Программная инженерия

Описание требований на языке UML

Вопросы

- Обсуждение курсовой работы
- Описание требований на языке UML:
  - Иерархия требований
  - Use Case и ссылки на требования
  - Диаграмма последовательности действий
  - Диаграмма состояний
- Обсуждение курсовой работы

Иерархия

Use Cases

Диаграммы последовательности и структуры

ОПИСАНИЕ ТРЕБОВАНИЙ НА ЯЗЫКЕ UML

Стандарты

- OMG Unified Modeling Language (OMG UML), Infrastructure Version 2.4.1
  - OMG Document Number: formal/2011-08-05
  - ISO/IEC 19505-1
- OMG Unified Modeling Language (OMG UML), Superstructure Version 2.4.1
  - OMG Document Number: formal/2011-08-06
  - ISO/IEC 19505-2

UML – 1/5

- UML – язык графического описания для объектного моделирования в области разработки ПО:
  - Определение
  - Визуализация
  - Проектирование
  - Документирование
- С его помощью создают абстрактную модель системы (UML-модель)

UML – 2/5

- Был создан, в основном, для программных систем, но используется шире
- Не является языком программирования, но на основании UML-моделей возможна генерация кода (и обратно – реверс-инжиниринг)
- Неотъемлемая часть идеологии ООА и ООП
- Прочно связан с CASE-системами

UML – 3/5

- Structure Diagrams:
  - Class diagram
  - Component diagram
  - Composite structure diagram
  - Deployment diagram
  - Object diagram
  - Package diagram
  - Profile diagram
- Структурные диаграммы:
  - Диаграмма классов
  - Диаграмма компонентов
  - Диаграмма композитной/составной структуры
  - Диаграмма развёртывания
  - Диаграмма объектов
  - Диаграмма пакетов
  - Диаграмма профилей

UML – 4/5

- Behaviour Diagrams:
  - Activity diagram
  - State Machine diagram
  - Use case diagram
  - Interaction Diagrams:
    - Communication diagram
    - Interaction overview diagram
    - Sequence diagram
    - Timing diagram

- Диаграммы поведения:
  - Диаграмма деятельности
  - Диаграмма состояний
  - Диаграмма вариантов использования
  - Диаграммы взаимодействия:
    - Диаграмма коммуникации
    - Диаграмма обзора взаимодействия
    - Диаграмма последовательности
    - Диаграмма синхронизации

UML – 5/5

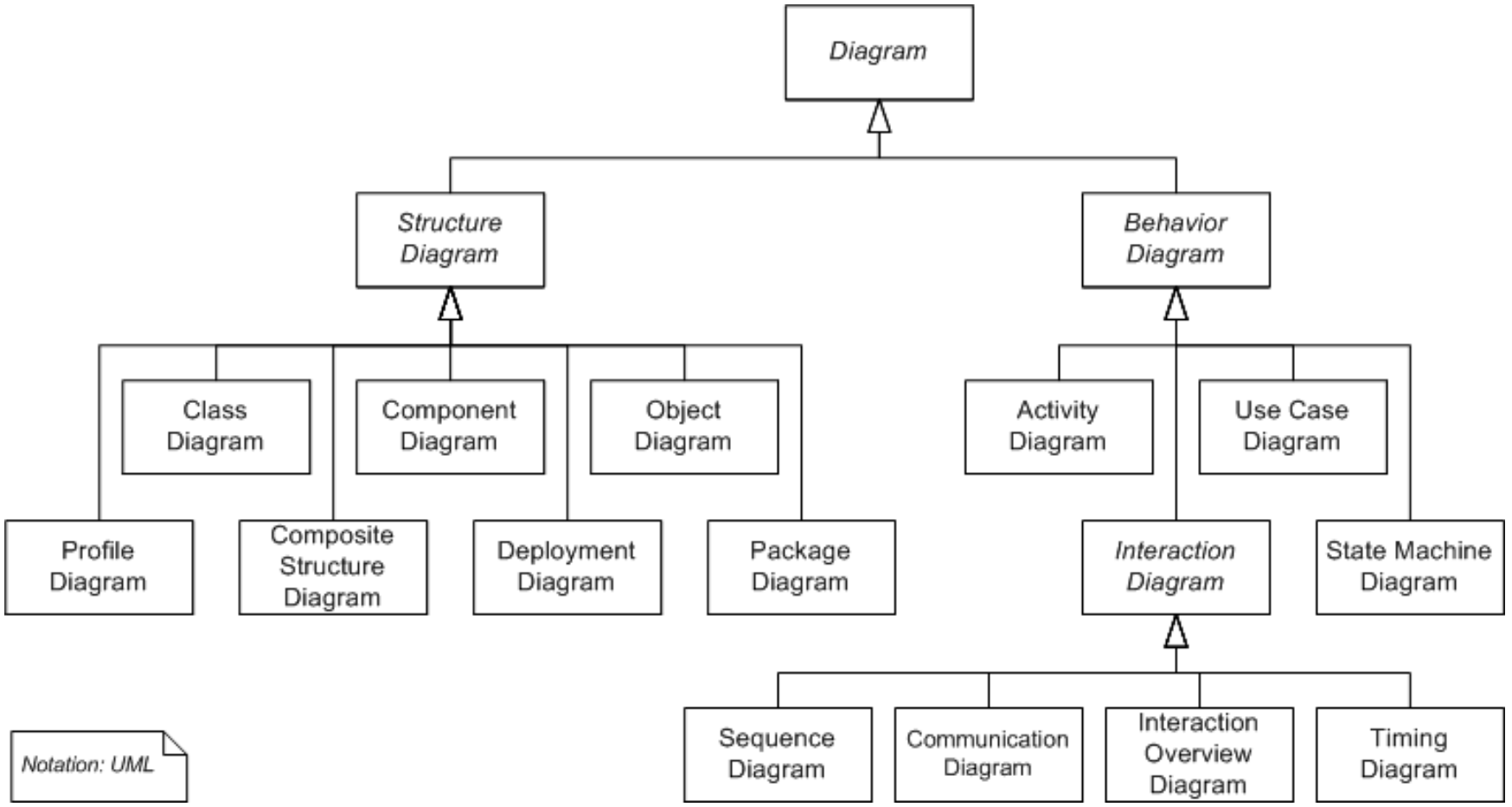
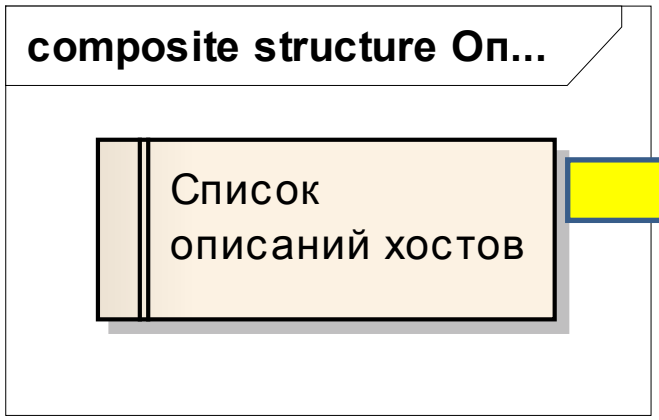
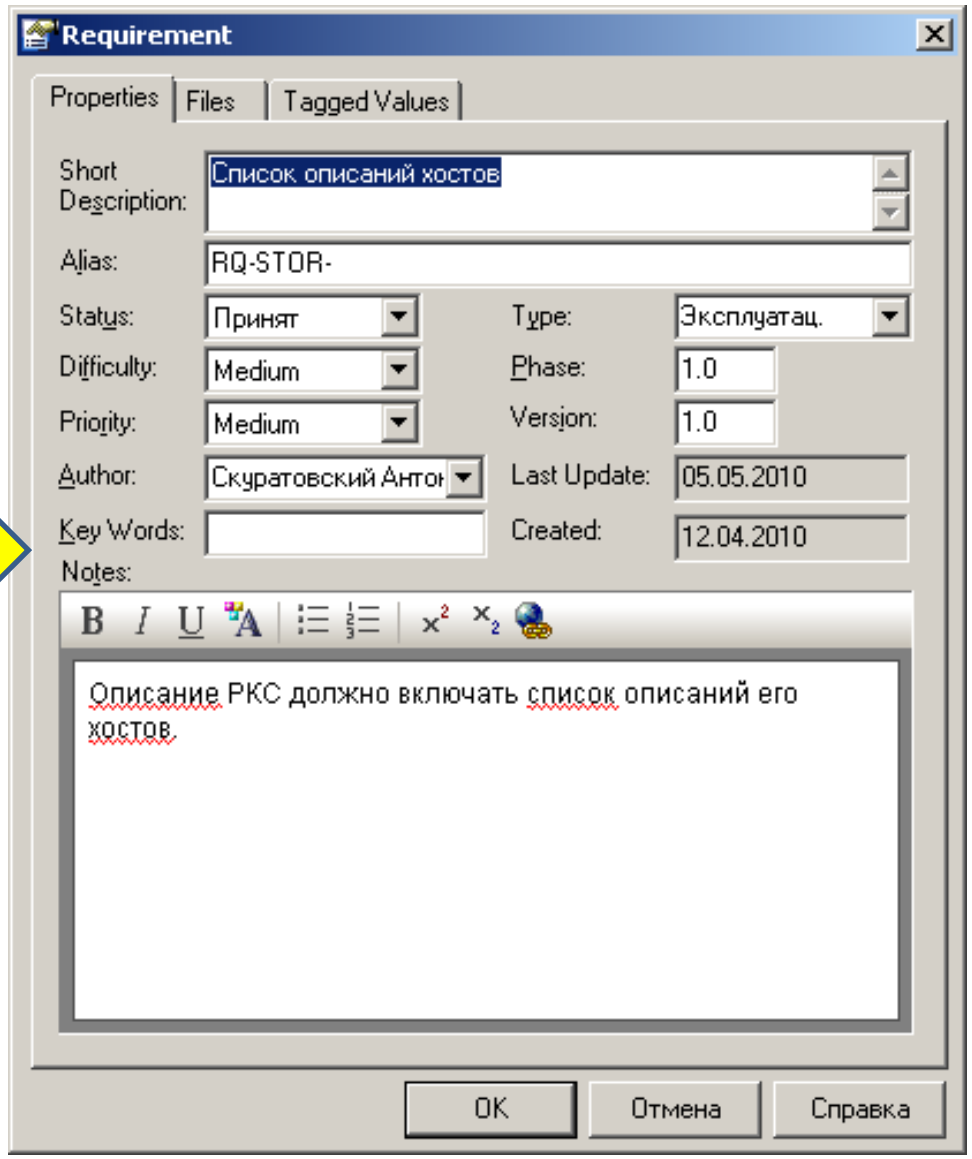


Diagram
Structure Diagram
Behavior Diagram
Class Diagram
Component Diagram
Object Diagram
Activity Diagram
Use Case Diagram
Profile Diagram
Composite Structure Diagram
Deployment Diagram
Package Diagram
Interaction Diagram
State Machine Diagram
Sequence Diagram
Communication Diagram
Interaction Overview Diagram
Timing Diagram
Notation: UML

Требование в UML – 1/2





Требование в UML – 2/2

- Краткое наименование
- Текстовое описание (формулировка)
- Тип
- Статус / фаза / версия
- Автор
- Сложность
- Приоритет

Требование в UML: надо ли?

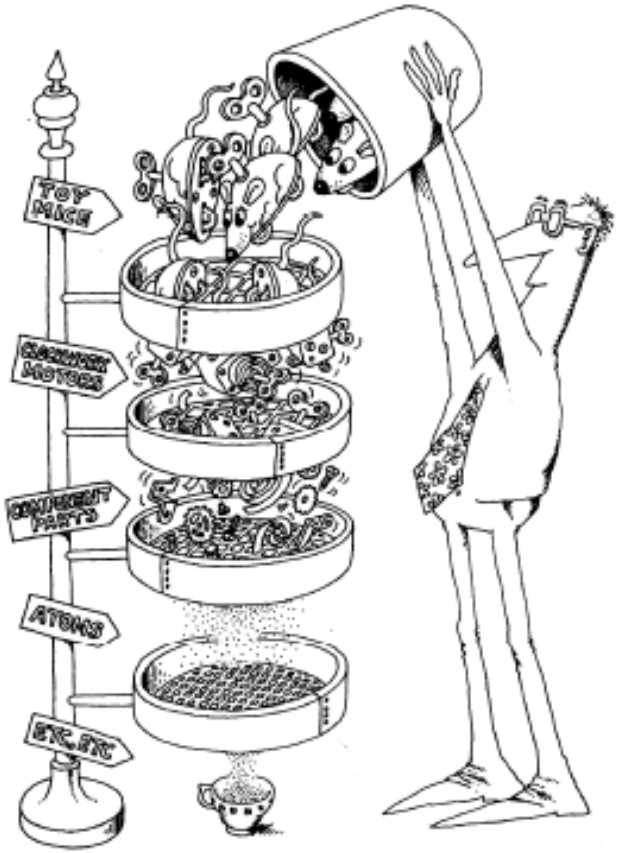
- При наличии линейного списка текстовых требований UML диаграмма для требований не имеет смысла
- Однако использование CASE-средств смысл имеет, так как:
  - Требования, описанные в том же формате, что и все остальные данные проекта облегчают трассировку
  - Единое место / формат для хранения проектной документации
  - Автоматическая генерация документации же!

Немного об ОО – 1/2

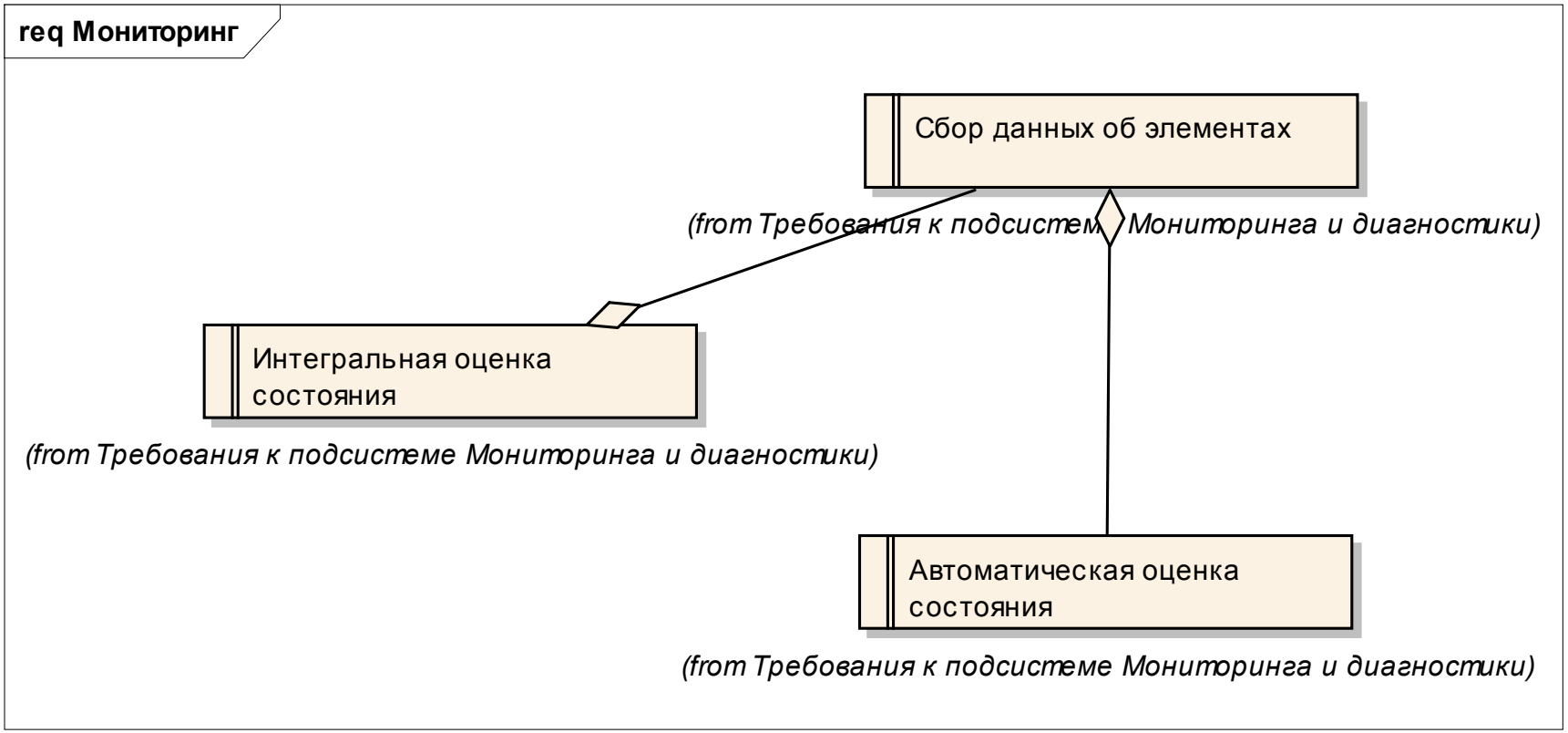
- ~~Не путать с О-о-о-о!~~
- Основными принципами построения объектной модели являются:
  - *Абстрагирование*
  - *Инкапсуляция*
  - *Модульность*
  - **Иерархия**
- Дополнительные принципы:
  - *Типизация*
  - *Параллелизм*
  - *Устойчивость (persistence)*

Немного об ОО: иерархия – 2/2

- Иерархия – ранжированная или упорядоченная система абстракций, расположение их по уровням в виде древовидной структуры
- Иерархия классов строится по наследованию, а **иерархия объектов – по агрегации**



Немного об ОО: Иерархия – 3/3



- Иерархия требований подразумевает агрегацию, то есть включение одного требования в другое (включенные требования полнее раскрывают требования верхнего уровня)

Use Case – 1/7

- Диаграмма вариантов использования (диаграмма прецедентов) отражает отношения, существующие между акторами и вариантами использования
- Используется как единое средство, дающее возможность заказчику, конечному пользователю и разработчику совместно обсуждать функциональность и поведение системы

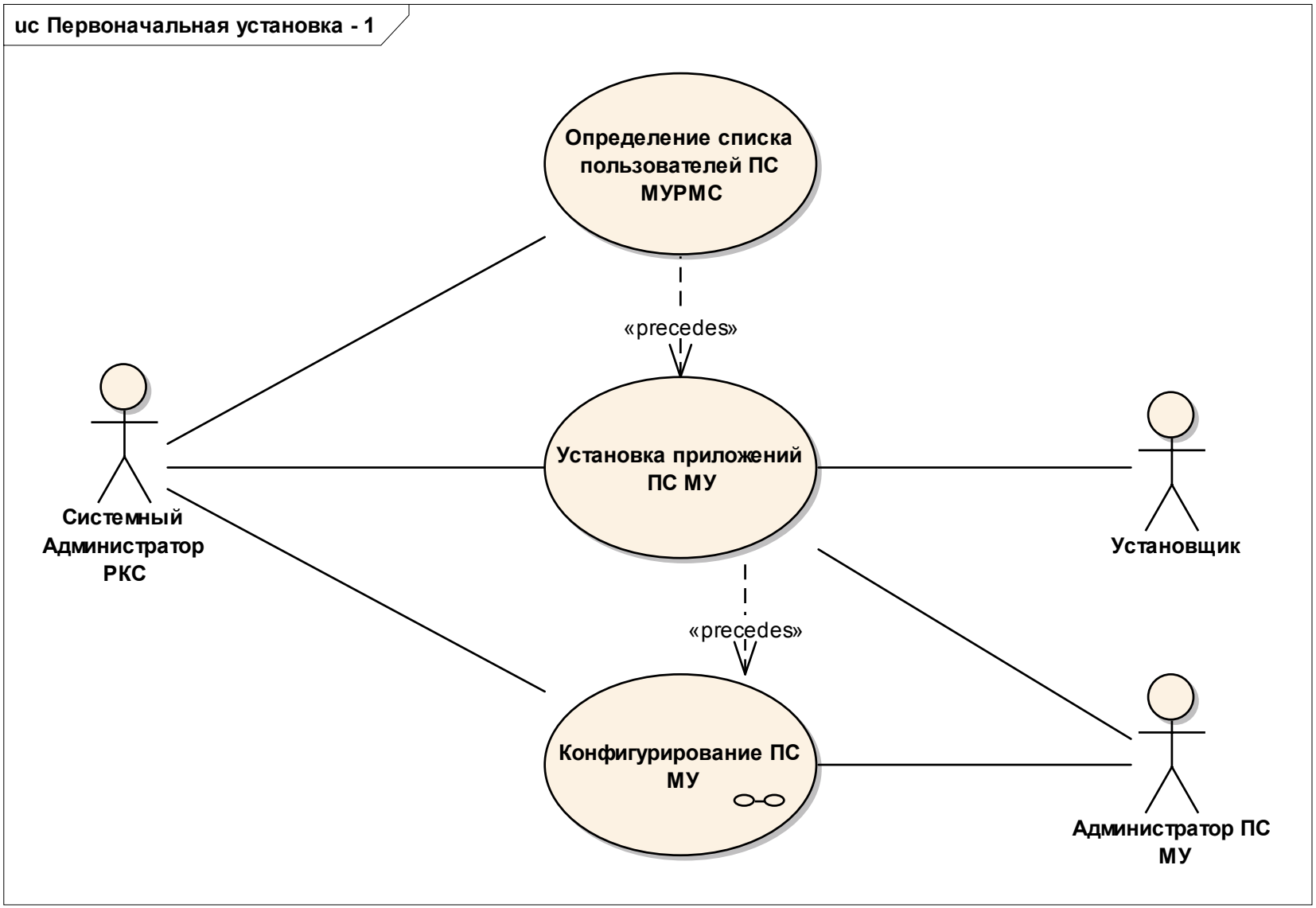
Use Case – 2/7

- Вариант использования (прецедент) – описание последовательности действий, которые может осуществлять система, подсистема или класс, взаимодействуя с внешними акторами
  - Могут описываться как линейные последовательности, так и варианты последовательностей и ошибочные последовательности
- Документируют функциональные требования

Use Case – 3/7

- Могут описывать поведение системы иерархически, с разной степенью детализации
- Акторы:
  - Люди
  - Системы
  - Процессы

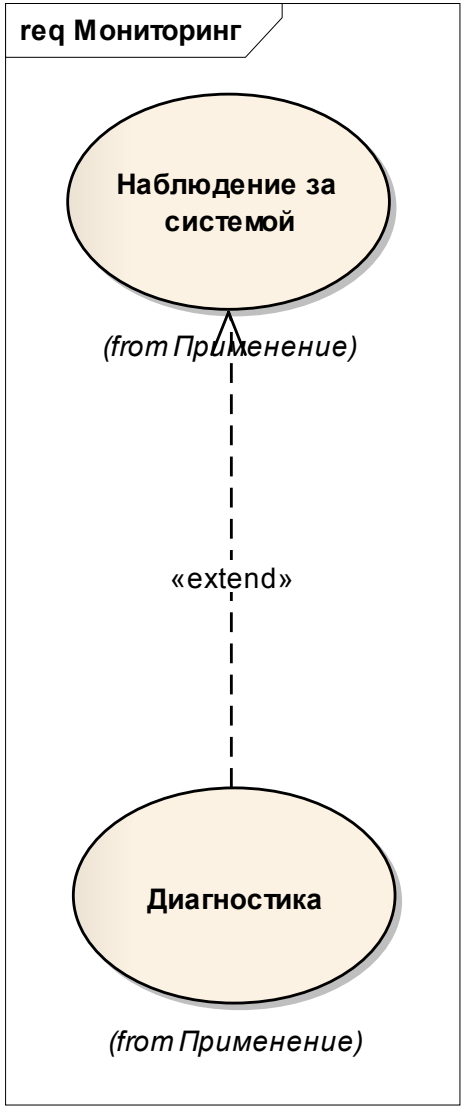
Use Case – 4/7


uc Первоначальная установка - 1
Определение списка пользователей ПС МУРМС
«precedes»
Установка приложений ПС МУ
«precedes»
Конфигурирование ПС МУ
Системный Администратор РКС
Установщик
Администратор ПС МУ

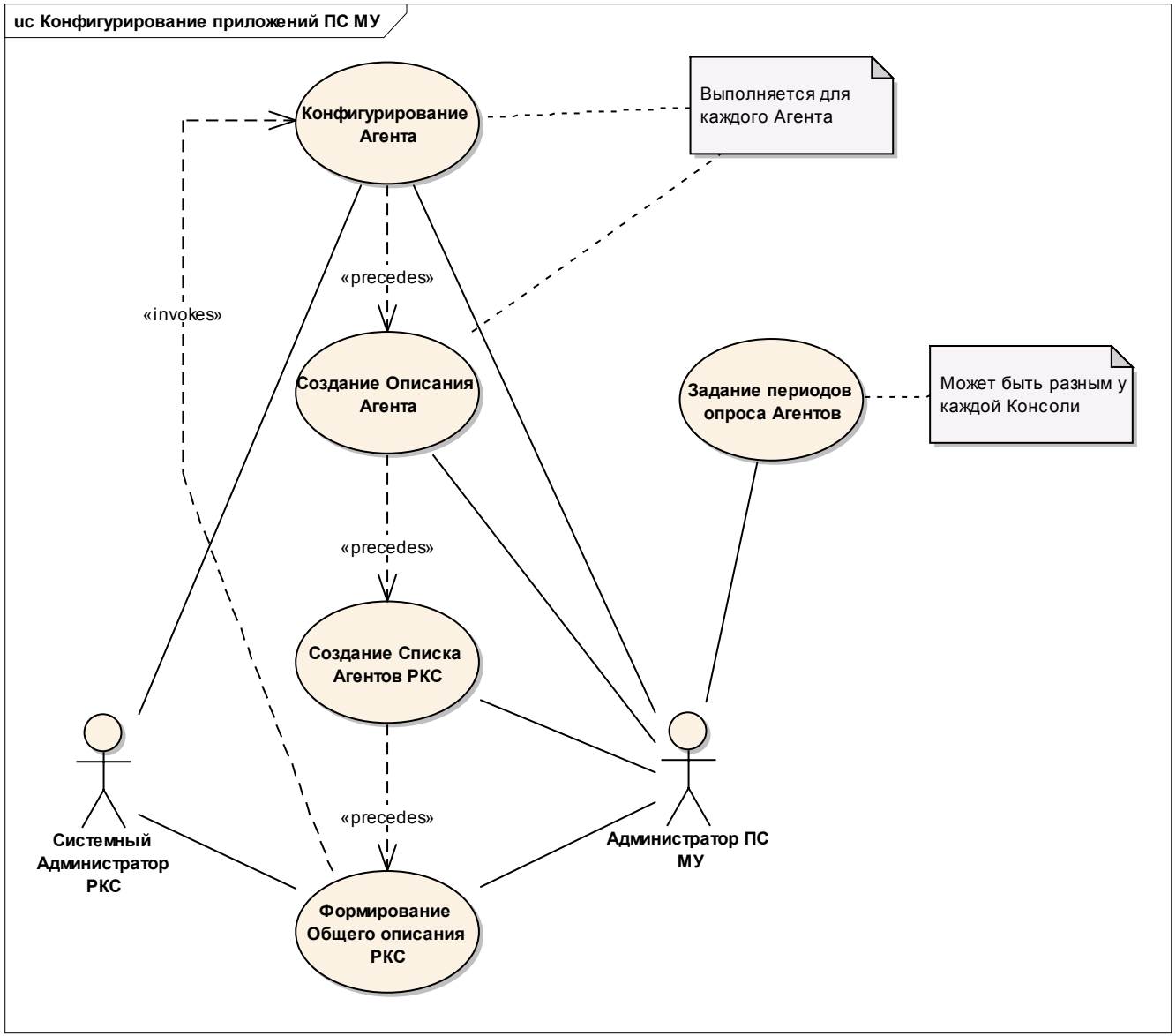
Use Case – 5/7

- Виды отношений:
  - Ассоциация (association) — актор инициирует вариант использования
- В том числе между прецедентами:
  - Расширение (extend) — между базовым вариантом использования и его специальным случаем
  - Включение (include) — взаимосвязь базового варианта использования с другим вариантом использования, функциональное поведение которого всегда задействуется базовым вариантом использования
  - Обобщение (Generalization, наследование) — моделирует соответствующую общность ролей

Use Case – 6/7


req Мониторинг
Наблюдение за системой
(from Применение)
«extend»
Диагностика
(from Применение)

Use Case – 7/7


uc Конфигурирование приложений ПС МУ
Конфигурирование Агента
Выполняется для каждого Агента
«precedes»
«invokes»
Создание Описания Агента
Задание периодов опроса Агентов
Может быть разным у каждой Консоли
«precedes»
Создание Списка Агентов РКС
«precedes»
Системный Администратор РКС
Администратор ПС МУ
Формирование Общего описания РКС

Связь Use Case с требованиями – 1/2

- Связь требований с Use Case (или целой диаграммой) может выражать:
  - Отношение «Реализуй такое поведение» – тогда целая диаграмма вариантов использования является иллюстрацией к требованию
  - Причину, по которой было сформировано то или иное требование – тогда используется связь типа Реализация (implements), которая показывается на общей диаграмме требований

Связь Use Case с требованиями – 2/2

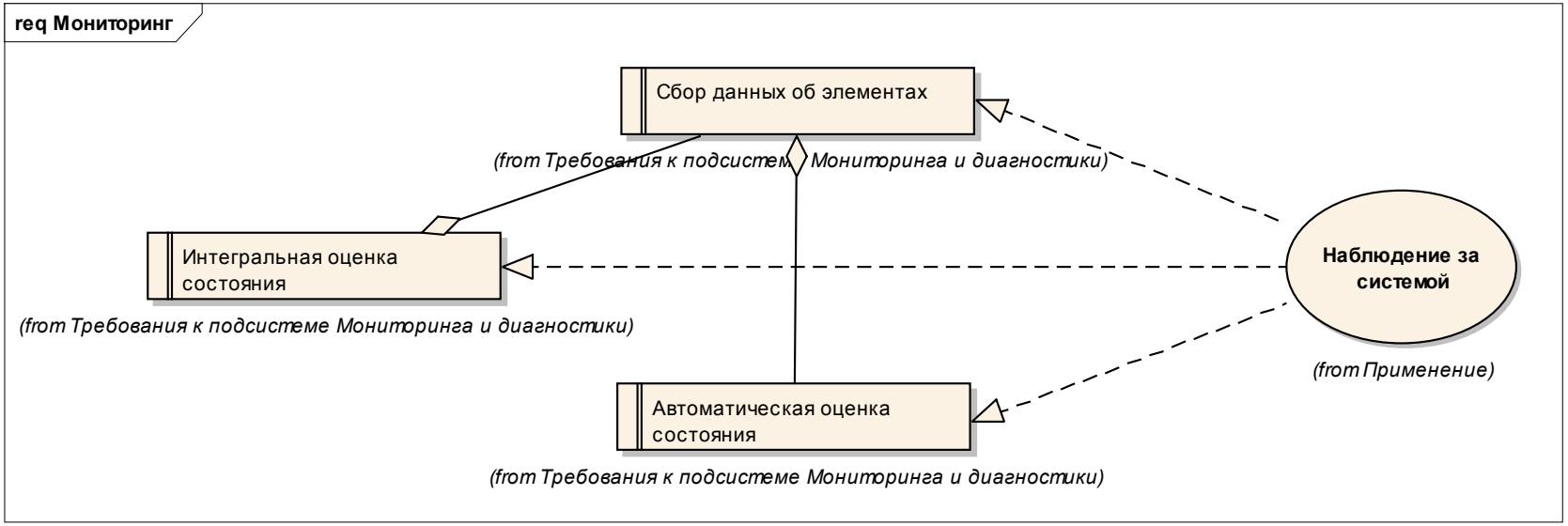


Диаграмма последовательности – 1/2

- Показывает взаимодействия объектов, упорядоченные по времени их проявления, с отражением продолжительности обработки и последовательности их проявления
- На ней изображаются:
  - Участвующие во взаимодействии объекты
  - Последовательность сообщений, которыми они обмениваются

Диаграмма последовательности – 2/2

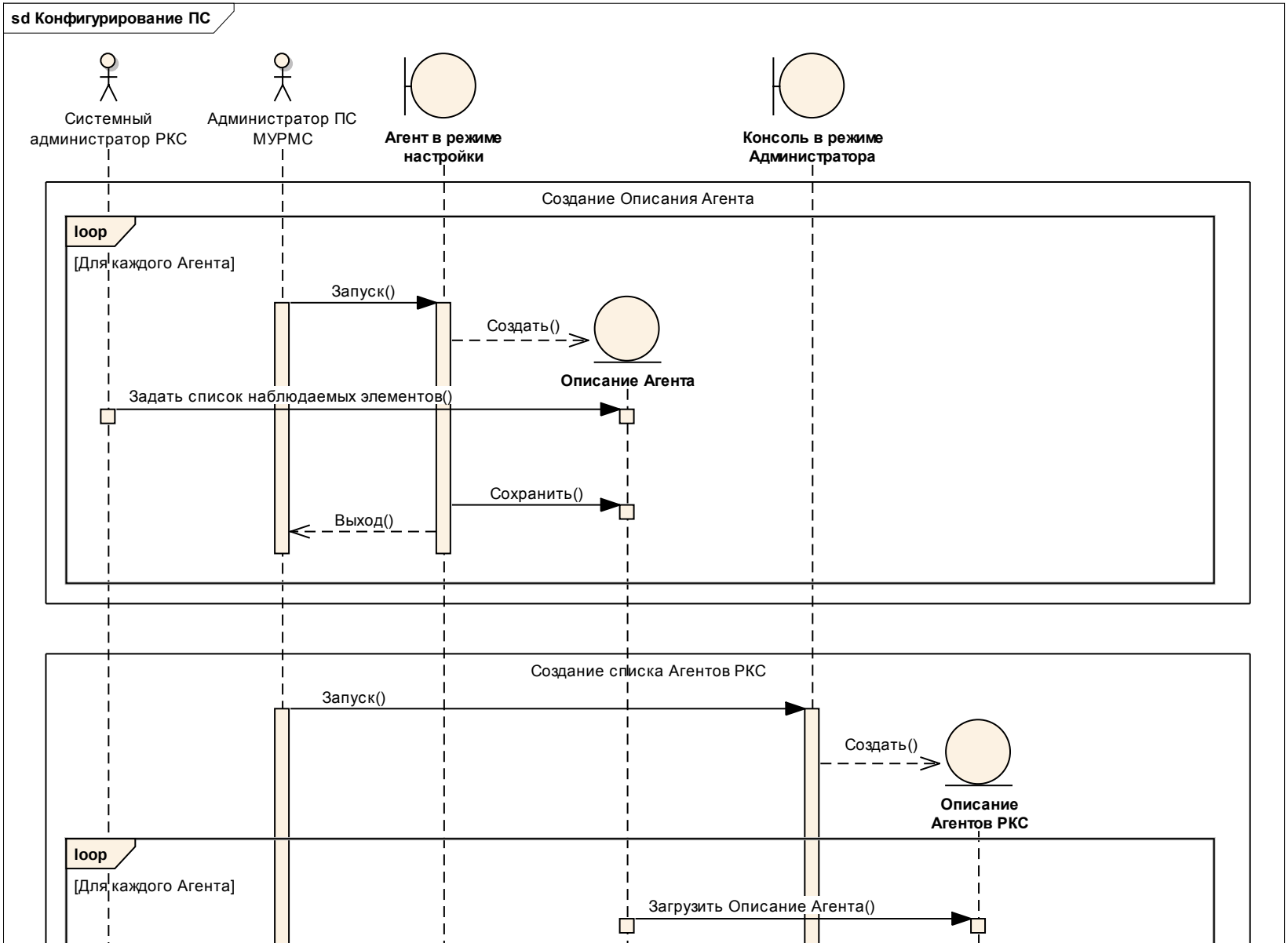


Диаграмма последовательности и требования

- Диаграммой последовательности может использоваться как иллюстрация к требованию, то есть выражать отношение «Реализуй такое поведение»
- Другим, более общим видом диаграммы, который описывает поведение системы, является Диаграмма деятельности, но она не описывает временные рамки

Диаграмма состояний – 1/3

- Представляет конечный автомат с простыми состояниями, переходами и композитными состояниями
  - Описывает поведение системы в целом или ее части
- На этапе составления требований используется для описания «типичных наборов условий» для системы
  - Предполагается, что поведение системы в каждом состоянии разное → функциональность
  - [Допустимые] Действия операторов / внешних систем различаются

Диаграмма состояний – 2/3

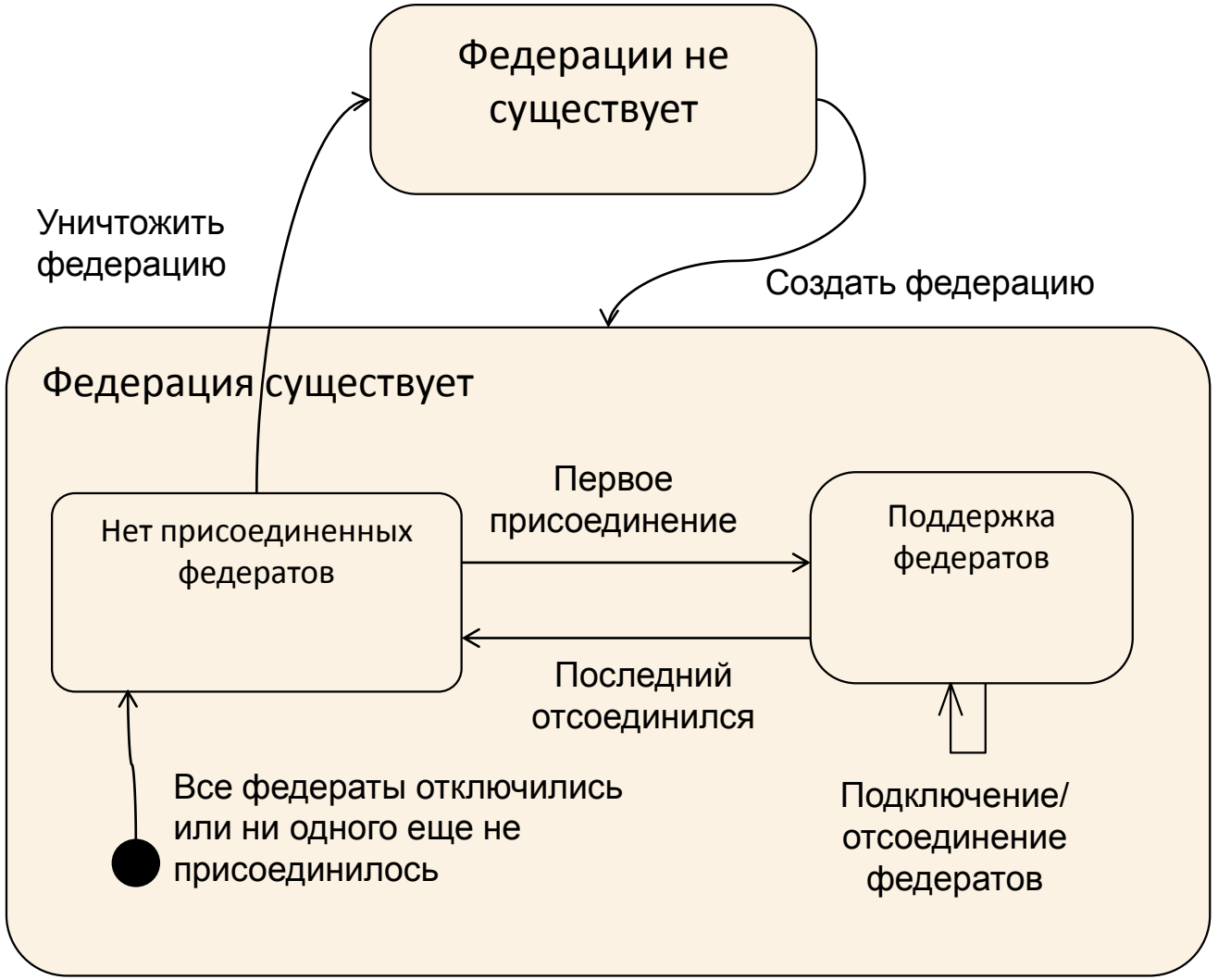


Диаграмма состояний – 3/3

- Начальное состояние / конечное состояние
- Состояние:
  - Название состояния
  - Активности, происходящие в данном состоянии
- Переходы
  - Название события, вызывающего переход
  - Охраняющее выражение
  - Действие при переходе
- Объединение или разветвление

Диаграмма состояний и требования

- Обычно встречается уже в требованиях разработчика
- Может привязываться:
  - К требованию в целом («реализуй такое поведение»)
  - К Use Case диаграмме («при переходе от варианта к варианту использования происходит смена состояния системы»)
  - К диаграмме последовательности («некоторые взаимодействия вызывают смену состояния»)

Все пропало

Ужас-ужас

КУРСОВОЙ ПРОЕКТ