

# **Анализ и проектирование на UML**

**Новиков Федор Александрович**

[fedornovikov@rambler.ru](mailto:fedornovikov@rambler.ru)

Курс подготовлен по заказу  
ООО Сан Майкросистемс СПб

**Часть 3**

Курс подготовлен при поддержке Sun Microsystems  
Правила использования материалов опубликованы на [www.sun.ru](http://www.sun.ru)

# План лекций

- Введение в UML
- Обзор языка
- ✓ Моделирование использования
- Моделирование структуры
- Моделирование поведения
- Управление моделями
- Тенденции развития языка
- UML и процесс разработки

# 3. Моделирование использования

- 3.1. Значение моделирования использования
- 3.2. Диаграммы использования
- 3.3. Реализация вариантов использования

# **3.1. Значение моделирования использования**

- **Зачем это нужно?**
- **Подходы к проектированию**
- **Преимущества моделирования  
использования**
- **Выводы**

# Зачем это нужно?

- **Диаграммы деятельности**  
**= блок схемы**
- **Диаграммы состояний**  
**= конечные автоматы**
- **Диаграммы классов**  
**= код в рамочке**
- **Ассоциации**  
**= диаграммы «сущность – связь»**
- **Диаграммы размещения**  
**= топология сети**
- **... и так далее**
- **Диаграммы использования**  
**= ... ???**

# Пример ТЗ: Отдел кадров

- ИС «Отдел кадров» предназначена для ввода, хранения и обработки информации о сотрудниках и движении кадров
- Прием, перевод и увольнение сотрудников
- Создание и ликвидация подразделений
- Создание вакансий и сокращение должностей

# Подходы к проектированию

- **Тор-down: система – подсистемы – модули ...**
  - Структура соответствует команде, а не задаче
- **БД: схема = таблицы + связи**
  - Табельный номер – атрибут сотрудника
- **ОО: словарь системы = классы**
  - Полнота и адекватность словаря

# **Недостатки традиционных ПОДХОДОВ**

- **Первый шаг выполняется в терминах проектируемой системы**
- **Только одна структура выбирается за основу:**
  - **Структурное проектирование – структура кода**
  - **Моделирование данных – структура хранения**
  - **Объектно-ориентированный подход – структура межмодульных интерфейсов**



# **Преимущества моделирования использования**

- **Простые утверждения**
  - **Субъекты, предикаты (и объекты)**
- **Абстрагирование от реализации**
  - **ЧТО делает система (но не КАК это делается и не ЗАЧЕМ это делать)**
- **Декларативное**
  - **но не императивное описание**
- **Выявление границ**
  - **но не черный ящик**

# **Прагматика моделирования использования**

- **ОЧЕНЬ** простые идеи и нотация
  - Применяется на ВСЕХ фазах (анализ, ..., тестирование)
  - Понимают ВСЕ (разработчики, заказчики, управленцы) одинаково
- **НЕ** зависит от остальных средств UML
  - Не меняется 1.1 → 2.0
  - Может использоваться отдельно
- **Управление разработкой**
  - Модель интерфейса (usability)
  - Выделение подсистем и компонентов (architecture)
  - План тестирования (use case → test cases)

# Зачем это нужно – вывод

- Традиционные подходы достаточны
- Опытный архитектор может с успехом использовать любой подход
- Моделирование использования позволяет неопытному архитектору совершать меньше грубых ошибок на ранних этапах проектирования

## 3.2. Диаграммы использования

- **Элементы и нотация**
- **Действующие лица и их идентификация**
- **Варианты использования и их идентификация**
- **Отношения элементов диаграмм использования**

# Элементы диаграмм использования

## ■ Сущности

- Действующие лица
- Варианты использования
- Примечания – применяются на всех диаграммах
- Пакеты – рассматриваются в части 6

## ■ Отношения

- Ассоциации между действующими лицами и вариантами использования
- Обобщения между действующими лицами
- Обобщения и зависимости между вариантами использования

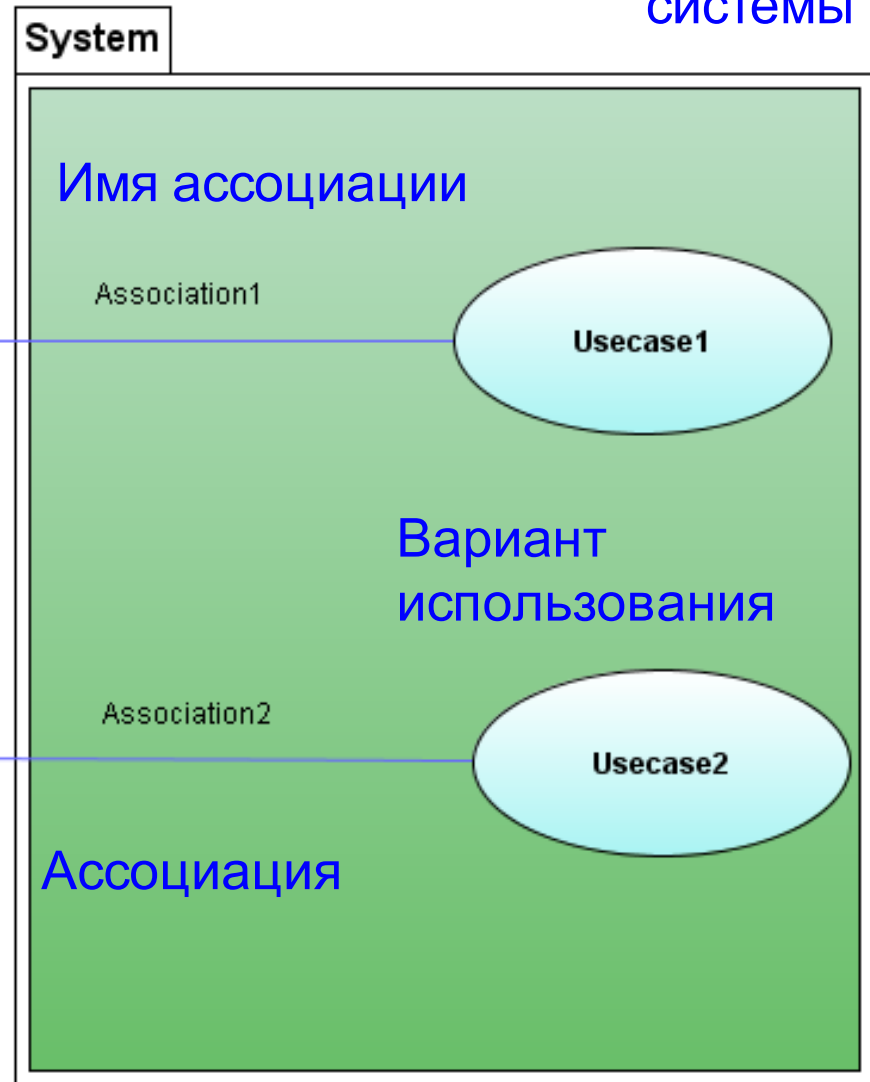
# Нотация (1)

Границы  
системы

Действующее  
лицо



Имя  
действующего  
лица



# Нотация (2)

Условие  
расширения

System

Вариант использования с  
точкой расширения

{condition}

Usecase1

values  
extension point1

<<extend>>

Usecase3

Обобщение  
вариантов  
использования

<<include>>

Зависимость

Usecase2

Usecase4

Обобщение  
действующих  
лиц

Actor1

Actor2



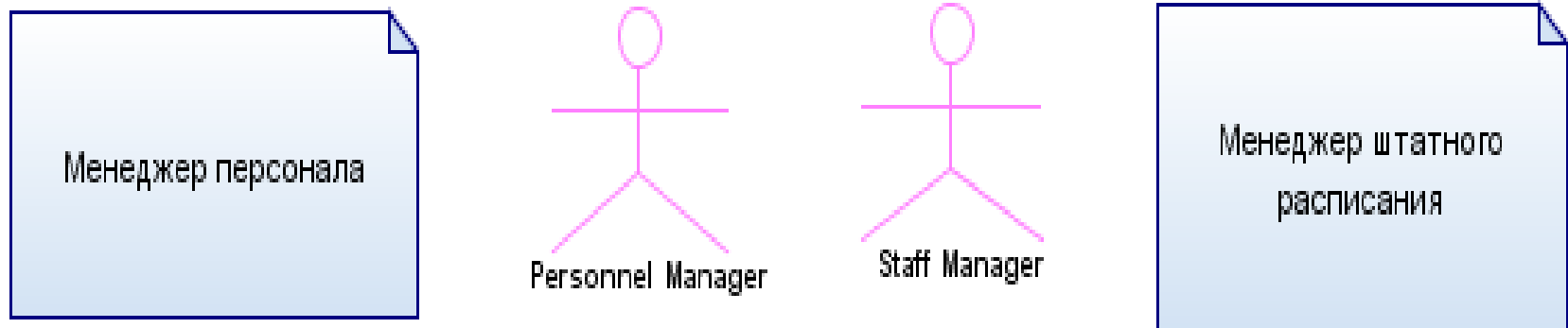
# Действующие лица и их идентификация

- Действующие лица находятся **ВНЕ** проектируемой системы
- Действующее лицо – это множество логически взаимосвязанных **РОЛЕЙ**
- Действующее лицо – это стереотипный **КЛАСС**
- Типовые случаи: категории пользователей, внешние программные и аппаратные средства



# Пример: действующие лица ИС ОК


- **Менеджер персонала**
  - Работает с конкретными людьми
- **Менеджер штатного расписания**
  - Работает с абстрактными должностями и подразделениями



# Варианты использования и их идентификация

- ***Вариант использования – множество возможных последовательностей событий/действий (сценариев), приводящих к значимому для действующего лица результату***
  - Иногда вариант использования называют сценарием ☹
- **Типичные случаи: пункты ТЗ**
- **Если ТЗ смутное, его можно (и нужно!) попробовать переписать фразами субъект – предикат – объект**

# Почему actor → «действующее лицо»?

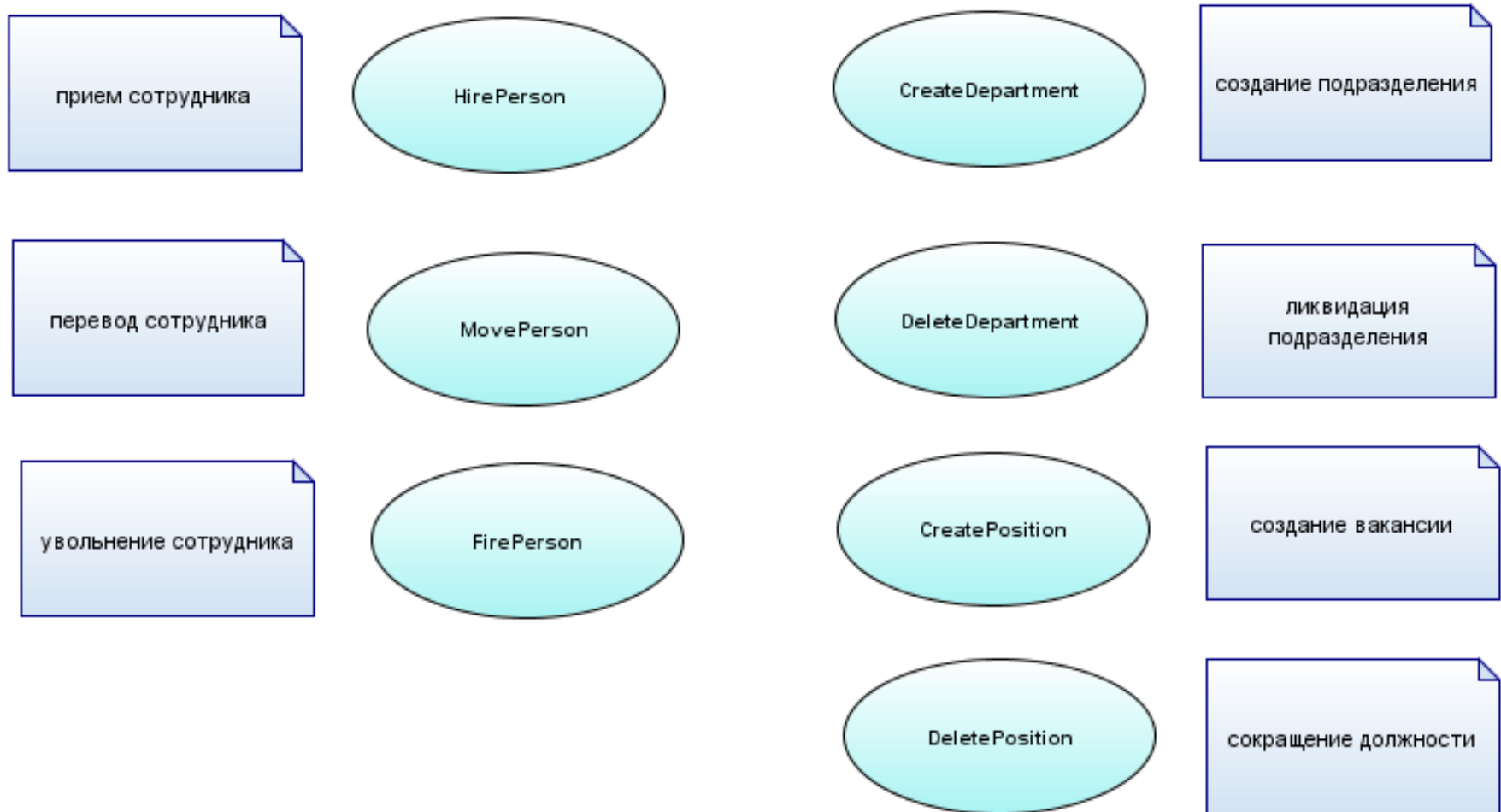
- Альтернативные варианты: актёр (неточно), актер (нет такого слова), актант (есть, но никто не знает)
- Пьеса Шекспира «Гамлет» – модель
- Фильм Козинцева – экземпляр модели
- Гамлет – действующее лицо пьесы
- Смоктуновский – актёр, играющий роль Гамлета в фильме Козинцева по пьесе Шекспира
- «Худой человечек»  означает Гамлета, но не Смоктуновского!

Actor

# Пример: варианты использования ИС ОК

- Менеджер персонала выполняет действия
- Прием сотрудника
- Перевод сотрудника
- Увольнение сотрудника
- Менеджер штатного расписания выполняет действия
- Создание подразделения
- Ликвидация подразделения
- Создание вакансии (=должности)
- Сокращение должности

# Пример: варианты использования ИС ОК



# Примечания

- T3 (requirements) – функциональные и НЕ функциональные требования
- Группировка требований (по приоритетам)

**<<requirement>>**

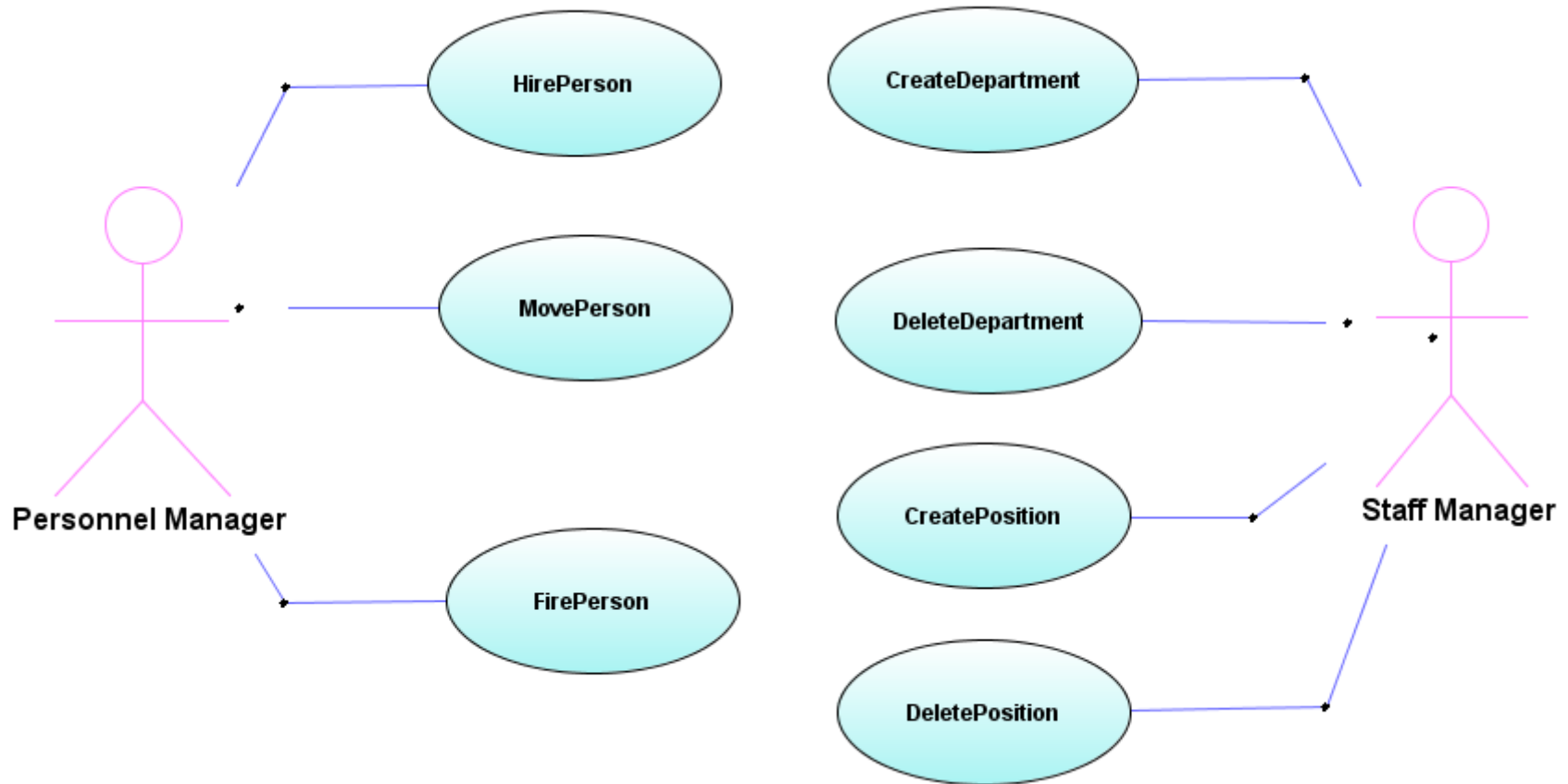
**Информация о текущем состоянии  
штатного расписания и составе  
персонала должна храниться  
постоянно**



# Отношения элементов диаграмм использования

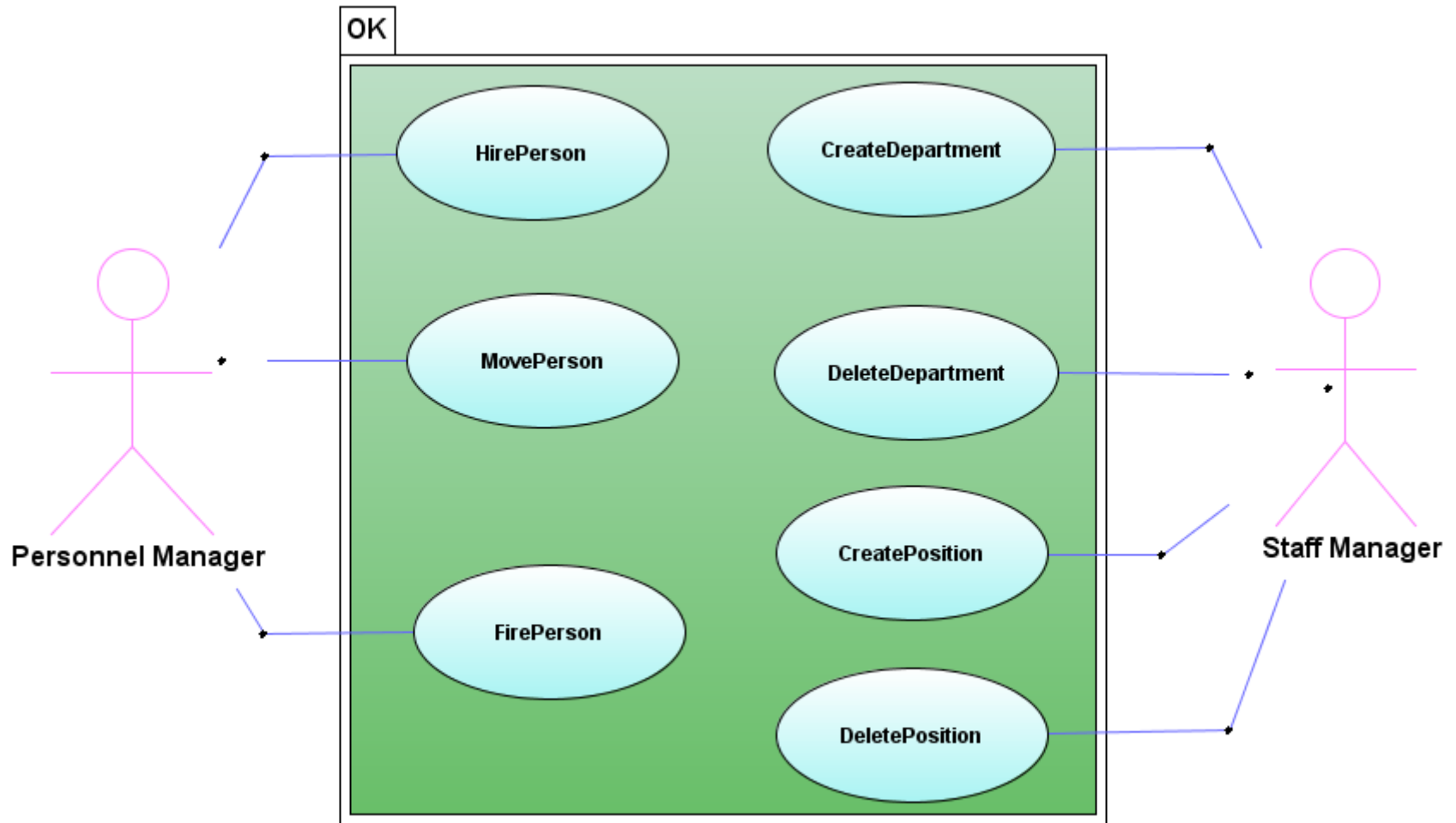
- Ассоциация между действующим лицом и вариантом использования
- Обобщение между действующими лицами
- Обобщение между вариантами использования
- Зависимость между вариантами использования
- Зависимость между пакетами

# Ассоциации между действующими лицами и вариантами использования

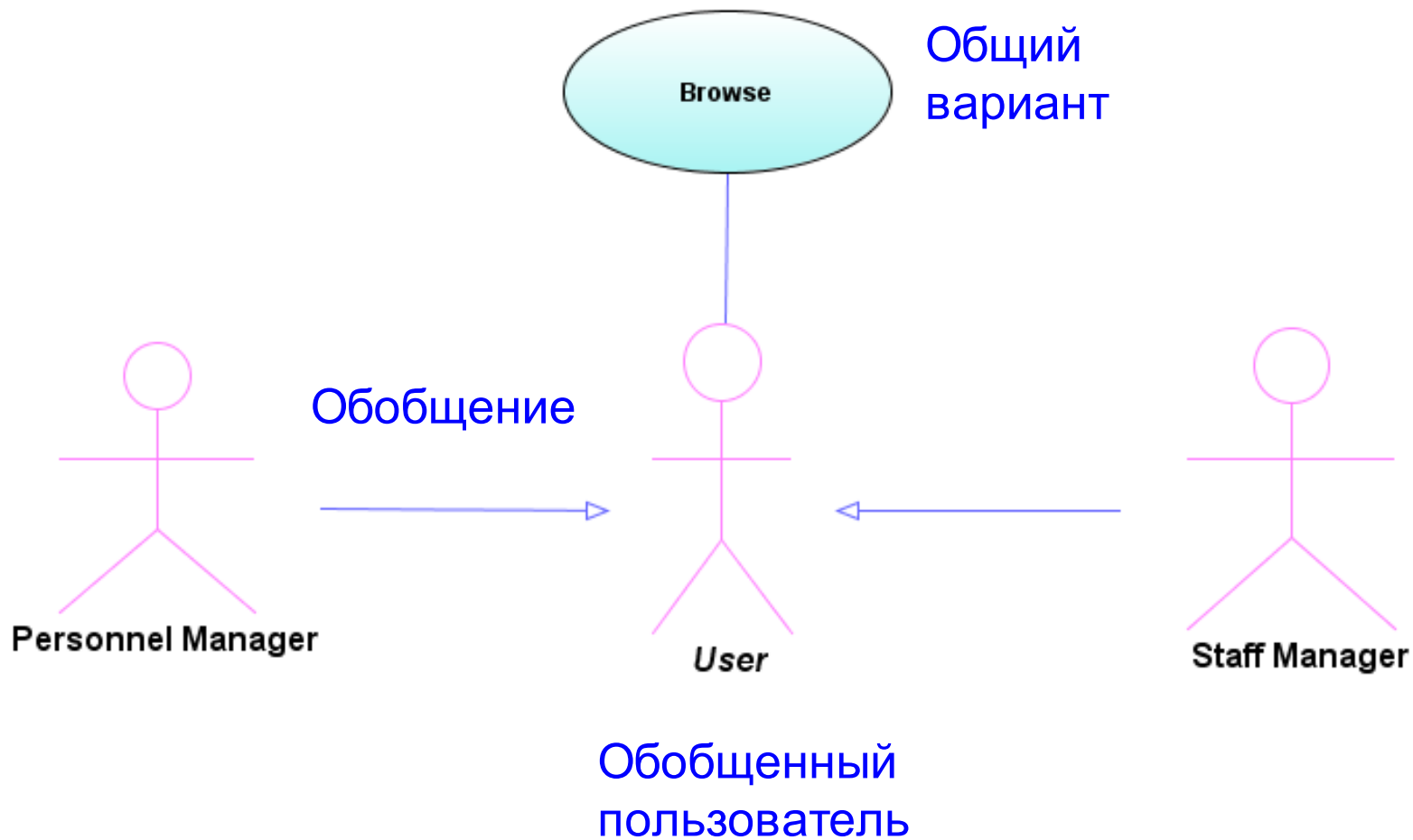




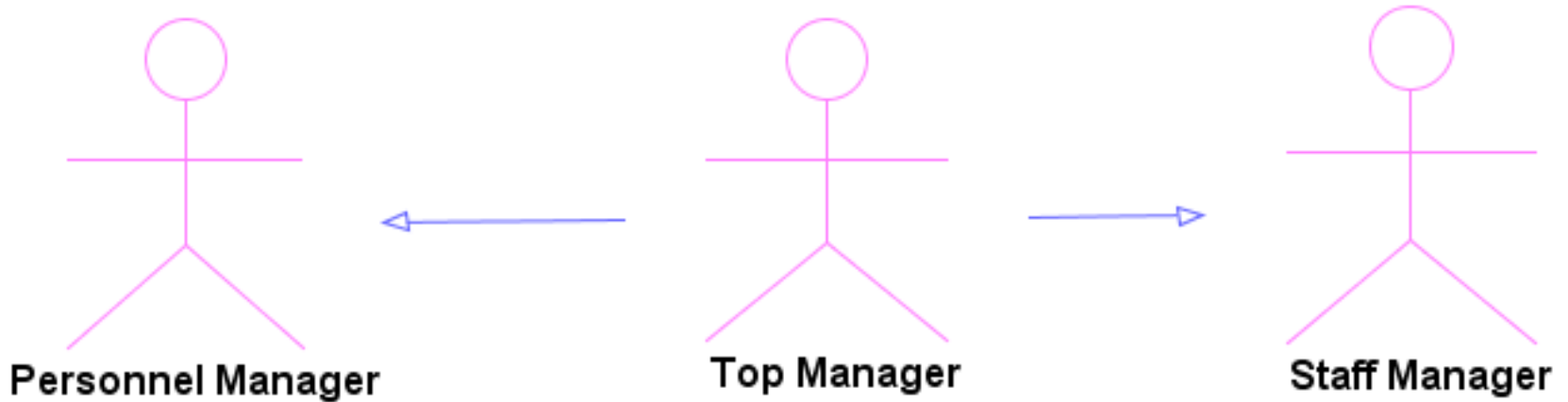
# Границы системы



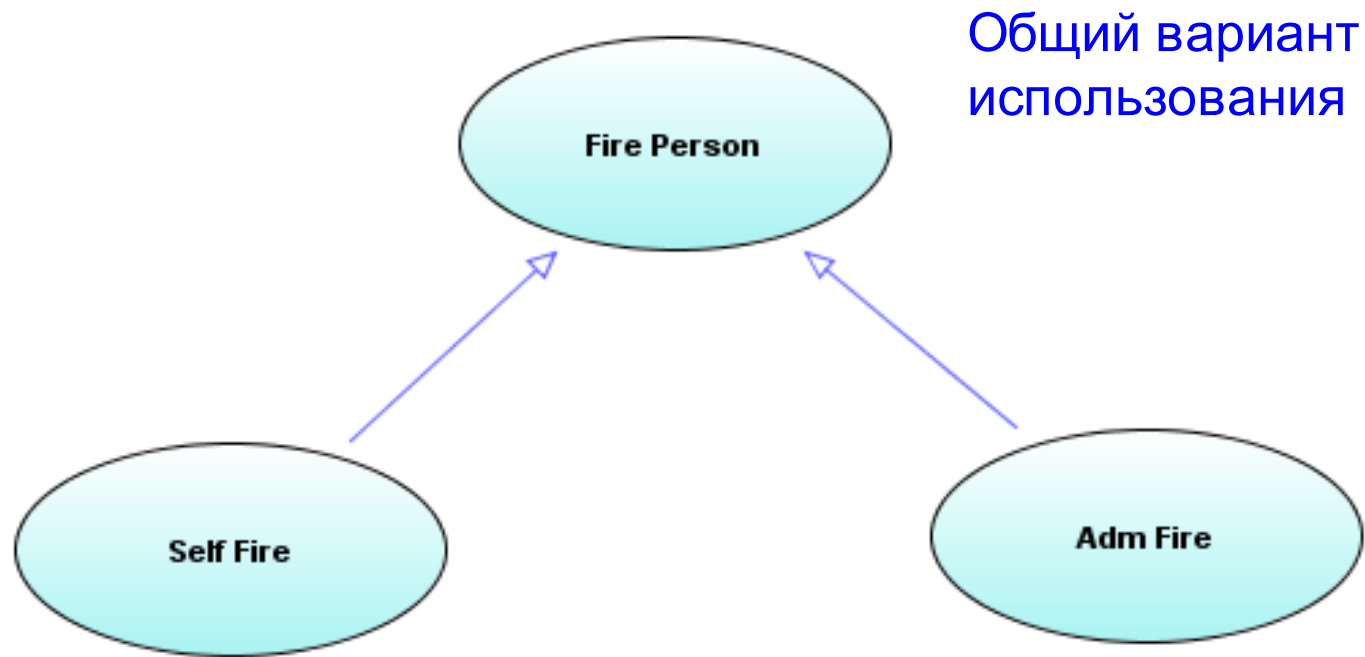
# Обобщение вариантов использования (1)



# Обобщение вариантов использования (2)



# Обобщение вариантов использования

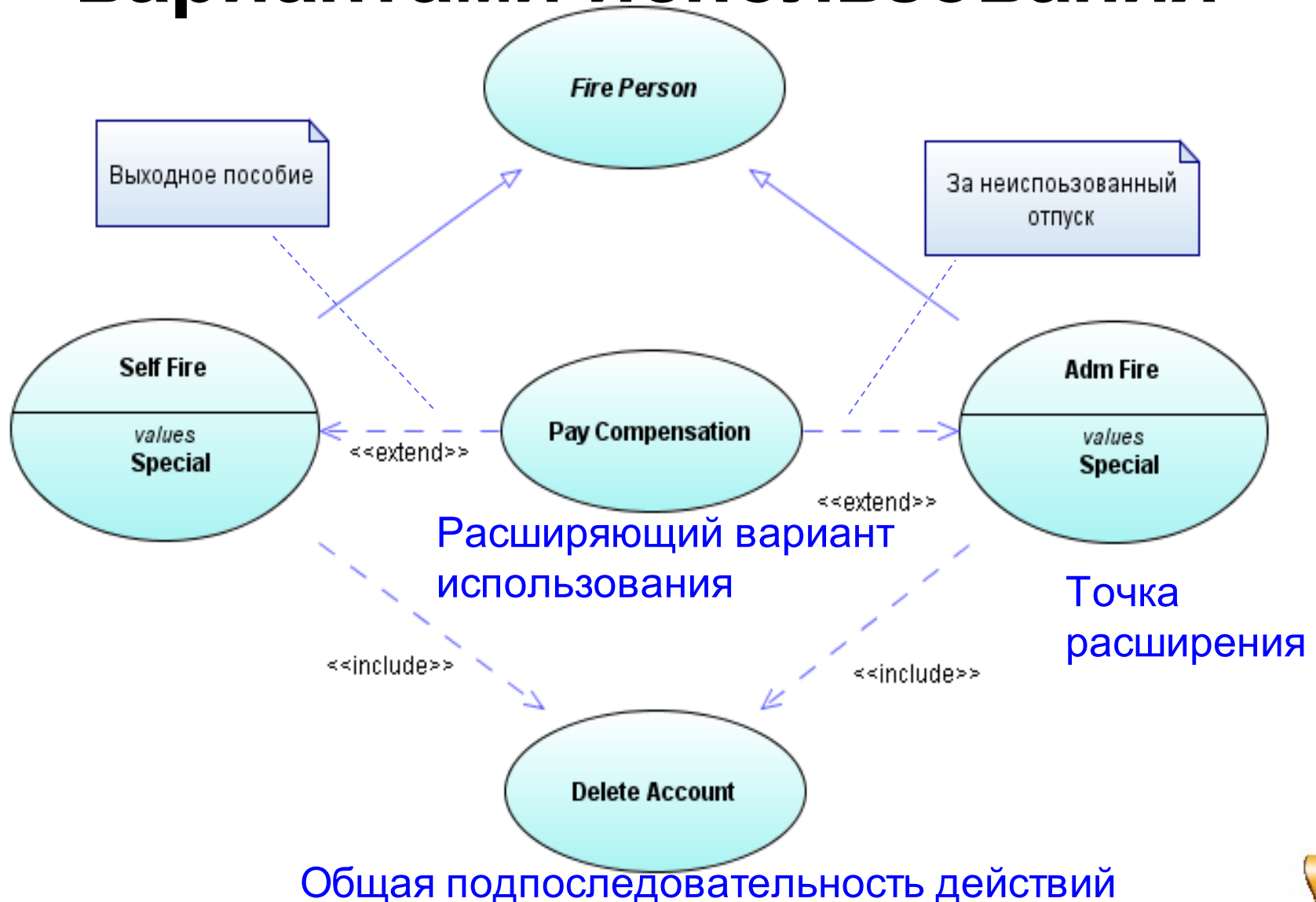


Общий вариант  
использования

Частный вариант  
использования



# Зависимости между вариантами использования



# **3.3. Реализация вариантов использования**

- **Переход от анализа и постановки задачи к решению (проектированию)**
- **Типовые сценарии и исключения**
- **Текстовые описания (user story)**
- **Программы на псевдокоде**
- **Диаграммы деятельности**
- **Диаграммы взаимодействия (последовательности и коммуникации)**

# Текстовые описания

## ■ Увольнение по собственному желанию

1. Сотрудник пишет заявление
2. Начальник подписывает заявление
3. Если есть неиспользованный отпуск, то бухгалтерия рассчитывает компенсацию
4. Бухгалтерия рассчитывает выходное пособие
5. Системный администратор удаляет учетную запись
6. Менеджер штатного расписания обновляет базу данных

# Программы на псевдокоде

## ■ SelfFire

- Получить заявление
- Special:
- Рассчитать сотрудника
- include DeleteAccount
- Обновить информацию в базе данных

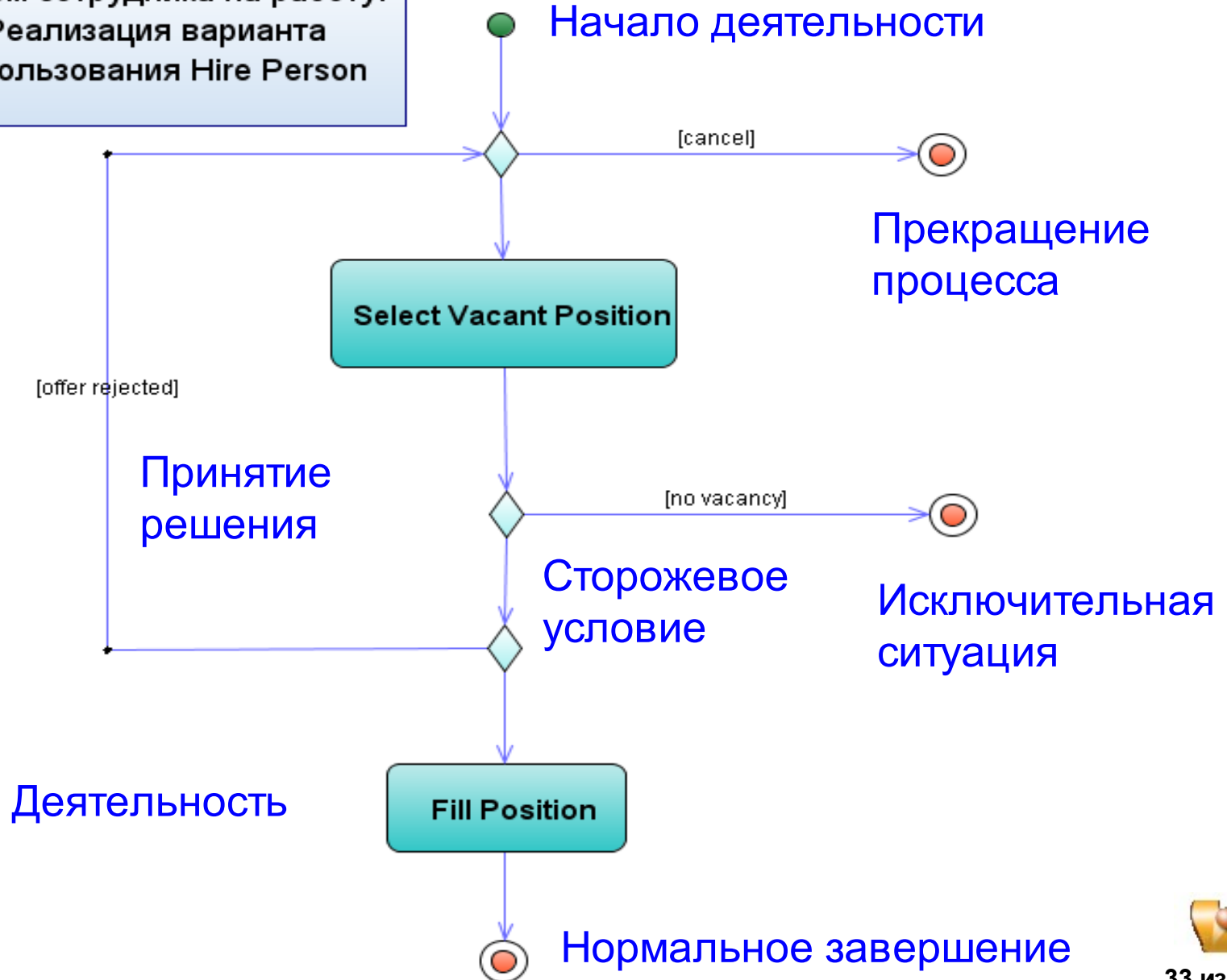
## ■ AdmFire

- Получить приказ
- Special:
- Рассчитать сотрудника
- include DeleteAccount
- Обновить информацию в базе данных

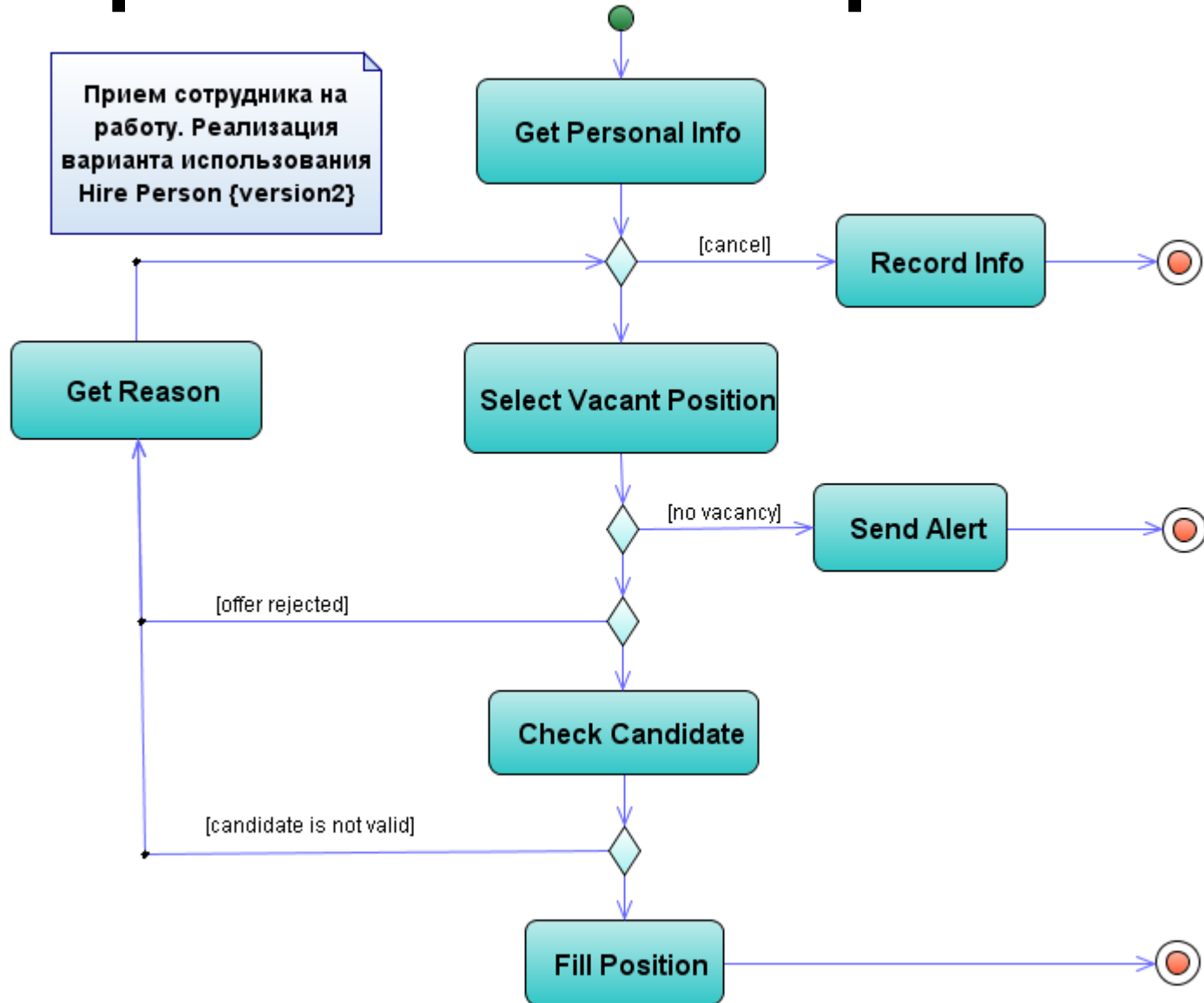


# Реализация диаграммами деятельности

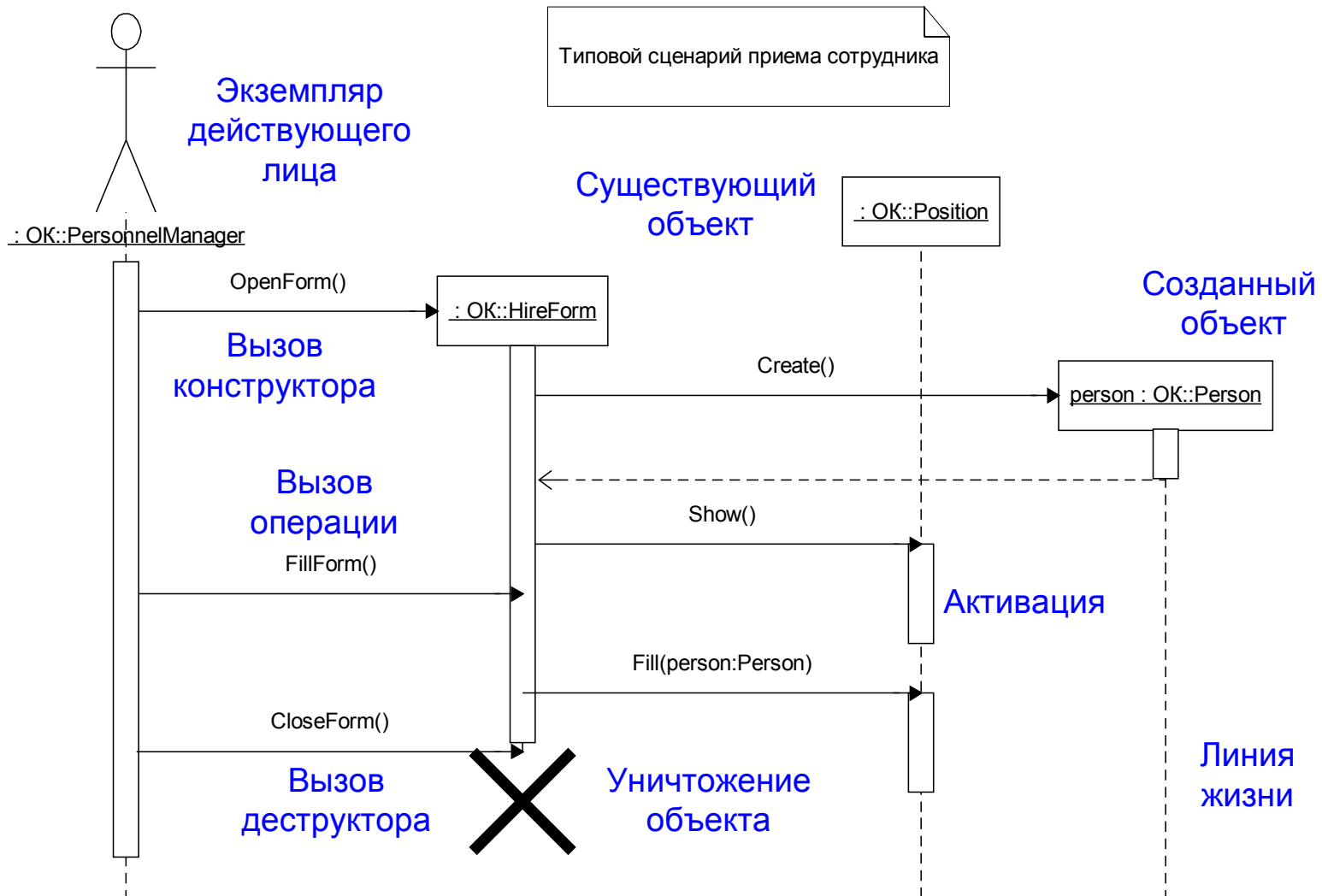
Прием сотрудника на работу.  
Реализация варианта  
использования Hire Person



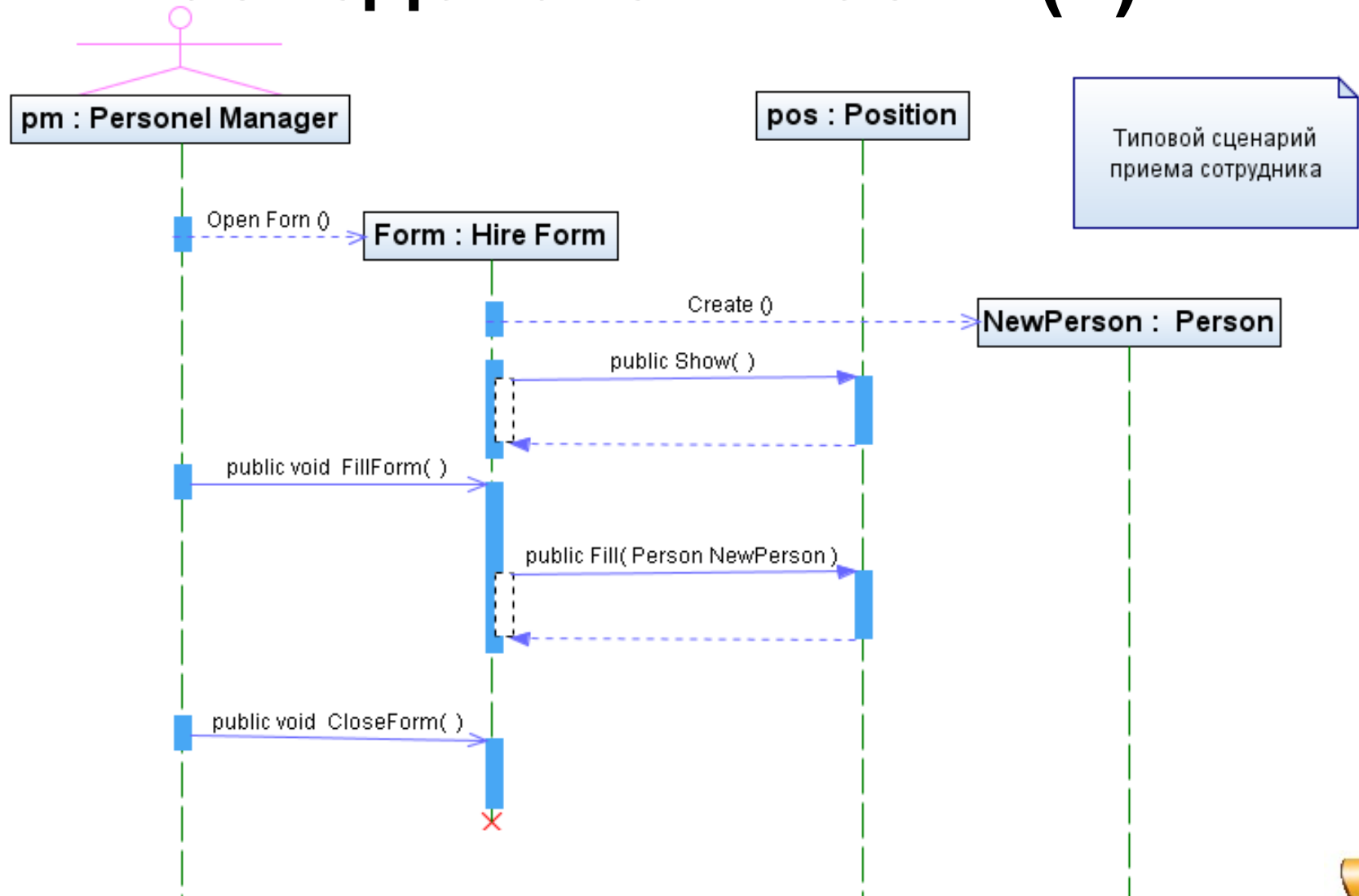
# Усовершенствование реализации



# Реализация диаграммами последовательности (1)

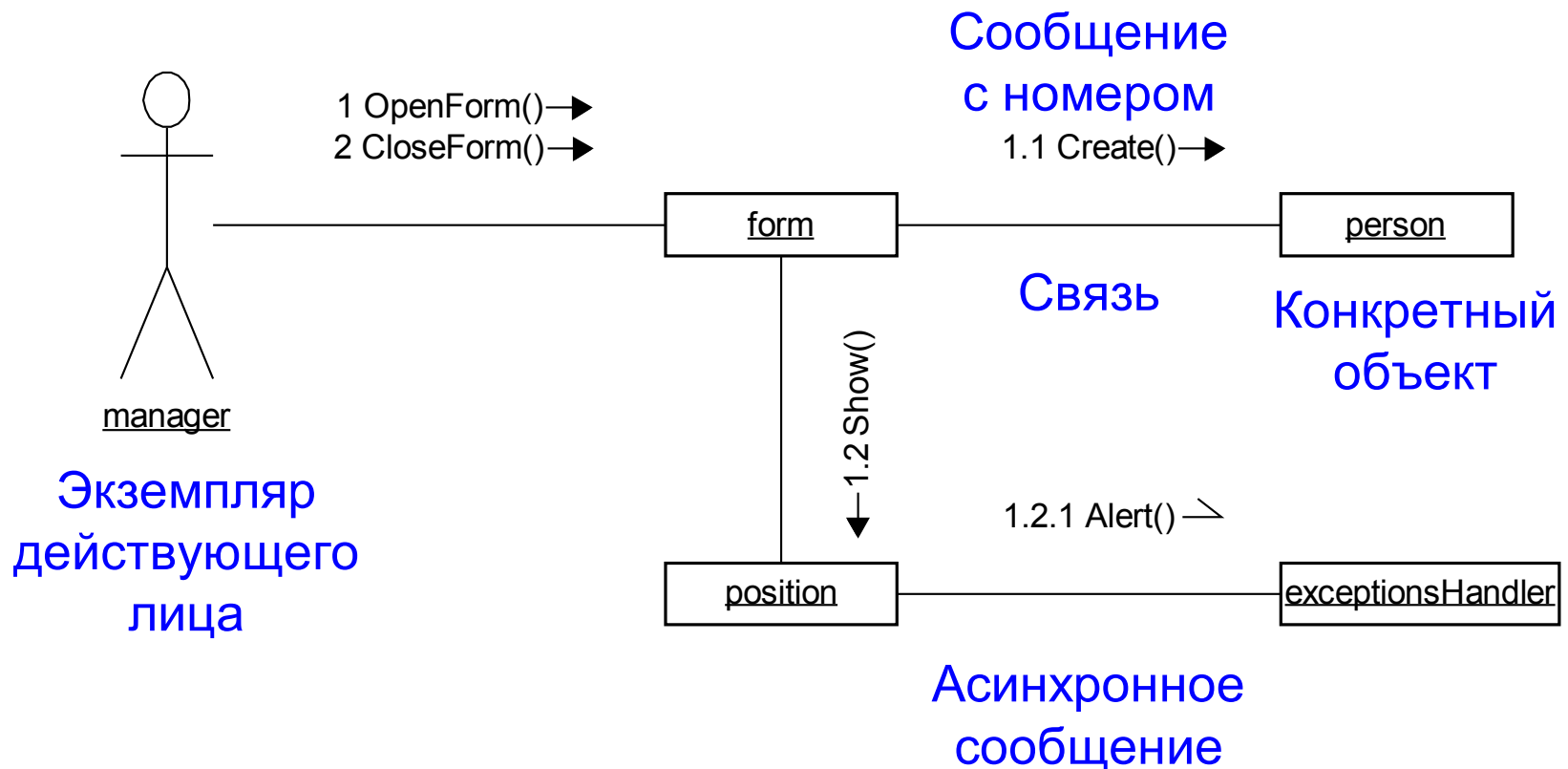


# Реализация диаграммами последовательности (2)



# Реализация диаграммами коммуникации

Исключительная ситуация при приеме сотрудника



# **Сравнение способов реализации вариантов использования (1)**

- **Текстовые описания**
  - **Всем понятно, привычно и удобно**
  - **Длинно и неточно, пропуски и ошибки**
  - **Есть трансляторы в варианты использования (!)**
- **Программы на псевдокоде**
  - **Традиционное средство программистов**
  - **Компактнее текстового описания**
  - **Навязывают структуру реализации**
  - **Не приближает к объектной модели**

# Сравнение способов реализации вариантов использования (2)

- **Диаграммы деятельности**
  - Псевдокод эквивалентен блок-схемам (с точностью до параллелизма)
  - Наглядно, но менее компактно
  - Почти не приближают к объектной модели
- **Диаграммы взаимодействия**
  - Сложная и непривычная нотация
  - Диаграммы объектного уровня – описывают **ОДИН** сценарий – нужно **МНОГО** диаграмм
  - Прямо ведут к объектной модели

# Выводы

- **Диаграмма использования – первый шаг моделирования**
- **Основное назначение – показать, что делает система во внешнем мире**
- **Не обязательно соответствует структуре классов, модулей и компонентов**
- **Адекватная идентификация действующих лиц и вариантов использования – ключ к успеху**
- **Способ реализации – дело вкуса**