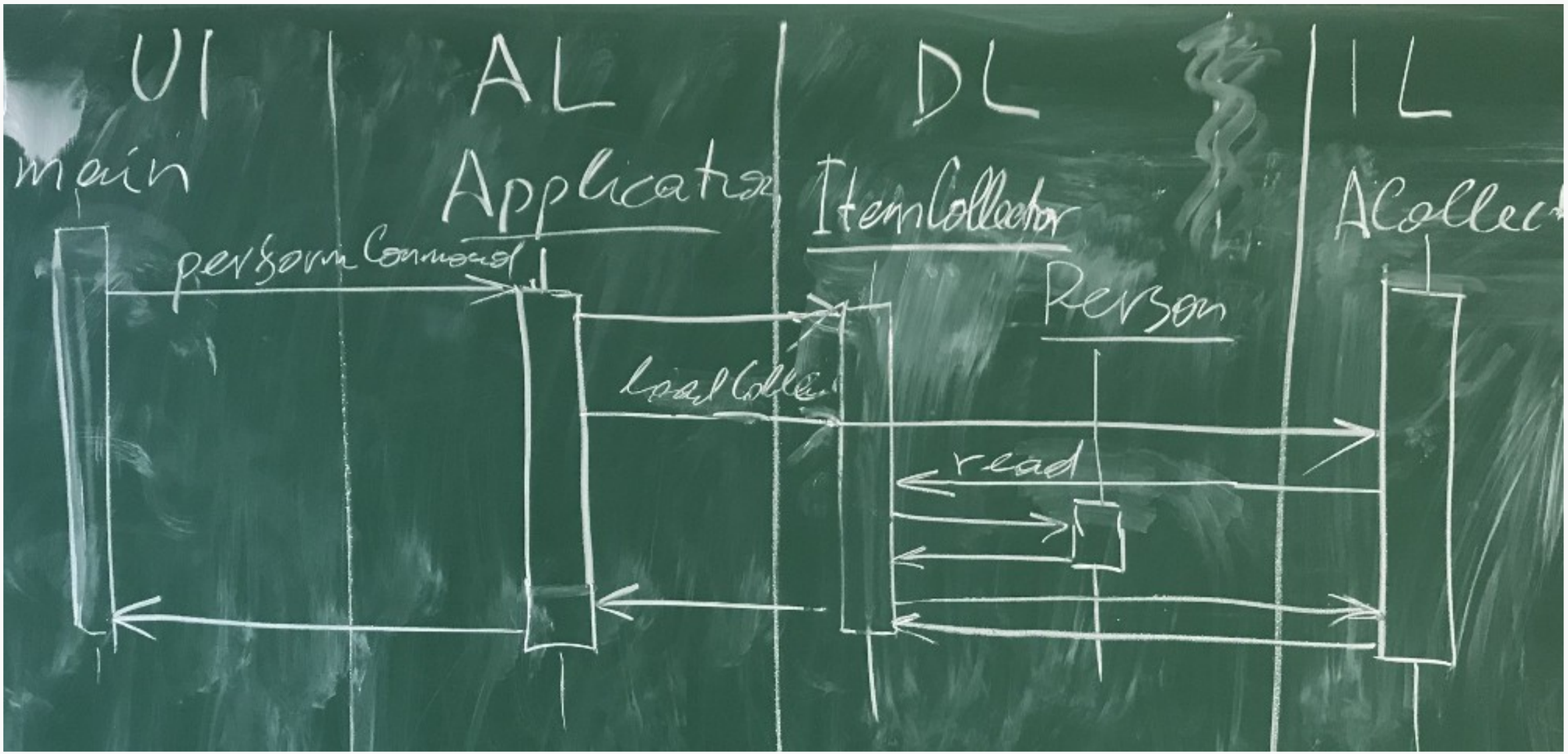
Инфраструктурный уровень

- 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



Вопросы?

Фетисов Михаил Вячеславович
fetisov.michael@bmstu.ru