Унифицированный процесс разработки ПО

Технологии разработки программного обеспечения

Виноградова М.В.
МГТУ им. Н.Э. Баумана
Кафедра СОИУ (ИУ5)

RUP - Rational Unified Process

- Результат различных подходов;
- Использует UML (визуальное проектирование);
- Унифицирует:
  - подходы к разработке;
  - многие методологии;
- Поддерживается Rational / IBM;
- Необходим для:
  - Больших и сложных систем;
  - Больших команд разработчиков с различным опытом;
  - Автоматизации разработки / согласования / модификации / сохранения системы.

Особенности RUP

- Управляется прецедентами

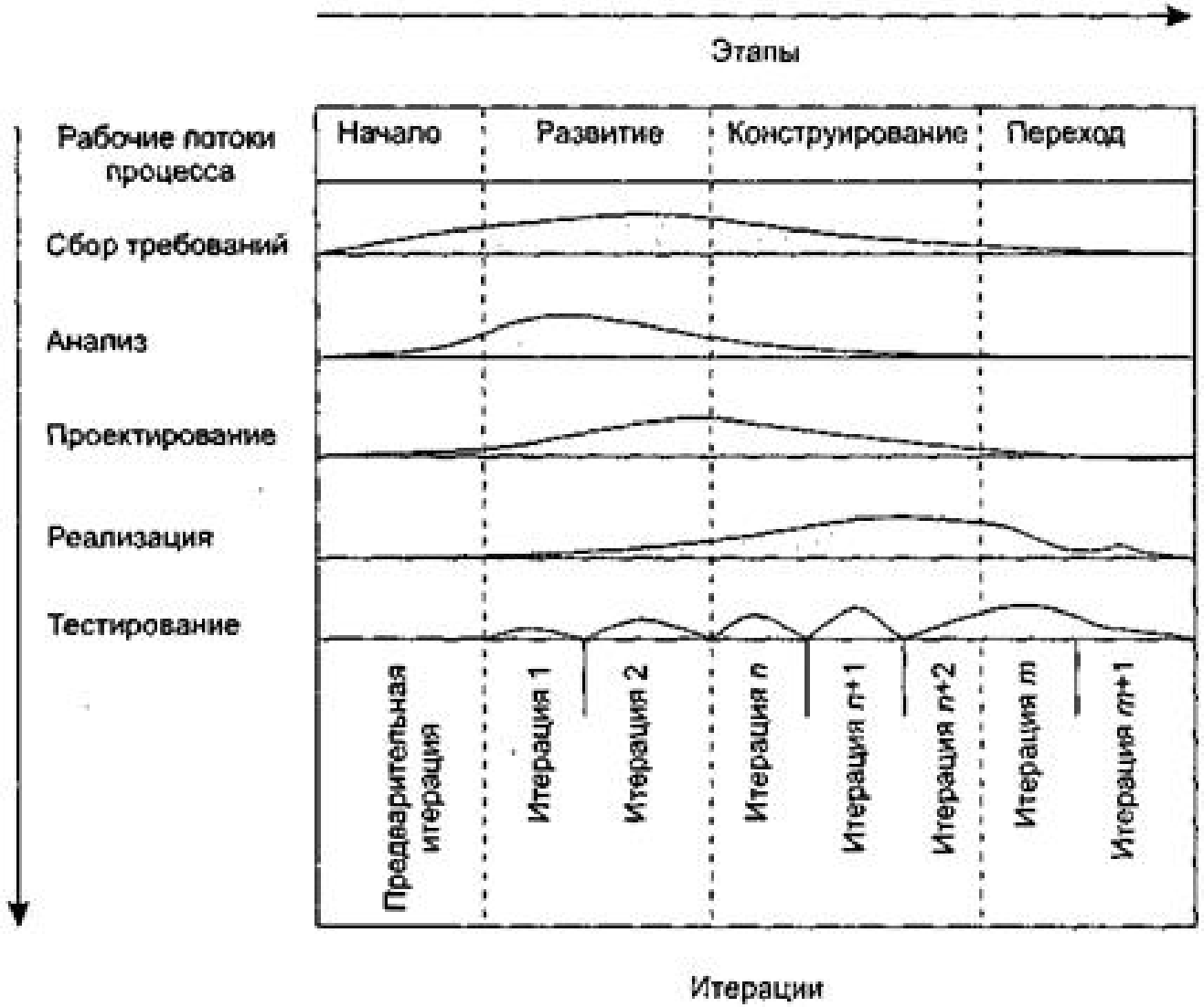
  Требования -> Анализ -> Проектирование -> Развертывание -> Реализация -> Тестирование

- Ориентирован на архитектуру

  (Сначала создается архитектура, потом она влияет на проект)

- Итеративность и инкрементность
  - Декомпозиция на мини-проекты (итерация), выполнение итераций.
  - Результат итерации - приращение, инкремент системы.
    (Инкремент функций в конце разработки, в начале - уточнение)

График RUP


Этапы
Рабочие потоки процесса
Начало
Развитие
Конструирование
Переход
Сбор требований
Анализ
Проектирование
Реализация
Тестирование
Предварительная итерация
Итерация 1
Итерация 2
Итерация n
Итерация n+1
Итерация n+2
Итерация m
Итерация m+1
Итерации

Этапы RUP

- **Начало** - спецификация представления продукта;
- **Развитие** - планирование действий и требуемых решений;
- **Конструирование** - построение ПО в виде серии инкрементных итераций;
- **Переход** - внедрение ПО в среду пользователя (промышленное производство, доставка и применение)

RUP - начало

- **Анализ и планирование требований (Начало):**
  - определение области применения СОИУ (предназначение, границы, внешние интерфейсы)
  - определение основных черт архитектуры (оценка реализуемости для основных сценариев)
  - создание демонстрационного макета (основные идеи, пользовательский интерфейс/ интересующие задачи)
  - определение основных (критических) рисков
  - определение общей стоимости и плана проекта

- **Цель:** убедиться, что система осуществима

RUP - развитие

- **Проектирование (развитие):**
  - Создания базового уровня архитектуры: модели, описание архитектуры, реализация (архитектура + возможности системы + важное для заказчика)
  - определение существующих рисков, их отслеживание и устранение (и уровень качества)
  - определение финансового плана, персонала и затрат -- составление плана итерации следующего этапа
  - определение до 80% прецедентов (в том числе анализ)

- **Цель:** создать архитектурный базис, обеспечить возможность выполнения

RUP - конструирование

- **Построение (конструирование):**
  - управление рисками, оптимизация процессов
  - отслеживание критических и существующих рисков
  - полная разработка и тестирование компонентов
  - оценка реализации ПО (оценка качества)

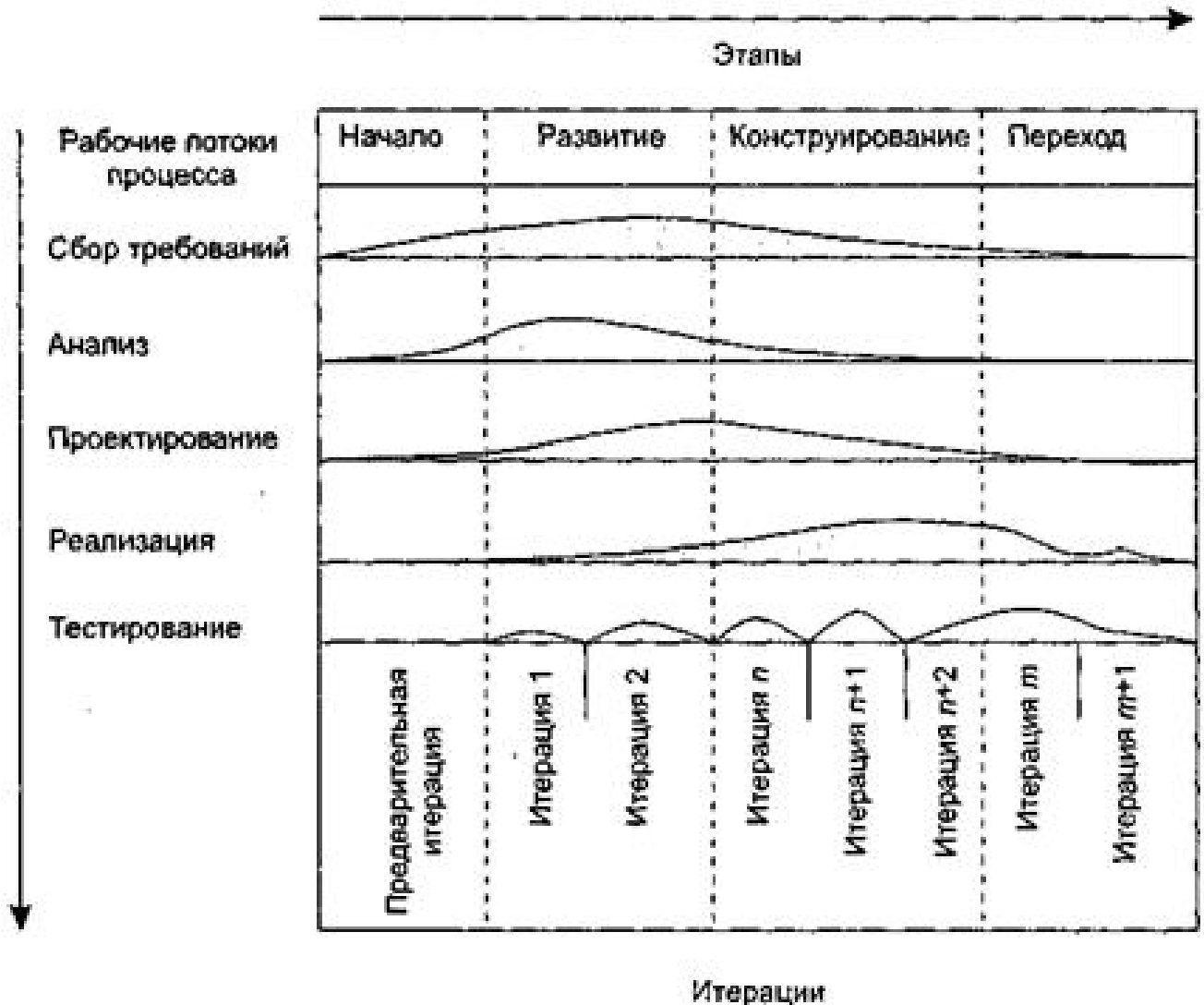
- **Цель:** создать систему

RUP - переход

- **Внедрение (переход):**
  - завершение реализации
  - подготовка АО, ПО (может быть модернизация)
  - создание рекомендаций по установке/ эксплуатации
  - написание документации и руководства
  - проведение В-тестирования, поиск ошибок и их устранение (в версии для приемки)

- **Цель**: применение в среде пользователя

График RUP


Этапы
Рабочие потоки процесса
Начало
Развитие
Конструирование
Переход
Сбор требований
Анализ
Проектирование
Реализация
Тестирование
Предварительная итерация
Итерация 1
Итерация 2
Итерация n
Итерация n+1
Итерация n+2
Итерация m
Итерация m+1
Итерации

Рабочие процессы RUP

- Сбор требований - что делает система;
- Анализ - преобразование требований в классы и объекты предметной области;
- Проектирование - создание статического и динамического представления системы для выполнения требований;
- Реализация - производство программного кода;
- Тестирование - проверка системы в целом.

Любой рабочий процесс определяет набор артефактов и действий.

Рабочий процесс Определение требований

- Перечисление кандидатов в требования
- Осознание контекста системы
- Определение функциональных требований (в виде прецедентов)
- Определение нефункциональных требований

Кандидаты в требования

- Название и краткое описание

- Список:

[название; затраты; приоритет; риск]

- Используют для планирования итераций

Пример кандидатов в требования

- **АСУ «Журнал системных событий» (ЖСС)**

- Отобразить события в таблице;
- Просмотр списка событий;
- Просмотр детальной информации о событии;
- Просмотр всех событий системного журнала;
- Поиск события по дате;
- Поиск события по источнику;
- Поиск события по категории;
- Поиск события по содержимому в названии события;
- Поиск события по содержимому в описании события;
- Выбор пользователем режима работы с XML форматом;
- Импорт данных из XML файла в журнал системных событий;
- Экспорт данных журнала в XML файл;
- Добавление соответствия между тегами XML и полями журнала.
- Экспорт данных из журнала в XML файл с предварительной фильтрацией по дате регистрации события;

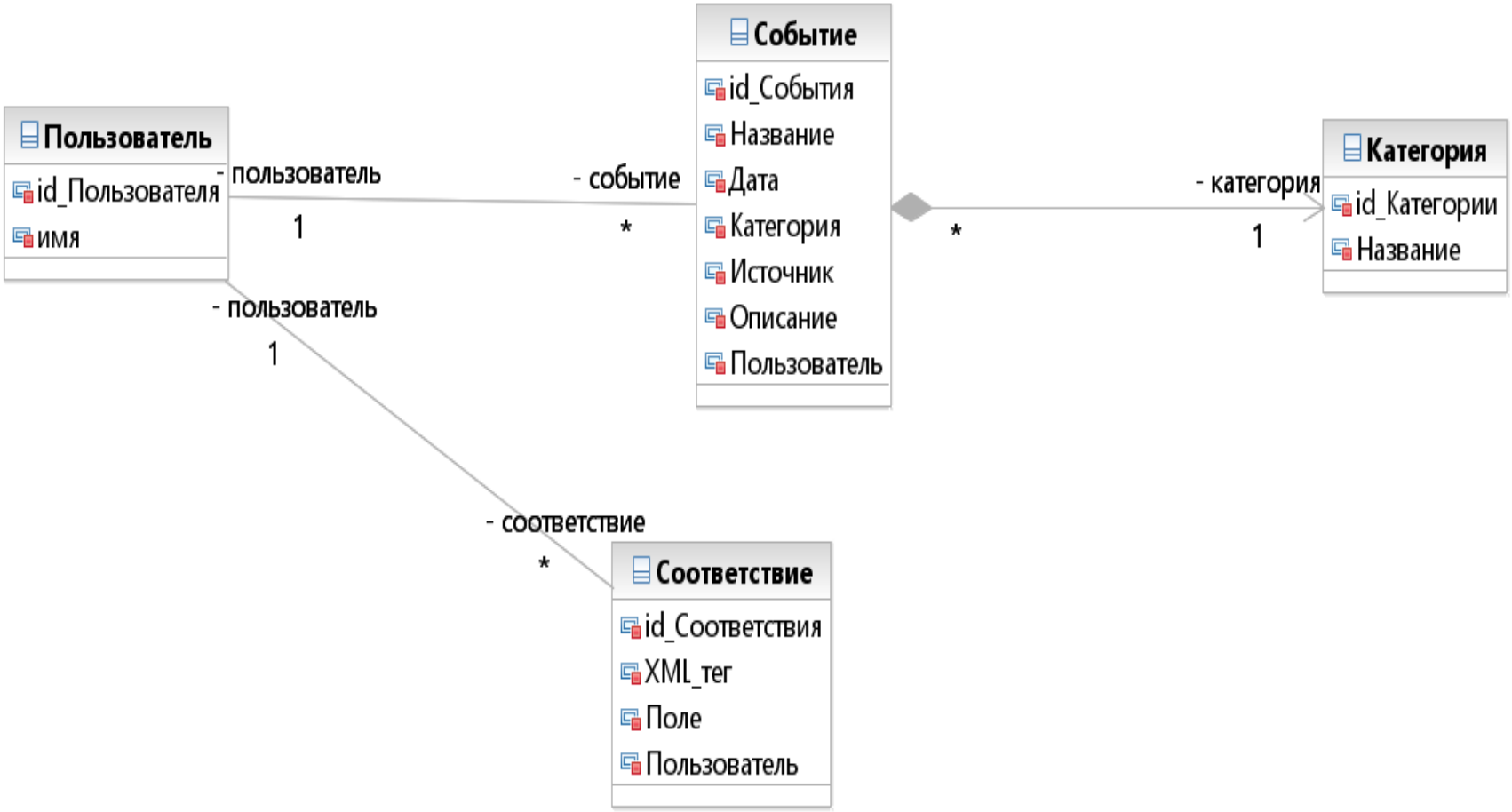
Контекст системы

- Модель предметной области:
  - диаграмма классов предметной области,
  - глоссарий понятий.

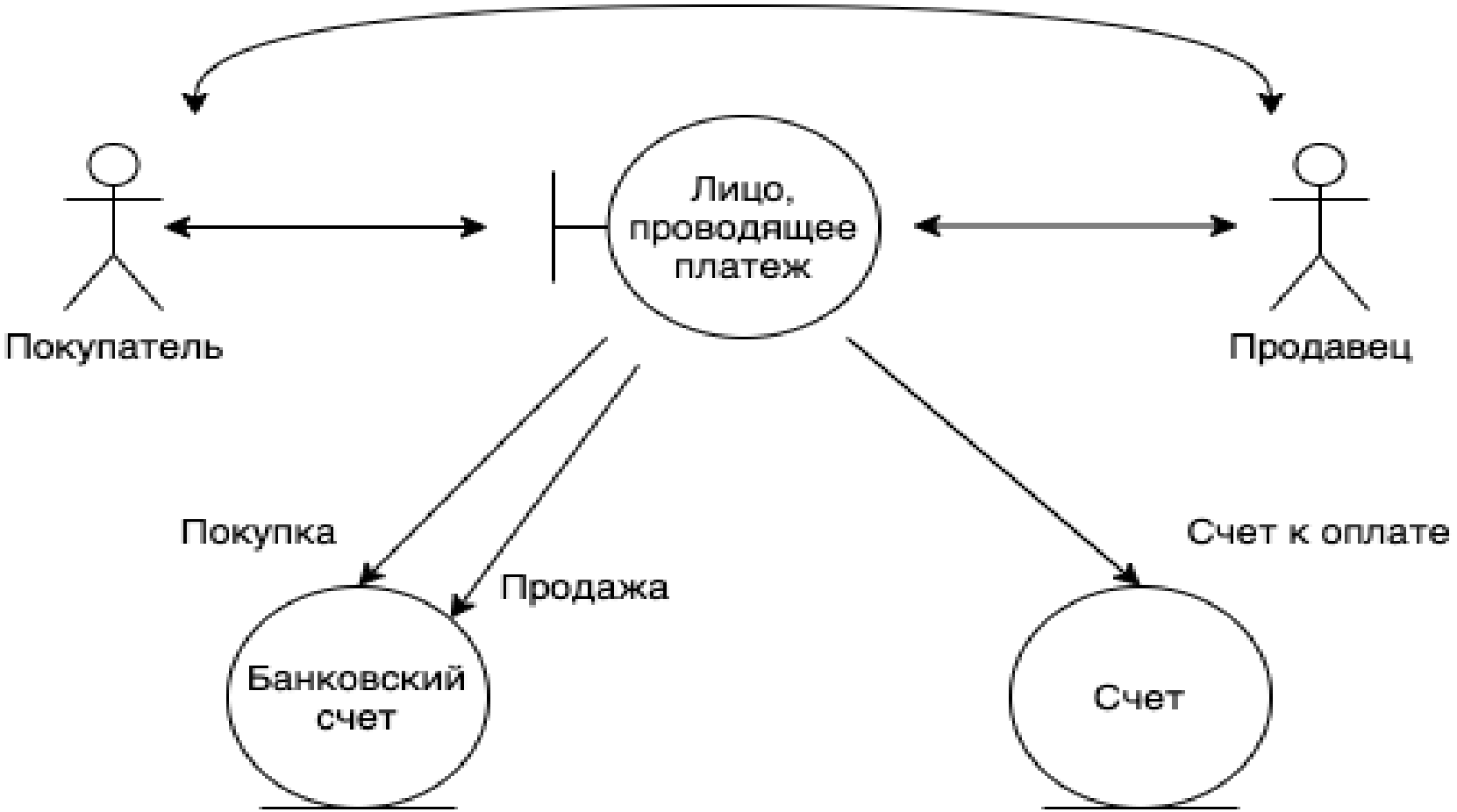
- Бизнес модель

  - Бизнес-модель прецедентов,
  - Объектная бизнес-модель.

Модель предметной области – пример ЖСС



Объектная бизнес-модель – пример АСУ Банка



Определение функциональных требований

- Нахождение актеров и прецедентов
- Определение приоритетов прецедентов
- Детализация вариантов использования (прецедентов)
- Создание прототипа пользовательского интерфейса
- Структурирование модели прецедентов

Нахождение актеров и прецедентов

- Индентификация актеров
  - по 1 на бизнес-сотрудника или бизнес-клиента;
  - по >= 1 на категорию пользователей;
  - по 1 на внешнее устройство и по >= 1 на внешнюю систему.

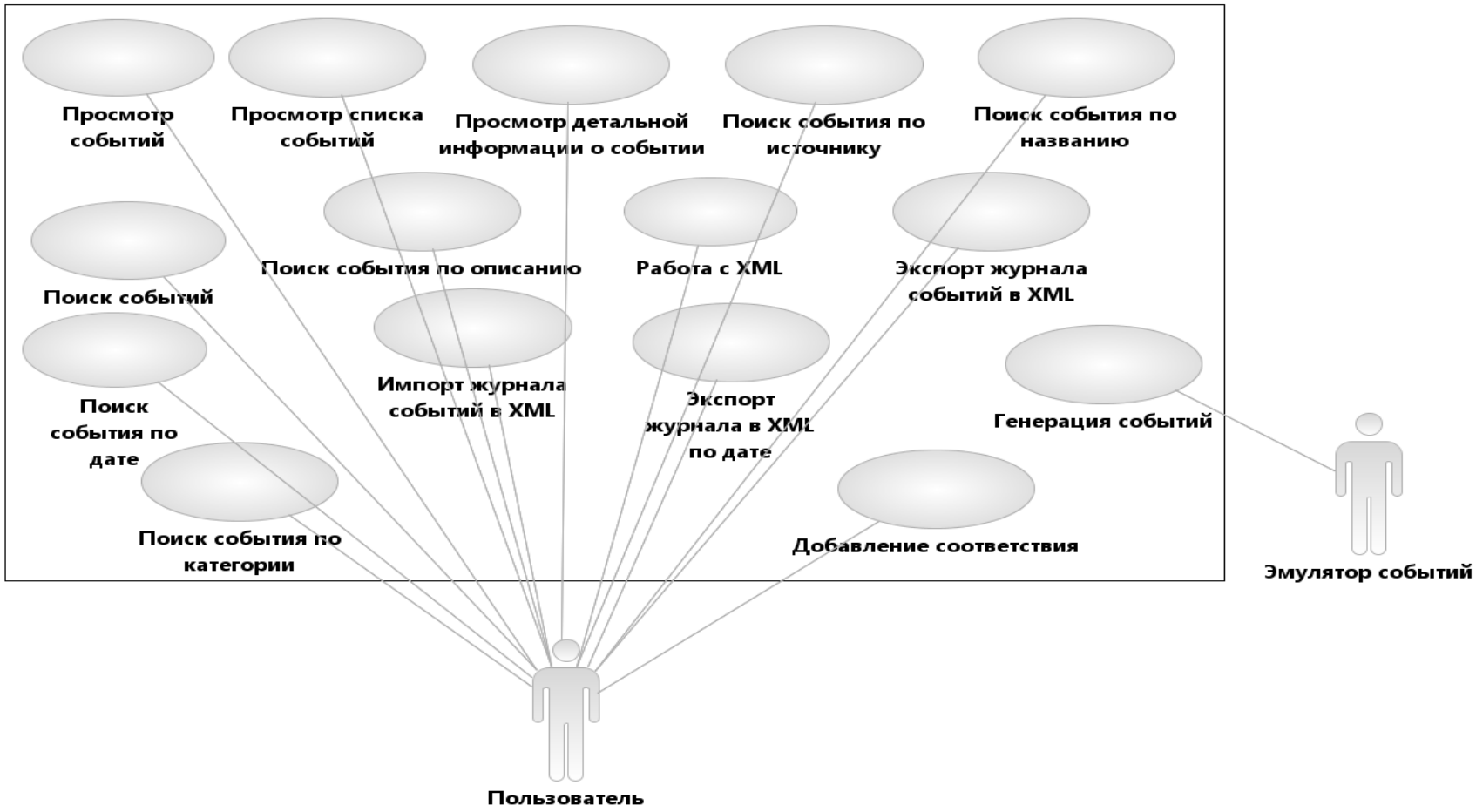
- Роли: по 1 на любой бизнес-процесс для актера.

- Идентификация прецедентов
  - для бизнес-модели: по 1 для любой роли актера;
  - приносит ценный (видимый) результат отдельному актеру;
  - самодостаточный (не является продолжением другого);
  - инициируется актером;
  - взаимодействует только с актером, но не с другими прецедентами.

- Краткое описание прецедентов

- Описание модели прецедентов в целом
  (произвольный набор диаграмм + глоссарий + комментарии)

Модель прецедентов – пример ЖСС



Детализация прецедентов

- Описание прецедента (**спецификация**) содержит:
  - предусловие (начальное состояние);
  - поток событий (запуск; порядок шагов; завершение) - основной путь;
  - постусловие (возможное конечное состояние)
  - описание альтернативных путей в основном;
  - описание альтернативных путей вне основного;
  - сообщения между актерами и прецедентами (последовательность сообщений)
  - атрибуты прецедента (ресурсы, объекты, значения)
- Специальные (нефункциональные) требования к прецеденту (время отклика и т.д.)

- Формализация описания прецедента
  - диаграмма состояния прецедента -> состояния;
  - диаграмма деятельности -> переходы;
  - диаграмма взаимодействия -> между актерами и прецедентами.

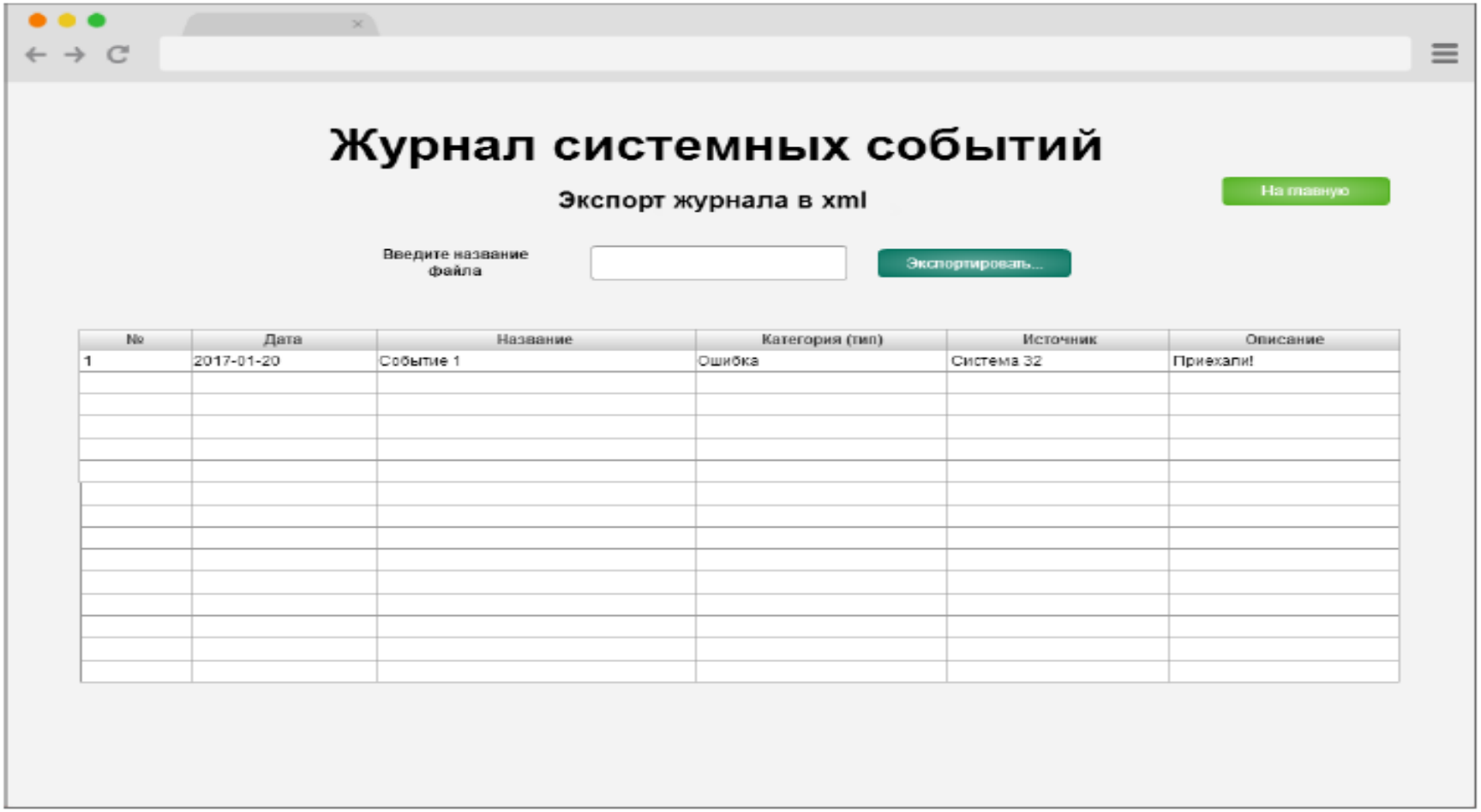
Спецификация прецедента – пример ЖСС

Краткое описание: Предназначен для экспорта журнала событий в документ с расширением XML
Главные актеры: Пользователь
Второстепенные актеры: Нет
Предусловие: 1. Выполнен прецедент Работа с XML
Основной поток: 1. Прецедент начинается после выбора пользователем 2. Система проверяет введенные параметры пользователем 3. Система формирует пользователю документ в формате XML с информацией о событиях из БД 4. Система загружает документ в формате XML пользователю 5. Система отображает документ в формате XML пользователю
Постусловие: Нет
Альтернативные потоки: 1. Прецедент начинается после выбора пользователем 2. Система проверяет введенные параметры пользователем 3. Система выдает ошибку пользователю
Прецедент: Импорт журнала событий из XML

Создание прототипа пользовательского интерфейса

- Создание логического проекта пользовательского интерфейса
  - для любого актера по всем его прецедентам определить:
    - элементы пользовательского интерфейса для прецедента (классы предметной области; бизнес-объекты)
    - связь между элементами;
    - вид элементов и возможности работы с ними;
    - действия и решения актера;
    - информация до прецедента; от актера; к актеру; средние значения параметров (количество в списке и т. д.)
- Создание проекта и прототипа пользовательского интерфейса
  - Набор элементов + дополнительные элементы => эскизы пользовательского интерфейса, наброски экранов.
  - Проверка целостности последовательности действий, достаточности информации, полноты возможностей; отсутствие лишнего при работе актера с прецедентом

Прототип ПИ – пример ЖСС



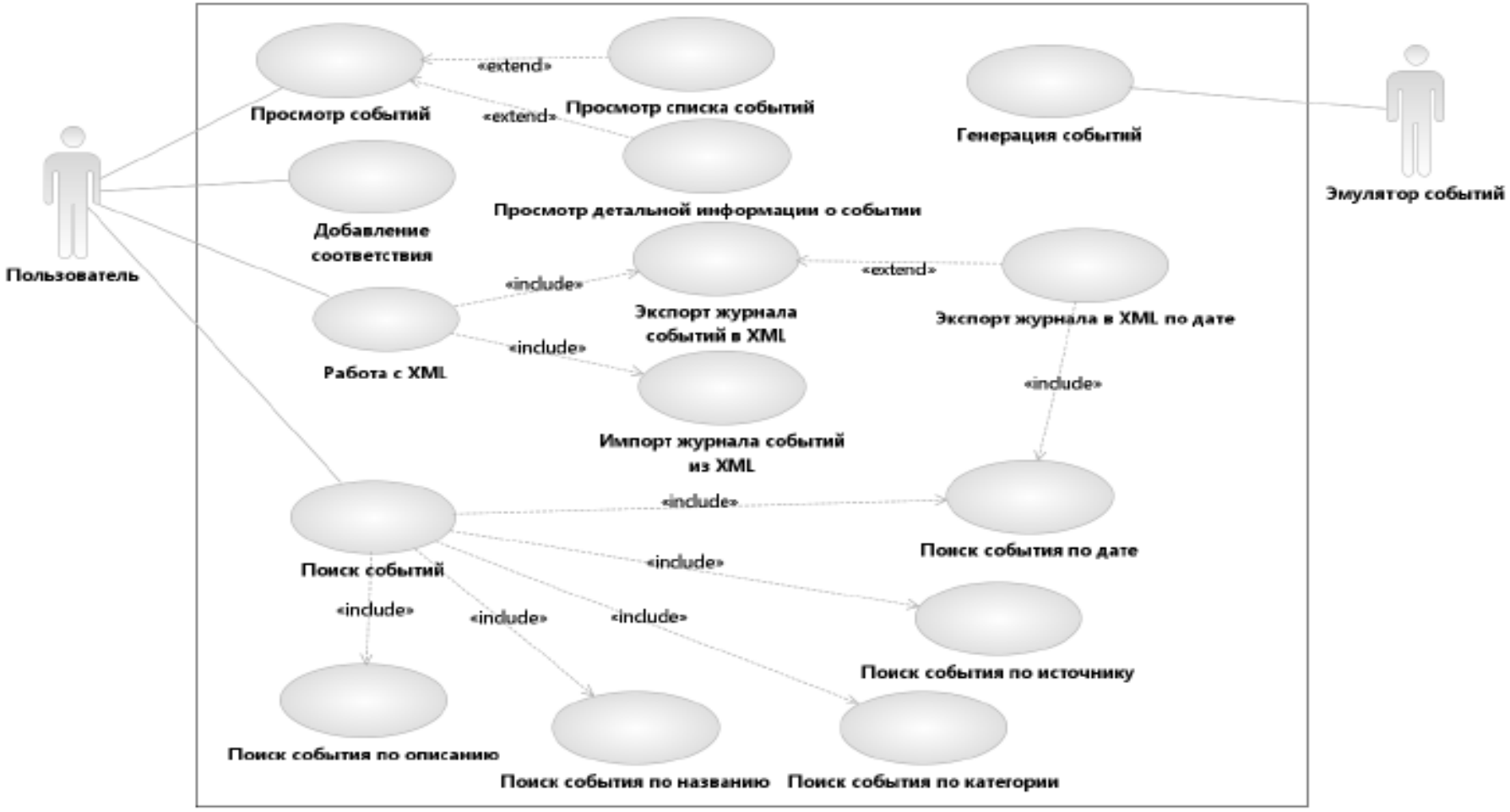
Структурирование модели прецедентов

- Определяются общие и совместно используемые описания функциональности в прецедентах - выделить их через обобщение

- Определить дополнительные и необязательные описания функциональности в прецедентах - выделить их через расширение (точка расширения и условия расширения) <<extend>>
- Определение других отношений между прецедентами: <<include>> - включение закрытой функциональности, без доступа к его атрибутам.

Пример структурирования модели прецедентов

Модель прецедентов – пример ЖСС



Нефункциональные требования

- ограничения среды и реализации;
- производительность (скорость, пропускная способность, время отключения, ОП);
- зависимость от платформы;
- расширяемость;
  надежность (точность, средняя наработка на отказ) и т.д.;
- требования к пользовательскому интерефейсу;
  форматы, протоколы;
- внешние компоненты / подсистемы;
- безопасность;
  легкость обучения (при использовании)

- Не связаны с конкретным прецедентом, относятся к нескольким или ни к одному

Нефункциональные требования - пример ЖСС

- Допустимое количество сбоев в портале должно быть не более 1 сбоя в квартал;
- Среднее время восстановления после сбоя должно составлять не более 1,5 часа;
- Дизайн должен быть лаконичен и понятен любому пользователю;
- Комплекс должен иметь минимальное время обработки запросов (не более 3 секунд на запрос);

Рабочий процесс Анализ (требований)

- Цель:
  - уточнить спецификацию требований;
  - перейти от языка заказчика к языку разработчика, анализ внутренних механизмов системы;
  - структурирование требований для дальнейшей разработки ПО

- Шаги:
  - Анализ архитектуры
  - Анализ кооперации
  - Анализ классов
  - Анализ пакетов

Анализ архитектуры

- Идентификация пакетов анализа
- Идентификация сервисных пакетов
- Определение зависимости между пакетами анализа
- Определение очевидных классов сущностей
- Определение общих специальных требований

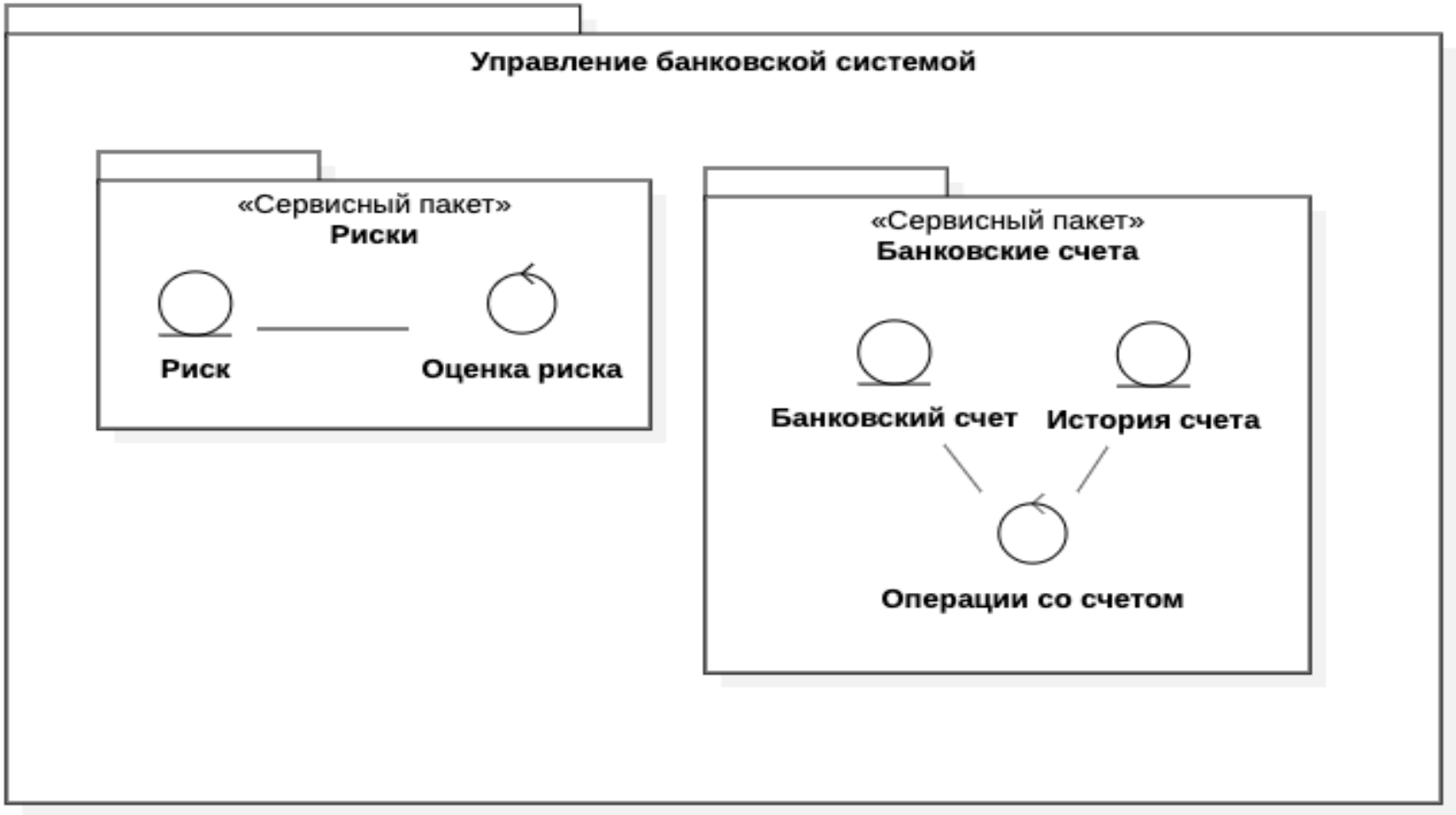
Пакеты анализа

- Пакет анализа:
  - сильная связность и слабое сцепление
  - на основе функциональных требований бизнеса, множество прецедентов для 1 актера / бизнес-процесса / обобщения и расширения.
  - для 2-х верхних уровней приложения.

- Если >= 1 пакета используют общий класс анализа, то класс надо вынести в отдельный пакет анализа или просто отдельно

- Сервисный пакет:
  - набор функциональности для клиента;
  - содержит множество функционально связных классов;
  - неделим при приобретении;
  - может повторно использоваться;
  - взаимозаменяем.

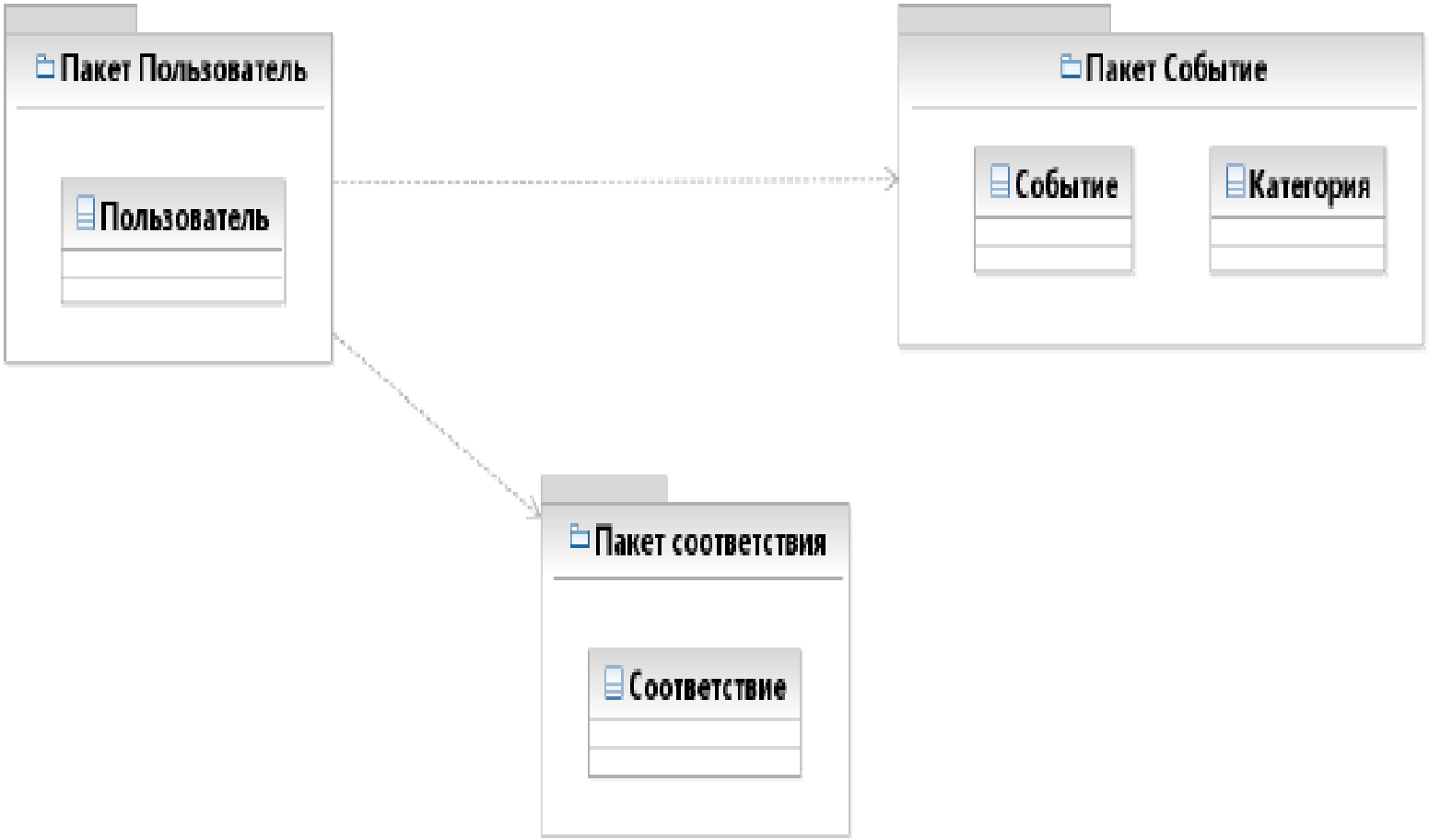
Анализ архитектуры - пример



Зависисмости между пакетами анализа

- связи зависимости;
- высокая связность и низкое сцепление;
- деление на уровни.

Пакеты анализа – пример ЖСС



Определение общих специальных требований

- хранение данных (скорость, частота, ...);
- распределение и распараллеливание разработки;
- безопасность;
- устойчивость к сбоям;
- упрвление транзакциями.

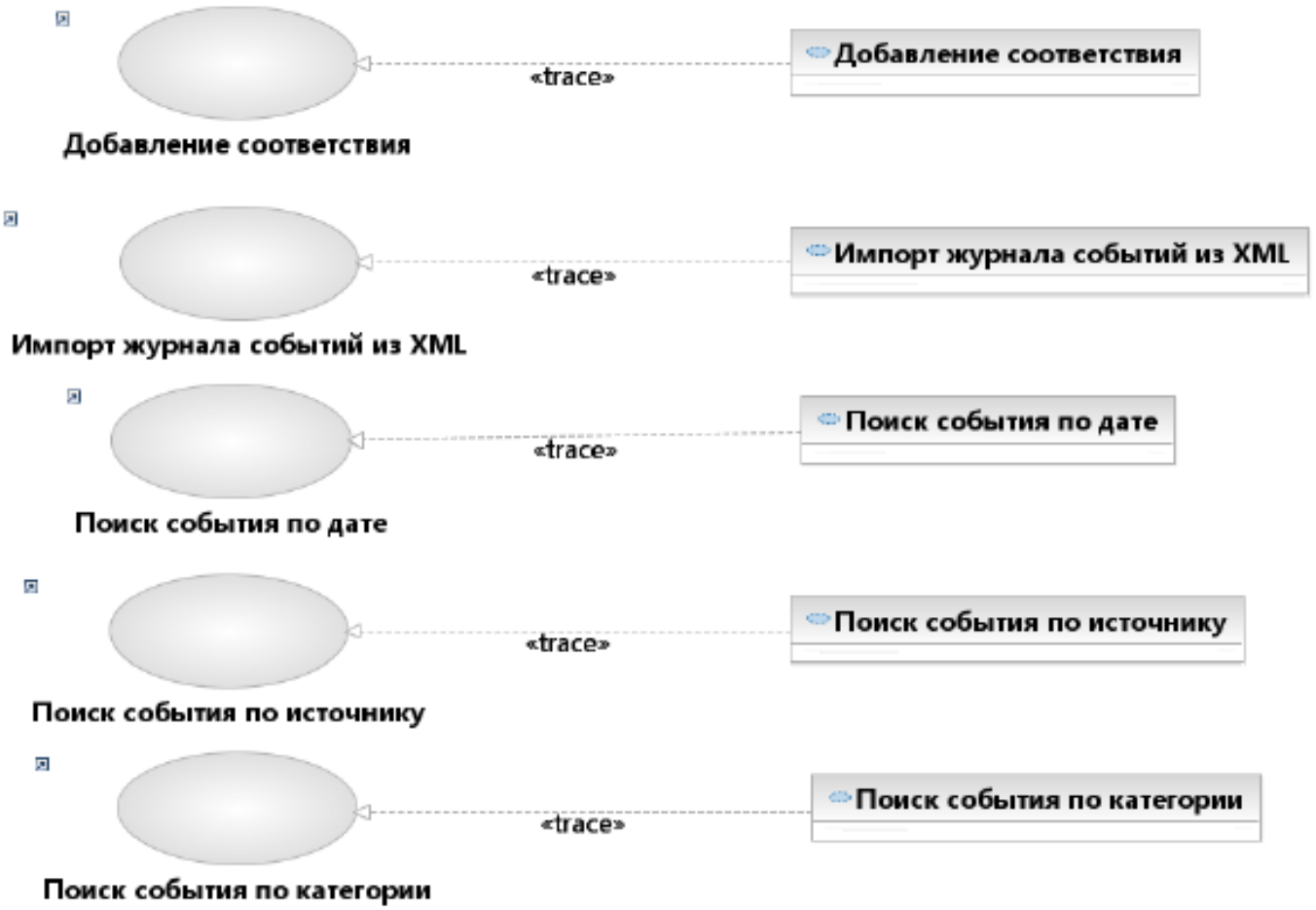
  – Влияют на проектирование и реализацию, характеристики требований приписаны к любому классу или кооперации, на которую ссылаются.

Анализ кооперации

- Кооперация – реализация прецедента
  - Статика – диаграмма классов-участников
  - Динамика – диаграмма взаимодействия или последовательностей

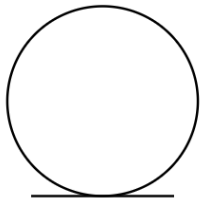
- Трассировка кооперации от прецедентов
- Определение классов анализа
- Описание взаимодействия объектов анализа
- Определение специальных требований

Трассировка кооперацій от прецедентов – пример ЖСС

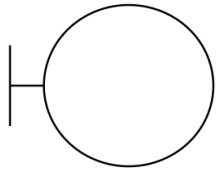


Классы анализа

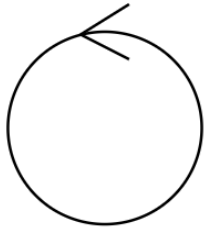
- Класс сущности - для моделирования долгоживущей системы, часто сохраняемой информацию о человеке / объекте / событии реального мира. Может иметь сложное поведение. Показывает логическую структуру данных.



- Граничный класс - для моделирования взаимодействия между системами и актерами:получение / передача информации / запросов. Соответствуют пользовательскому интерфейсу / устройству/ коммуникации / API на внешнем уровне без реализации.



- Управляющий класс - координация / последовательность /взаимодействие / управление другими объектами для прецедента. Обрабатывают и координируют действия и потоки управления; реализуют бизнес-логику.



Определение классов анализа

- определение классов сущностей (10 - 20 штук) на основе модели предметной области или бизнес-объектов, которые участвуют в прецедентах.
  Для них определяют атрибуты и ассоциации.

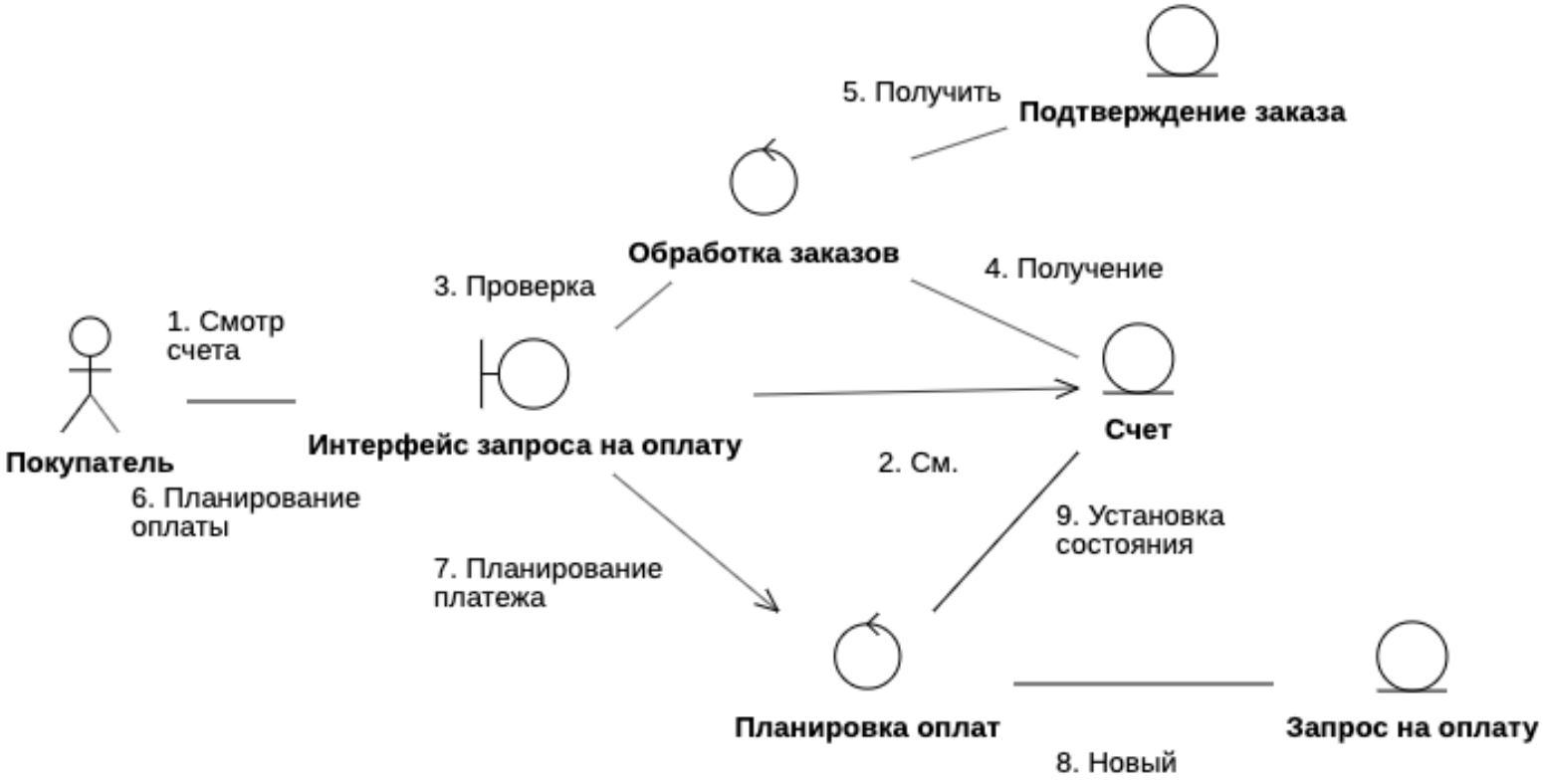
- определение по одному основному граничному классу на любого актера человека - это главная форма его интерфейса,
- определение по одному основному граничному классу на любую внешннюю систему - это коммуникационный интерфейс. Если N уровней коммуникации, то по одному граничному классу на уровень.

- Определение по одному управляющему классу на кооперацию (может быть управляющий класс в граничном или больше двух управляющих классов для сложных кооперациий)

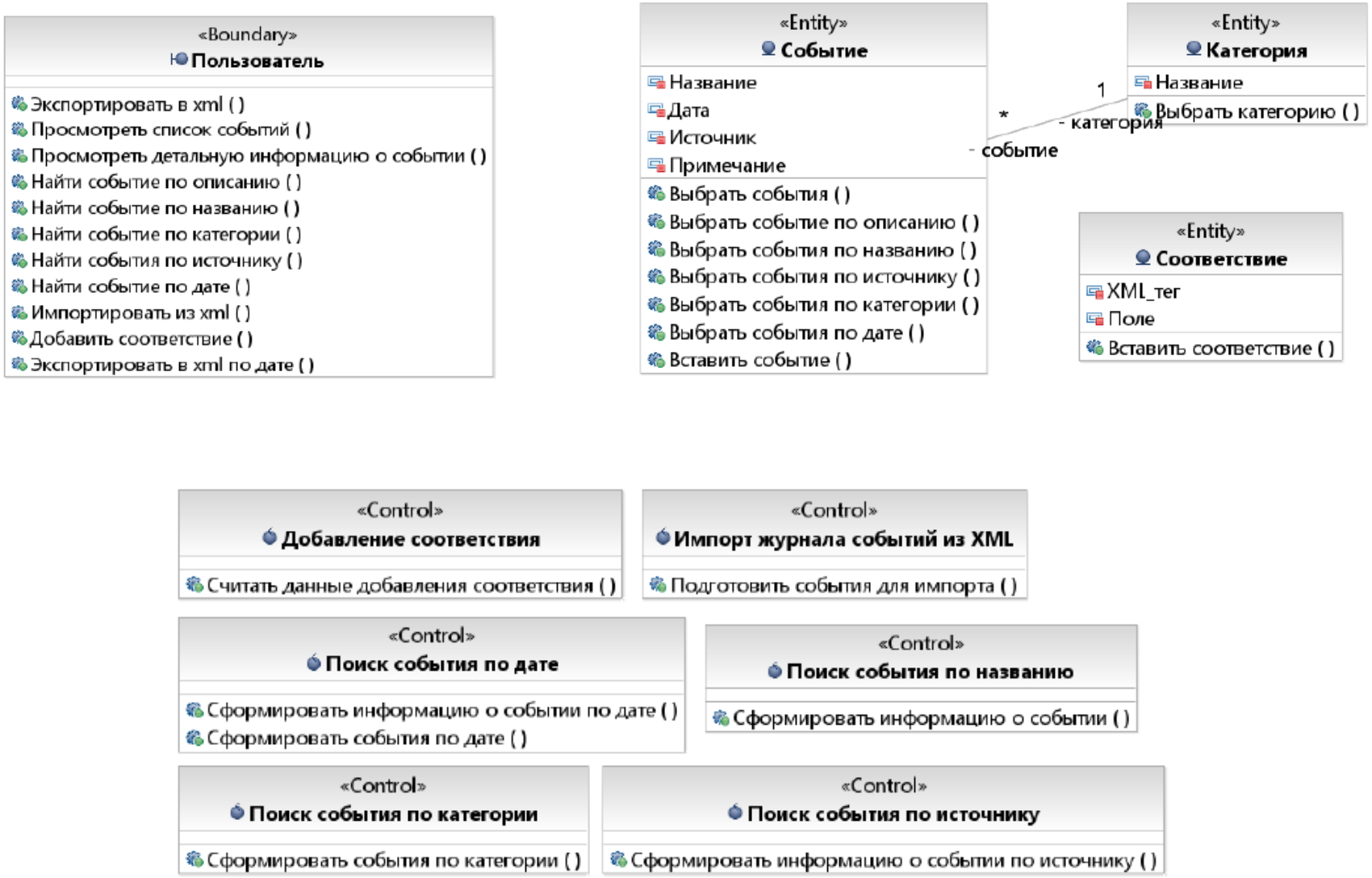
Описание взаимодействия объектов анализа

- Построение диаграммы кооперации для прецедента.
- Если существует N потоков или подпотоков, то по 1 диаграмме кооперацией на поток.
- Выбрать участников кооперации: актер(ы), граничный – управляющий – сущность(и).
- Кооперация создается событием от актера.
- Направленные ассоциации.

Диаграмма взаимодействия - пример



Классы анализа – пример ЖСС



Кооперация – пример ЖСС

Импорт журнала событий из XML

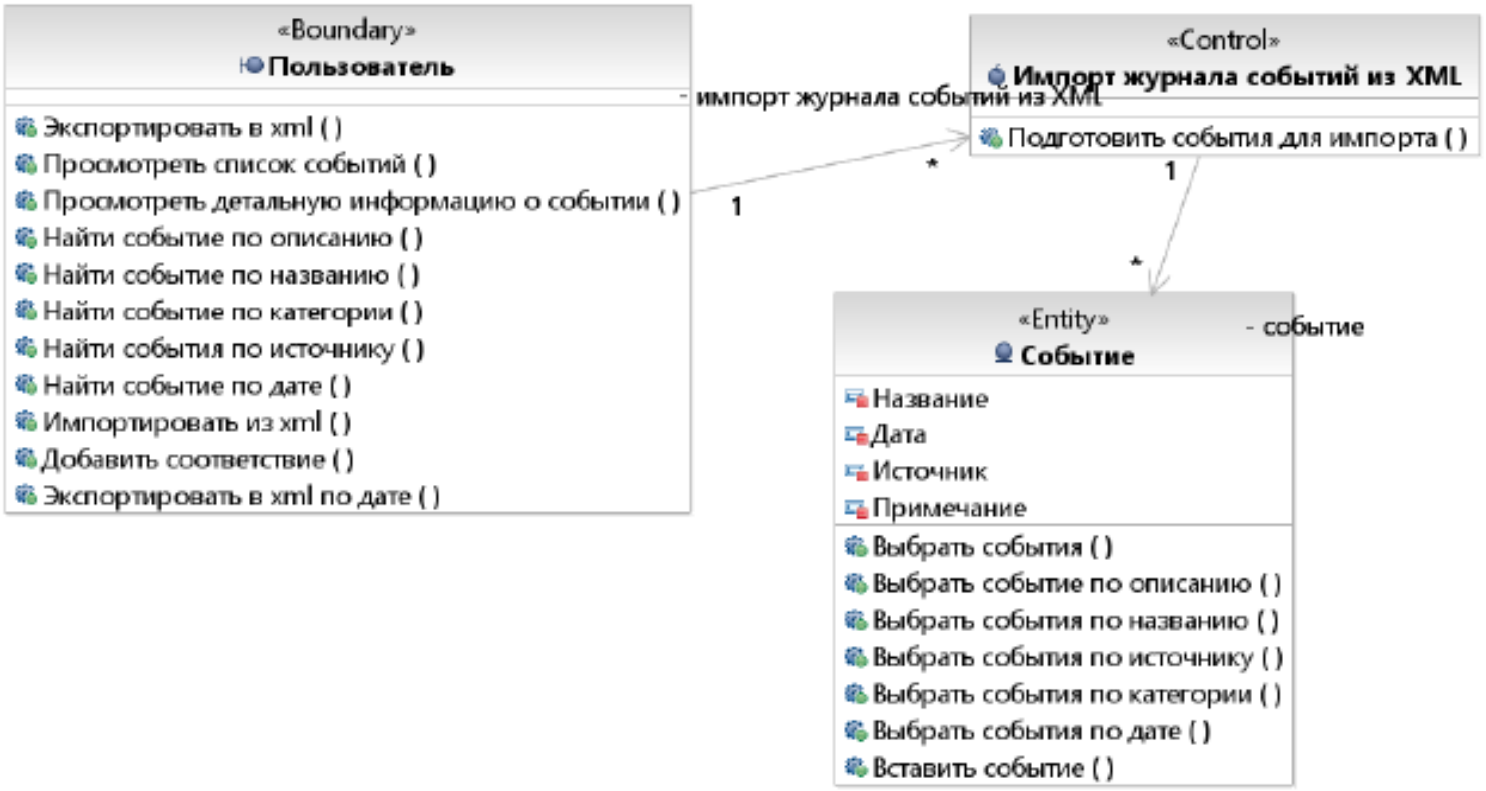
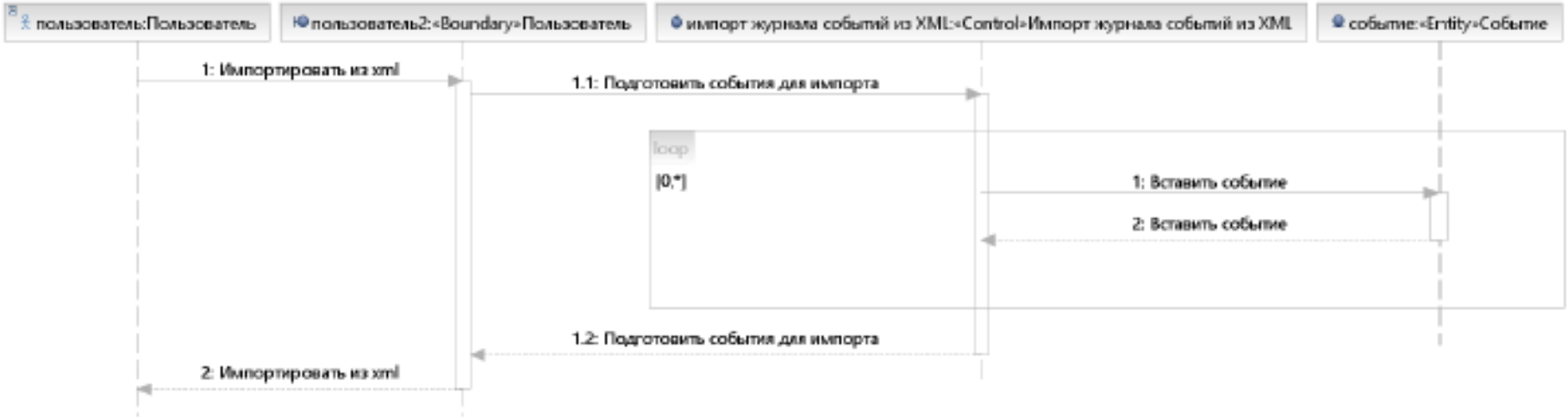


Диаграмма последовательностей кооперации – пример ЖСС



Анализ классов

1. Определение ответственности - из комбинации всех ролей класса во всех прецедентах
2. Определение атрибутов
   - концептуальные типы
   - если сложные и много атрибутов, то часть атрибутов выносят в отдельный класс
   - общие атрибуты у N классов выносят в отдельный класс

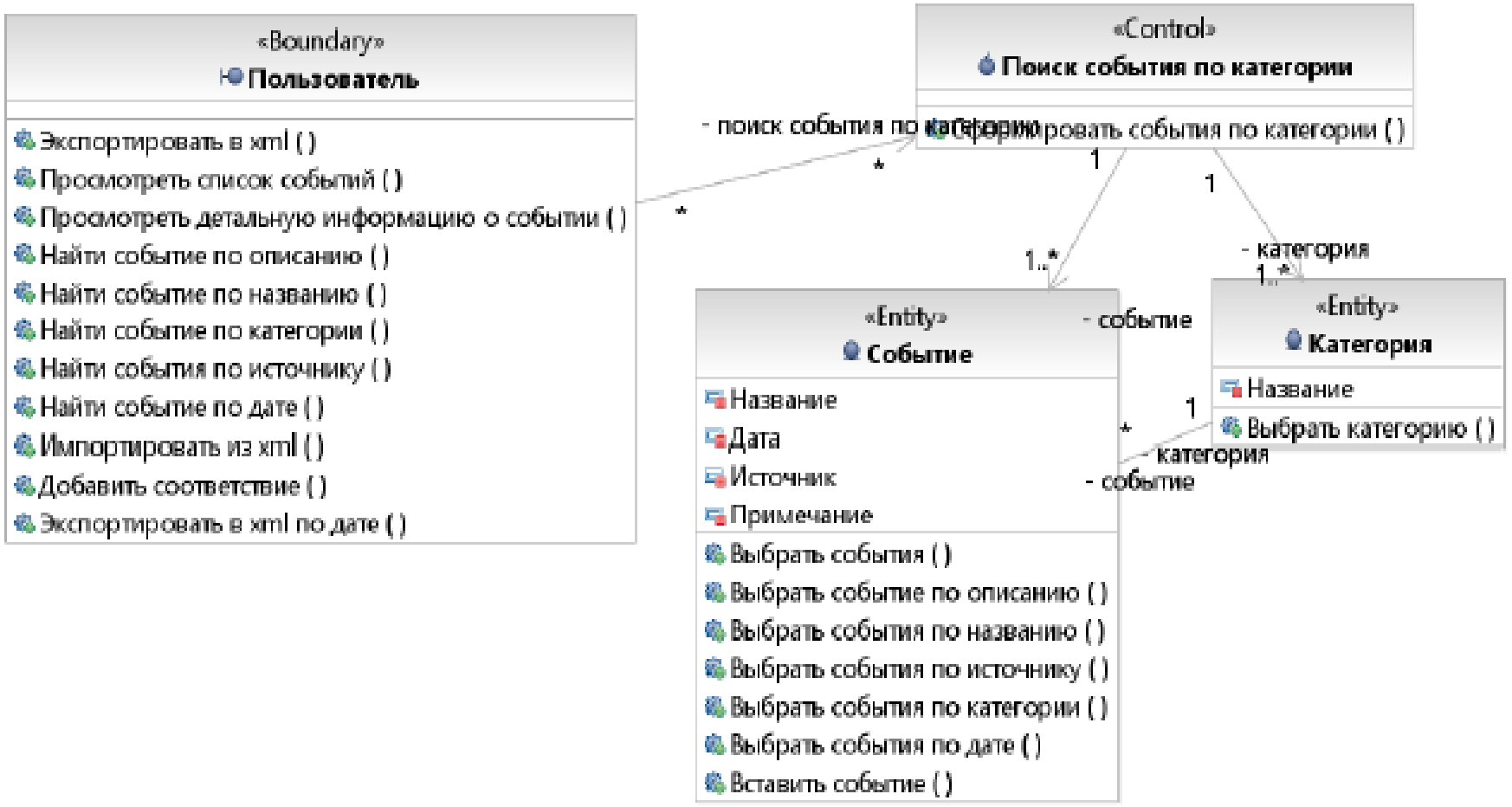
Атрибуты классов:

Классы сущностей	Граничные классы	Управляющие классы
- просты; - от предметной области; - полезные исходные данные	- Свойства коммуникационных протоколов - поля ввода	- редки; - аккумулируют / порождают значения для прецедента

Анализ классов

3. Определение ассоциаций и агрегаций
   - для указания взаимодействия объектов в кооперации
   - определяются: множество ассоциаций; роли; арность; класс ассоциаций;
   - определяют агрегацию, если: - физическая вложенность (авто и детали); - коллекция (родитель и дети).
4. Определение обобщений
   - Получить разделяемое или общее поведение классов.
5. Определение специальных требований

Анализ классов – пример ЖСС

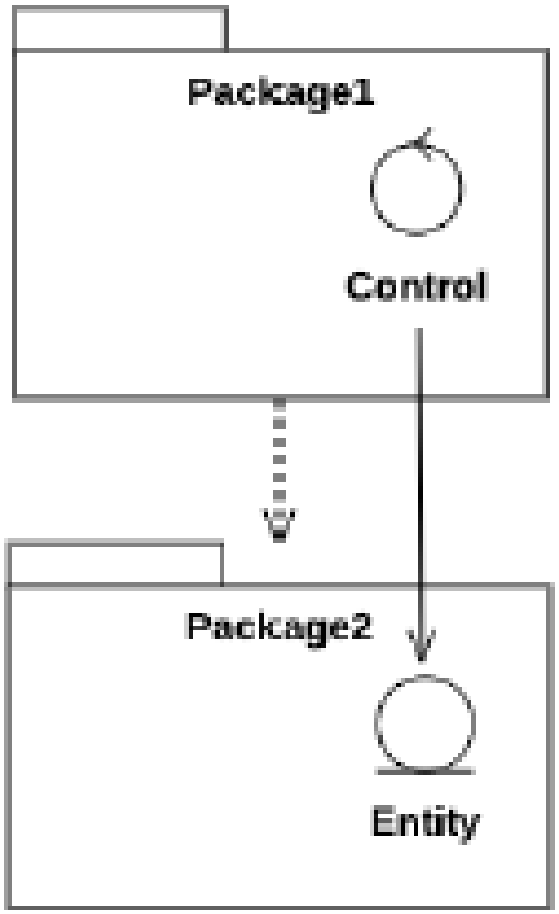


Анилиз пакетов

- Цель:
  - независимость пакета от других;
  - реализация пакетом его прецедентов / классов предметной области;
  - определить зависимости пакетов (оценка внесенных изменений);
  - пакет содержит функционально ориентированные классы.

- Возможно перемещение классов

Анализ пакетов - пример



Анализ пакетов – пример ЖСС