

Моделирование на UML. Вторая ступень

Моделирование использования



Иванов Д.Ю., Новиков Ф.А.



Ай Ти Ментор ©2008-2010

<http://uml3.ru/>

Структура курса

Об этом курсе

Часть 1. Введение в UML

✓ **Часть 2. Моделирование использования**

Часть 3. Моделирование структуры

Часть 4. Моделирование поведения

Часть 5. Дисциплина моделирования

Содержание

1. Бизнес-анализ и моделирование
2. Значение моделирования использования
3. Диаграммы использования
4. Выявление и анализ требований
5. Реализация вариантов использования
6. Выводы

1. Бизнес-анализ и моделирование

Зачем рассматривать?

Бизнес-приложения =

центр области применимости UML →

Что это? = Введение в бизнес-анализ

Зачем это? = Реинжиниринг бизнеса

Как это? = Типы моделирования

Как было? = История развития средств
моделирования

Что сейчас? = Современное состояние

Введение в бизнес-анализ

- **Бизнес (business)** — это самостоятельная, осуществляемая на свой риск **деятельность**, направленная на **систематическое** получение прибыли от пользования имуществом, продажи товаров, выполнения работ или оказания услуг лицами в установленном законом порядке.
- **Бизнес** – систематическая деятельность
- **Бизнес-процесс** – последовательность действий (операций), в результате которой происходит выполнение некоторой *функции*
- **Бизнес-модель** – конструктивное (измеримое) описание *бизнес-процессов*
- **Бизнес-анализ** – процесс построения бизнес-модели

Реинжиниринг бизнеса (i)

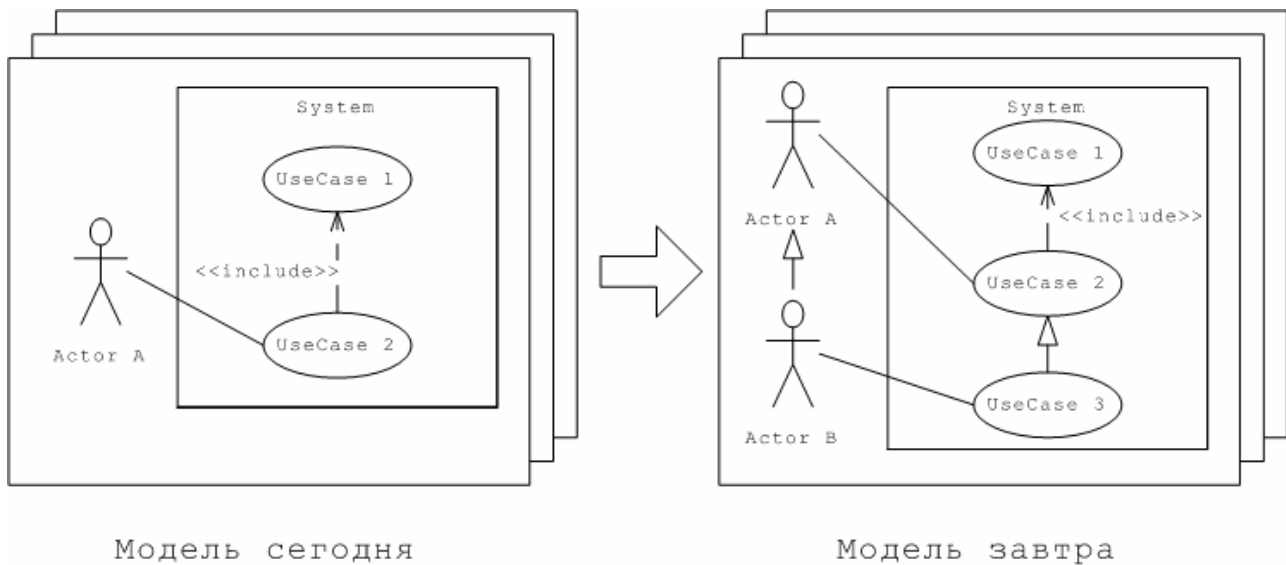
Меняется жизнь →

→ надо менять бизнес процессы →
→ модификация бизнес-приложений

Целенаправленное изменение бизнес-процессов требует **бизнес-модели**

В противном случае “неуклонный подъем”

Реинжиниринг бизнеса (ii)



Типы моделирования

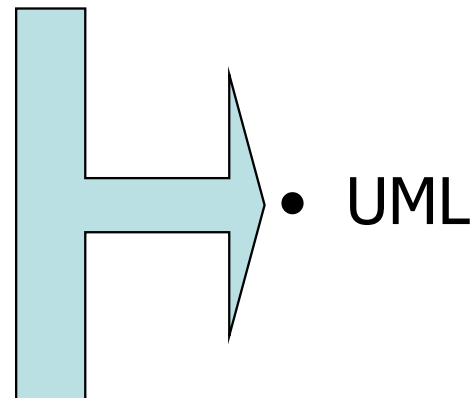
- **Имитационное моделирование** (simulation) – составление моделей с целью **количественного анализа**: число, структура и т.д.
- ✓ **Описательное моделирование** (modeling) – составление моделей с целью **качественного анализа**: есть – нет, много – мало и т.д.

Описательное моделирование

- Применение:
 - Руководство мыслительным процессом
 - Описание требований к системе
 - Разработка проекта системы
 - Навигация в больших системах
- Принципы:
 - Абстракция, а не детализация
 - В модель следует включать только важное
 - Спецификация, а не реализация
 - Что делает система, а не как это делается
 - Описания, а не реальные объекты

История развития средств моделирования

- Структурный анализ – SADT
- IDEF0
- ...
- Поток данных
- ...
- IDEF3
- ...
- Сущность-связь – ERD
- ...
- BPMN



Структурный анализ и проектирование

- **SADT (Structured Analysis and Design Technique):**
 - методология построения моделей
 - графический язык для представления моделей
- Идея структурности = последовательное уточнение = функциональная декомпозиция:
 - Блок, описывающий функцию на верхнем уровне может быть раскрыт на диаграмме нижнего уровня

Дуг Росс

Дуглас Т. Росс (D. T. Ross) (1929–2007) американский ученый и инженер, пионер компьютерной индустрии

- Разработал APT – Automatically Programmed Tools – язык для программирования станков с ЧПУ
- Придумал термин CAD (computer-aided design).
- Предложил методологию SADT, на основе которой было в дальнейшем развито семейство стандартов IDEF



SADT : основные понятия

- **Система** — совокупность взаимодействующих компонентов и взаимосвязей между ними
- **Моделирование** — процесс создания точного описания системы
- **SADT-Модель** — полное и точное описание системы с помощью SADT, для получения ответов на вопросы относительно системы с заданной точностью
- **Точка зрения** — место человека или объекта, с которого можно увидеть систему в действии

Функциональное моделирование

- **Блоки** изображают функции моделируемой системы
- **Дуги** связывают блоки и отображают взаимодействия и взаимосвязи между ними
- **Доминирование** — влияние, которое один блок оказывает на другие блоки диаграммы (выше-ниже)

Типы взаимосвязей между блоками

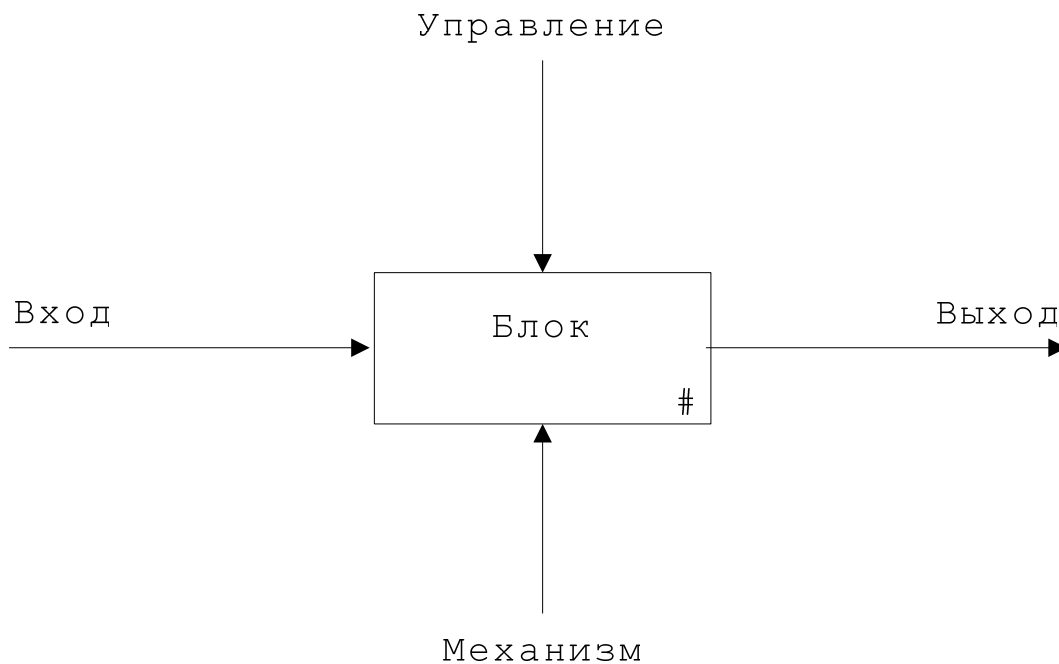
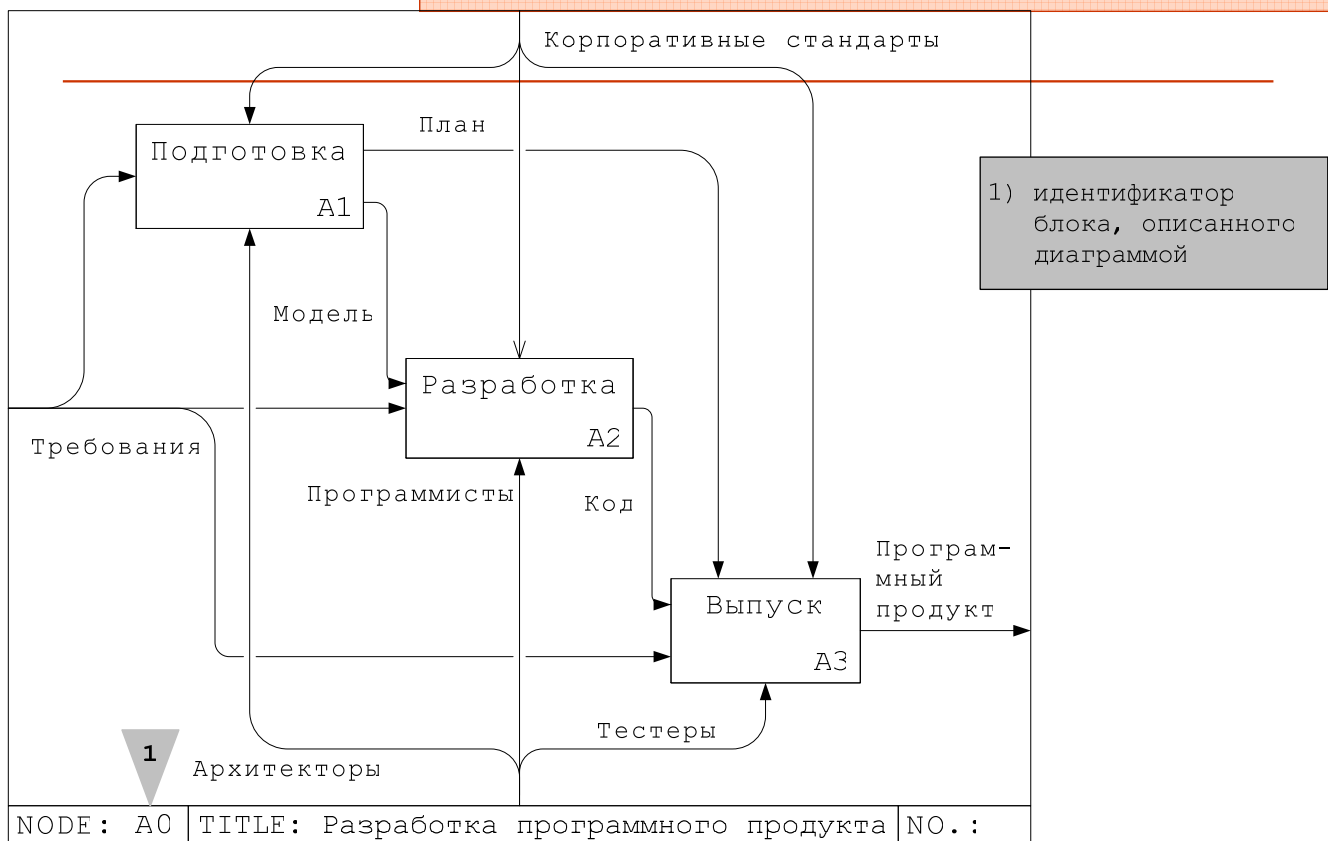


Диаграмма
верхнего
(нулевого)
уровня

Нотация –
стандарт IDEF0

- 1) вход
- 2) выход
- 3) управление
- 4) механизмы
- 5) идентификатор блока

Диаграмма первого уровня



© 2008-2010

АйТи Ментор

Д. Иванов, Ф. Новиков. Моделирование на UML. Вторая ступень
Моделирование использования

17 из 81

Диаграммы потоков данных

- Диаграмма потоков данных это орграф:
 - **Узлы** — процессы обработки данных
— сущности, производящие, потребляющие и хранящие данные
 - **Дуги** — данные, передающиеся между процессами и сущностями
- **НЕ** показана последовательность операций

© 2008-2010

АйТи Ментор

Д. Иванов, Ф. Новиков. Моделирование на UML. Вторая ступень
Моделирование использования

18 из 81

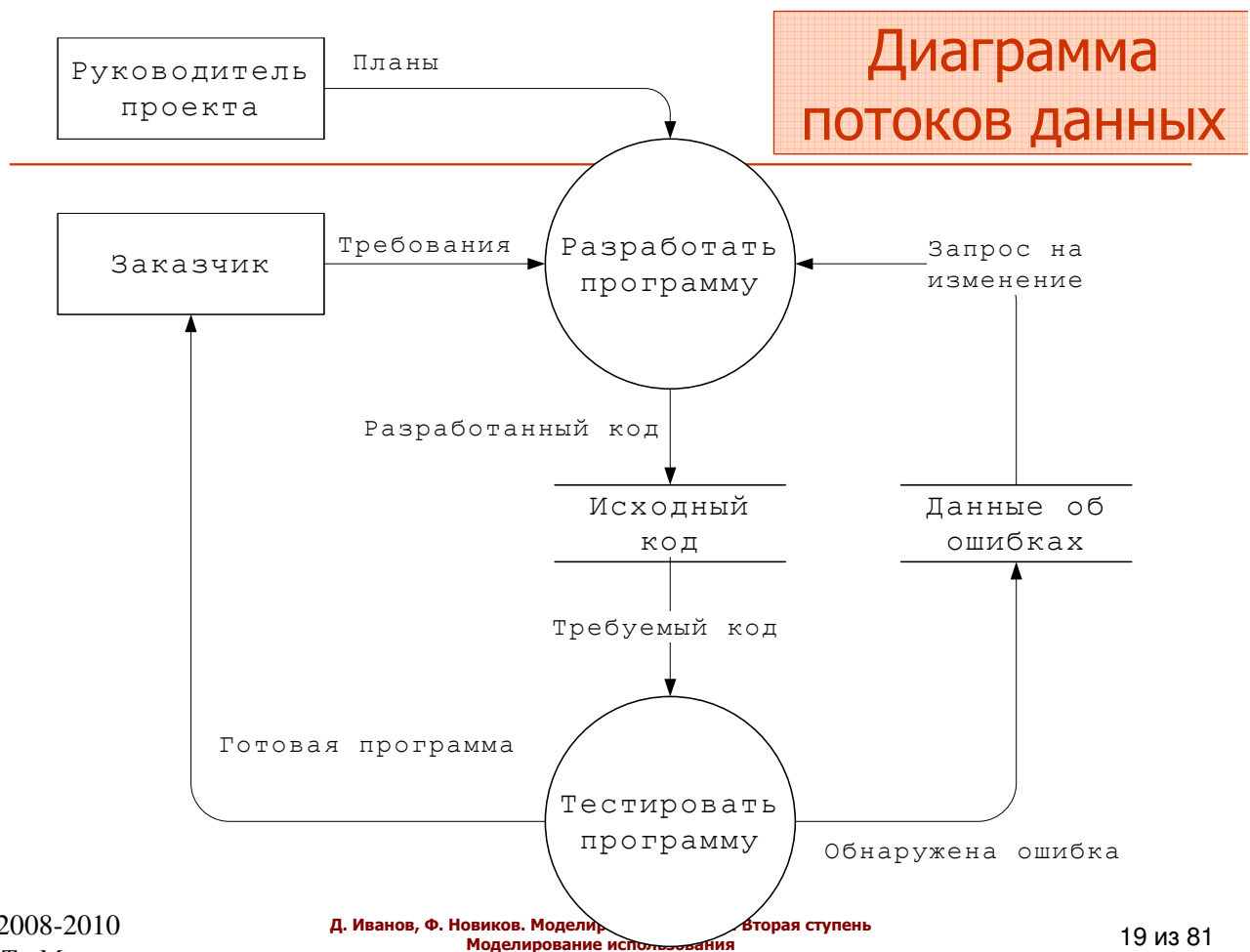


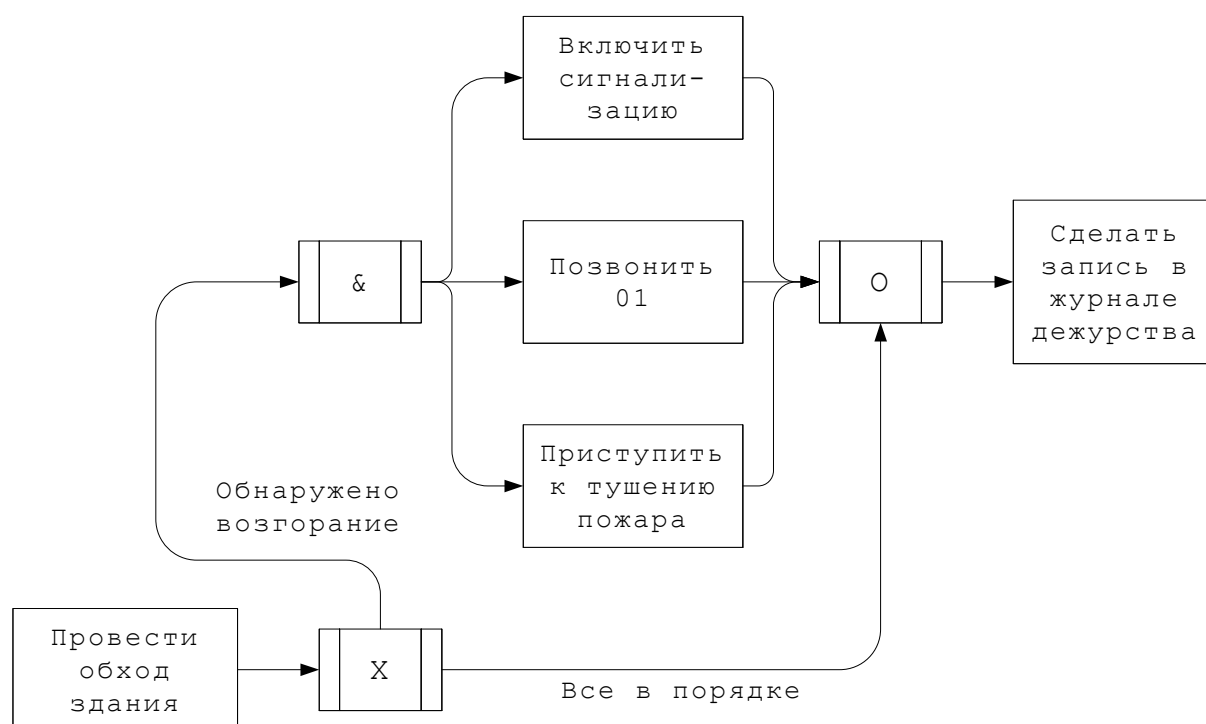
Диаграмма в нотации IDEF3

- Вариация на тему блок-схем алгоритмов
 - выполняемые действия
 - нотация – прямоугольник
 - последовательность действия
 - нотация – стрелки
- Унифицированный подход к описанию **альтернативного** и **параллельного** выполнения действий

Типы соединений IDEF3

Знак	Название	Вид	Правило
&	And	Разворачивающее	Все целевые действия одновременно запускаются
	И	Сворачивающее	Все исходные действия должны завершиться
X	Xor	Разворачивающее	Одно и только одно целевое действие запускается
	Исключающее ИЛИ	Сворачивающее	Одно и только одно исходное действие должно завершиться
O	Or	Разворачивающее	Одно или более целевое действие запускается
	или	Сворачивающее	Одно или более исходное действие должно завершиться

Блок-схема в нотации IDEF3



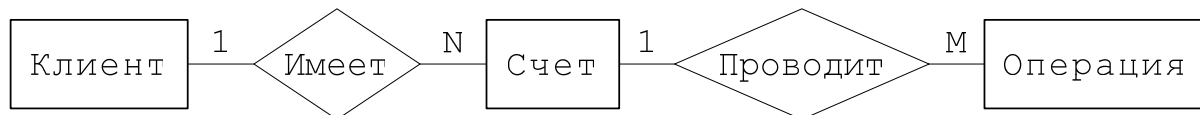
Диаграммы “сущность – связь”

Сущность – объект, видимый пользователю

- Атрибут – свойство сущности
- Идентификатор – набор атрибутов, идентифицирующих экземпляры сущности

Связь – это отношение между сущностями

- один к одному
- один ко многим
- многие ко многим



Питер Чен

Питер Чен (Peter Pin-Shan Chen) — американский ученый тайваньского происхождения

- Лауреат множества премий в области информационных технологий
- Предложил модель “сущность-связь” в 1976 г. в статье “The Entity-Relationship Model – Towards a Unified View of Data”



Business Process Modeling Notation

- **закрытый бизнес-процесс** (private business process) — процесс описывается сам по себе, без связи с окружением, как последовательность действий
- **полуоткрытый бизнес-процесс** (abstract business process) — бизнес-процесс связан с другими бизнес-процессами, но эти последние описаны абстрактно, только как приемники и источники сообщений
- **открытый бизнес-процесс** (collaboration business process) — показана внутренняя структура участвующих процессов и их взаимодействие

© 2008-2010
АйТи Ментор

Д. Иванов, Ф. Новиков. Моделирование на UML. Вторая ступень
Моделирование использования

25 из 81

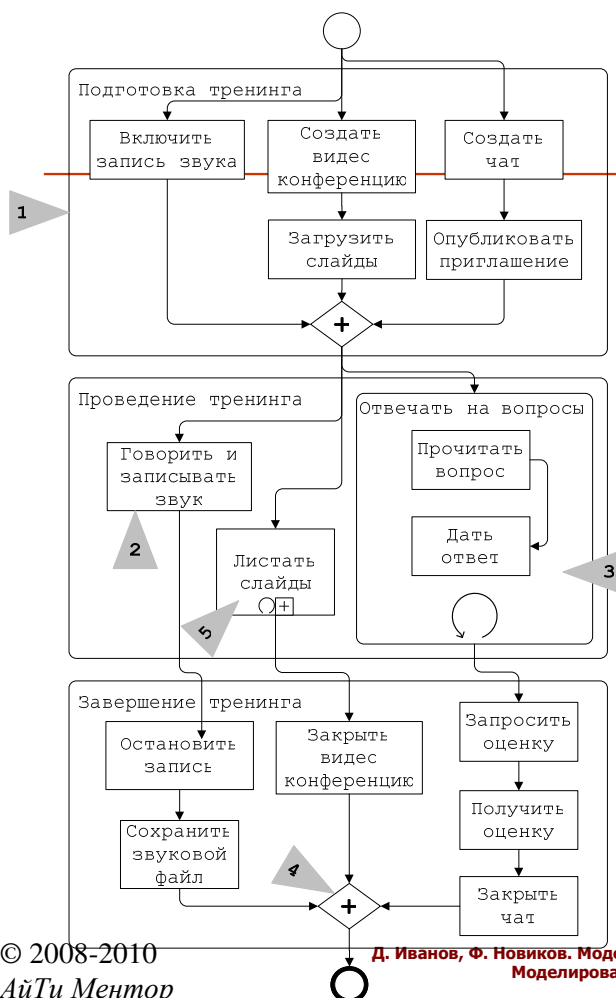


Диаграмма закрытого процесса BPMN 1.2

- 1) Блок
- 2) Элементарное действие
- 3) Неэлементарное раскрытое действие
- 4) Слияние управления
- 5) Неэлементарное нераскрытое действие

© 2008-2010
АйТи Ментор

Д. Иванов, Ф. Новиков. Моделирование на UML. Вторая ступень
Моделирование использования

26 из 81

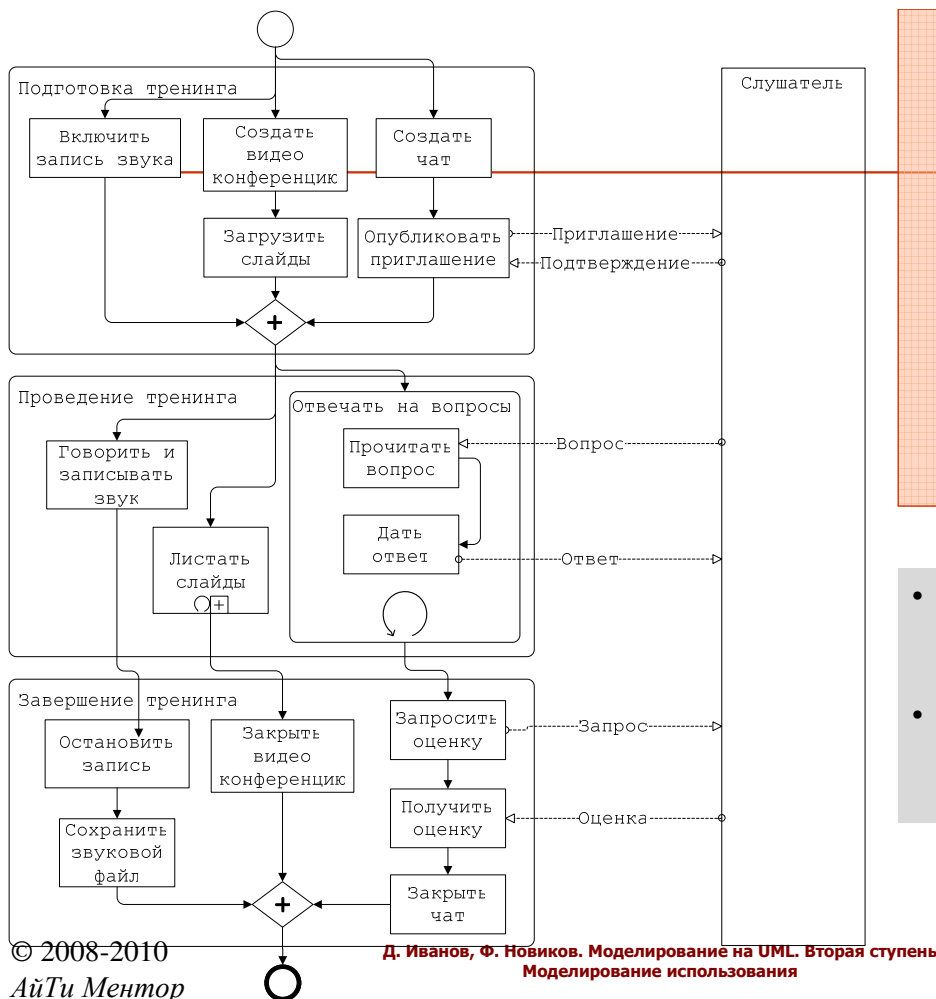
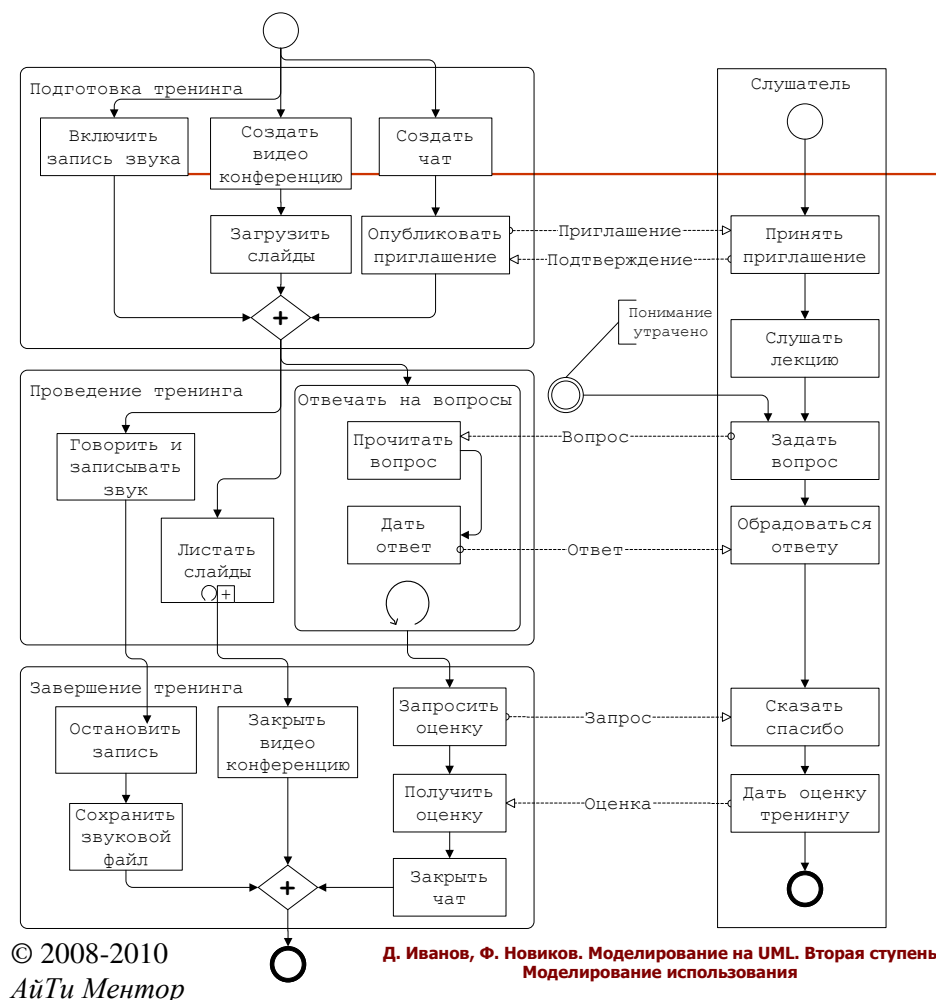


Диаграмма полу открытого процесса VRMN 1.2

- Сообщения (типизированные данные)
- Несколько взаимодействующих процессов

27 из 81



Диаграмм а открытого процесса VRMN 1.2

- Множество различных типов событий
- Множество структур управления

28 из 81

Связь моделирования использования с бизнес-анализом

- | | |
|---|---|
| • Моделирование использования | • Бизнес-анализ |
| • Варианты использования | • Функции организации |
| • ЧТО делает система | • ЧТО делает организация |
| • Сценарии | • Бизнес-процессы |
| • Последовательность событий и действий | • Последовательность событий и действий |
| • Действующие лица | • Поставщики и потребители |
| • Получают значимый результат | • Получают значимый результат |

2. Значение моделирования использования

- Диаграммы деятельности
= блок схемы
- Диаграммы состояний
= конечные автоматы
- Диаграммы классов
= диаграммы «сущность – связь»
- ... и так далее
- Диаграммы использования
 - = ... ???

Сквозной пример

Моделирование информационной системы отдела кадров

- Предметная область знакома всем
- Типичное офисное приложение
- Авторам случилось проектировать на самом деле



Техническое задание



Информационная система «Отдел кадров» (сокращенно ИС ОК) предназначена для ввода, хранения и обработки информации о сотрудниках и движении кадров. Система должна обеспечивать выполнение следующих основных функций.

1. Прием, перевод и увольнение сотрудников
2. Создание и ликвидация подразделений
3. Создание вакансий и сокращение должностей

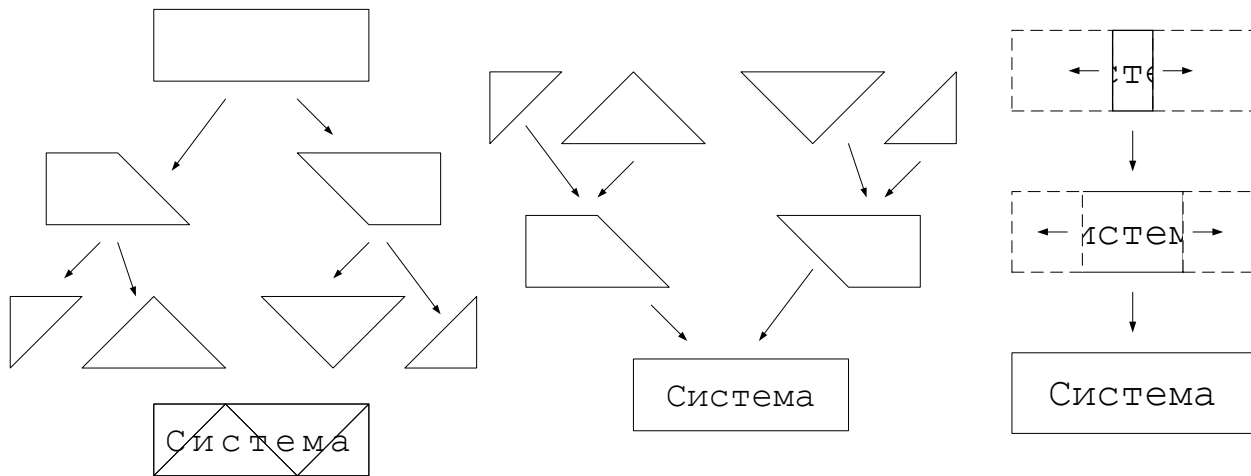
Подходы к моделированию

- **Структурный подход**: 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

Три направления разработки (i)

- Программирование **сверху вниз** = программирование без **Go To** методом пошагового уточнения
 - Проектирование первично, реализация вторична
 - Разбиение исходной задачи на очевидные подзадачи
- Программирование **снизу вверх**
 - Уровень языка программирования повышается, пока исходная задача не станет очевидной
- Программирование **вширь**
 - Начиная с самого первого шага, создается и на всех последующих шагах поддерживается работоспособная версия программы
 - Требуется дополнительных трудозатрат, но вызывает положительные эмоции!
- На практике всегда применяется их комбинация

Три направления разработки (ii)



Сверху вниз

Снизу вверх

Вширь

С.С. Лавров

Лавров Святослав Сергеевич
(1923–2004) – член-корреспондент
РАН, классик программирования

- Основоположник ракетно-космической баллистики в СССР
- Разработал первый отечественный транслятор Алгола
- Разработал программное обеспечение первых полетов человека в космос
- Ему принадлежит термин "программирование вширь"

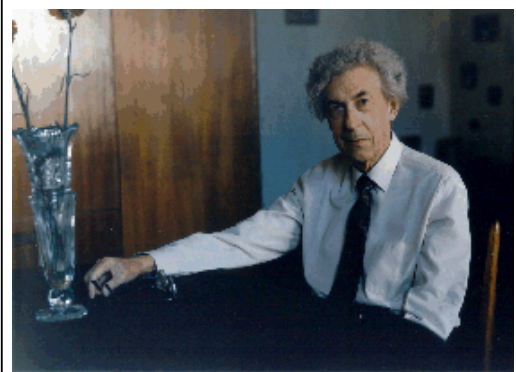
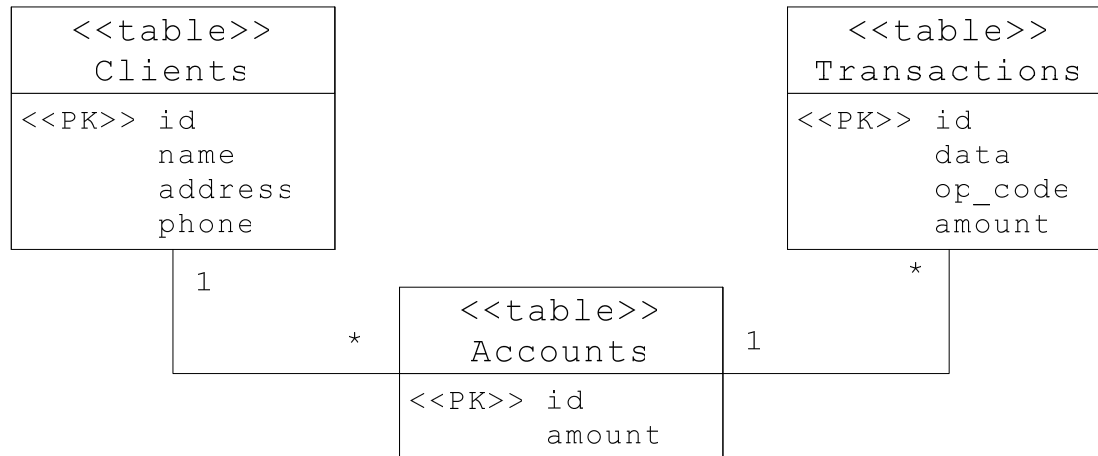


Схема базы данных

Описание данных и взаимосвязей между ними



ОО подход: словарь предметной области

Термин	Категория	Пояснение
Клиент	Внешняя сущность	Клиент банка
Счет	Постоянно хранимый объект	Счет клиента в банке
Остаток	Атрибут	Атрибут класса Счет, значение которого является текущей денежной суммой на счете
Номер	Атрибут	Атрибут класса Счет, значение которого однозначно идентифицирует экземпляр класса Счет

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 взаимодействующих элементов

Преимущества моделирования использования

- Простые утверждения
 - Субъекты, предикаты (и объекты)
- Абстрагирование от реализации
 - **ЧТО** делает система (но не **КАК** или **ЗАЧЕМ**)
- Декларативное описание
 - но не императивное
 - нет возможности указать, какая функция “раньше”, а какая “позже”
- Выявление границ
 - но не черный ящик

3. Диаграммы использования

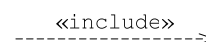
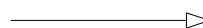
- **ОЧЕНЬ** простые идеи и нотация
 - Применяется на ВСЕХ фазах (анализ, ..., тестирование)
 - Понимают ВСЕ (разработчики, заказчики, управленцы) одинаково
- **НЕ** зависит от остальных средств UML
 - Изобретены Иваром Якобсоном в 1986 г.
 - Не меняется 1.1 → 2.2
 - Может использоваться отдельно

Элементы диаграмм использования

- Сущности:
 - действующие лица
 - варианты использования
 - комментарии

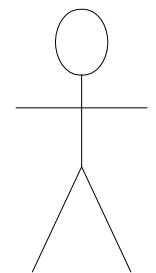


- Отношения:
 - ассоциация
 - обобщение
 - зависимости



Почему actor = “действующее лицо”?

- Альтернативные варианты: **актёр** (неточно), **актор** (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 человек” означает Гамлета, но не Смоктуновского!



Actor

Почему use case = “вариант использования”?

- Альтернативные варианты: **сценарий** (неточно), **элемент Use Case** (нет такого слова), **прецедент** (абсолютно неверно)
- Ивар Якобсон (швед) использовал термины **usage case** и **usage scenario**
 - Американские пользователи упростили
 - Английские юристы издавна использовали понятие
 - Use Case (прецедент) – принятие судебного решения на основании другого судебного решения
- Вариант использования – это функция, но не прецедент!



Make Order

Действующие лица и их идентификация

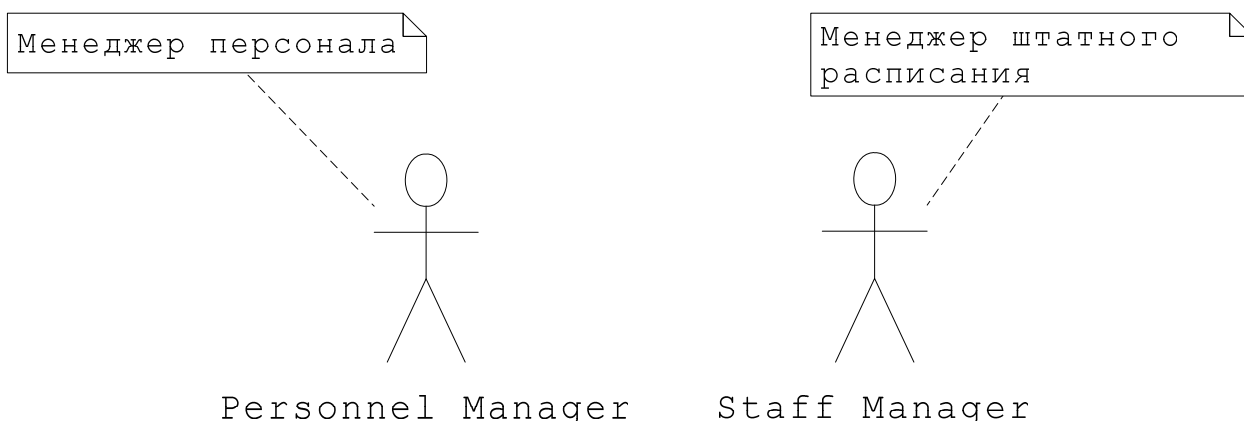
Действующее лицо — это множество логически взаимосвязанных **ролей**

- Стереотипный **класс**
- Находятся ВНЕ проектируемой системы
- **Роль** в UML — это **контракт** (сервисы), поддерживаемый данным классификатором в данной ассоциации
- Типовые случаи: категории пользователей, внешние программные и аппаратные средства
- Выделение категорий пользователей :
 - пользователи участвуют в *разных* бизнес-процессах
 - пользователи имеют *различные* права
 - пользователи взаимодействуют с системой в *разных* режимах: от случая к случаю, регулярно, постоянно

Действующие лица ИС ОК



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



Варианты использования

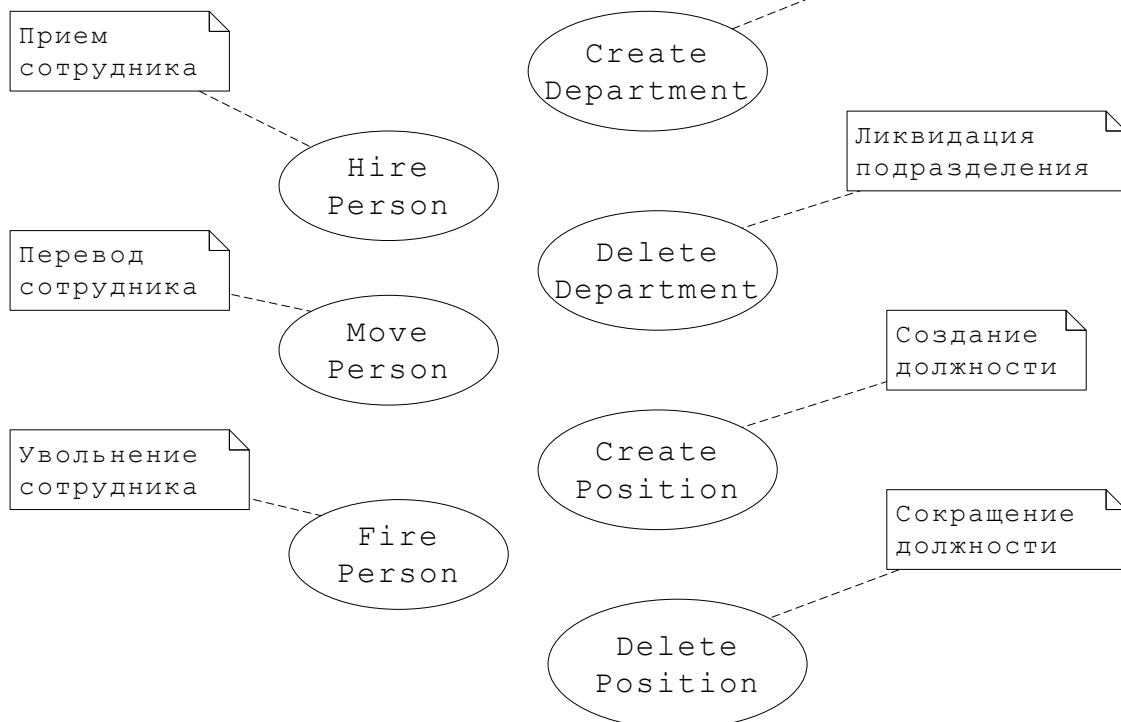
Вариант использования – множество возможных последовательностей событий/действий (**сценариев**), приводящих к значимому для действующего лица результату

- Типичные случаи: пункты **T3**
- Если T3 смутное, его можно (и нужно!) попробовать переписать фразами
субъект – предикат – объект

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



Дисциплина имен

- translit – выглядит ужасно
- English – нужно чтобы ВСЕ знали одинаково
- Выход: жаргон из слов и сокращений
- Имена действующих лиц — существительные
- Имена вариантов использования — глаголы (или отглагольные существительные) *с прямым дополнением*

Комментарии

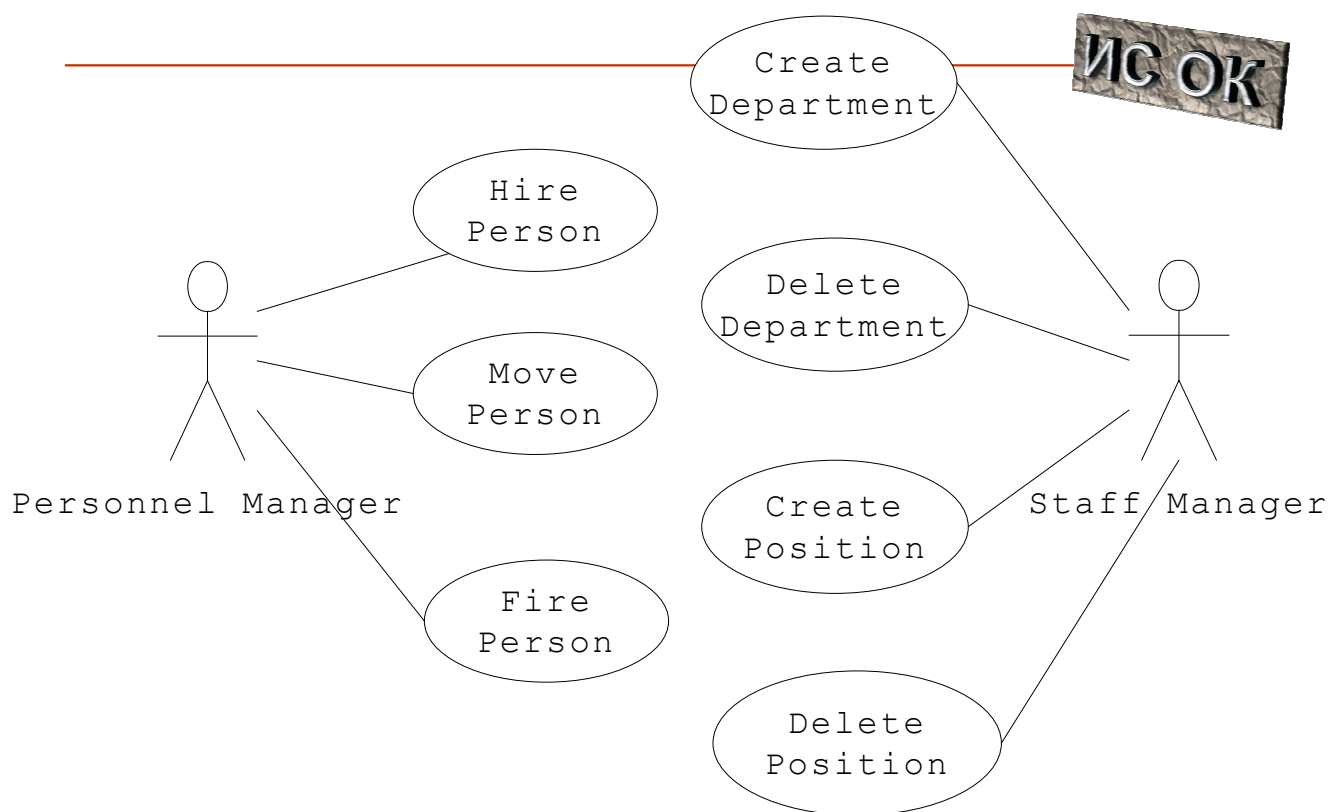
<<requirement>>

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

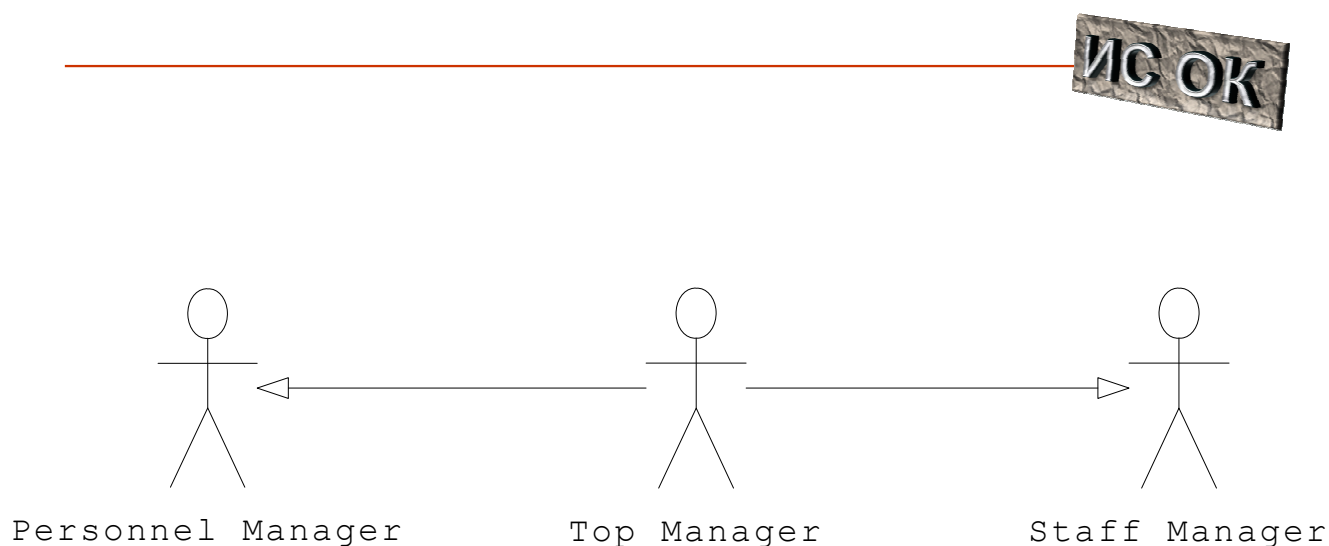
- Важно и нужно!
- Позволяют хранить ЛЮБУЮ информацию (в т.ч. необрабатываемую и внешнюю)
- Могут иметь **стереотипы**:
 - «requirement» — описывает общее требование к системе
 - «responsibility» — описывает ответственность сущности (классификатора)

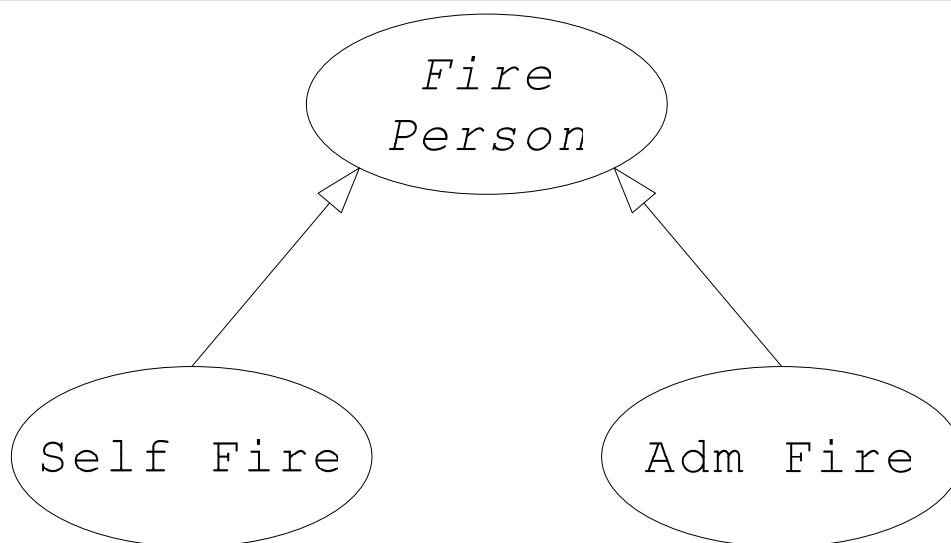
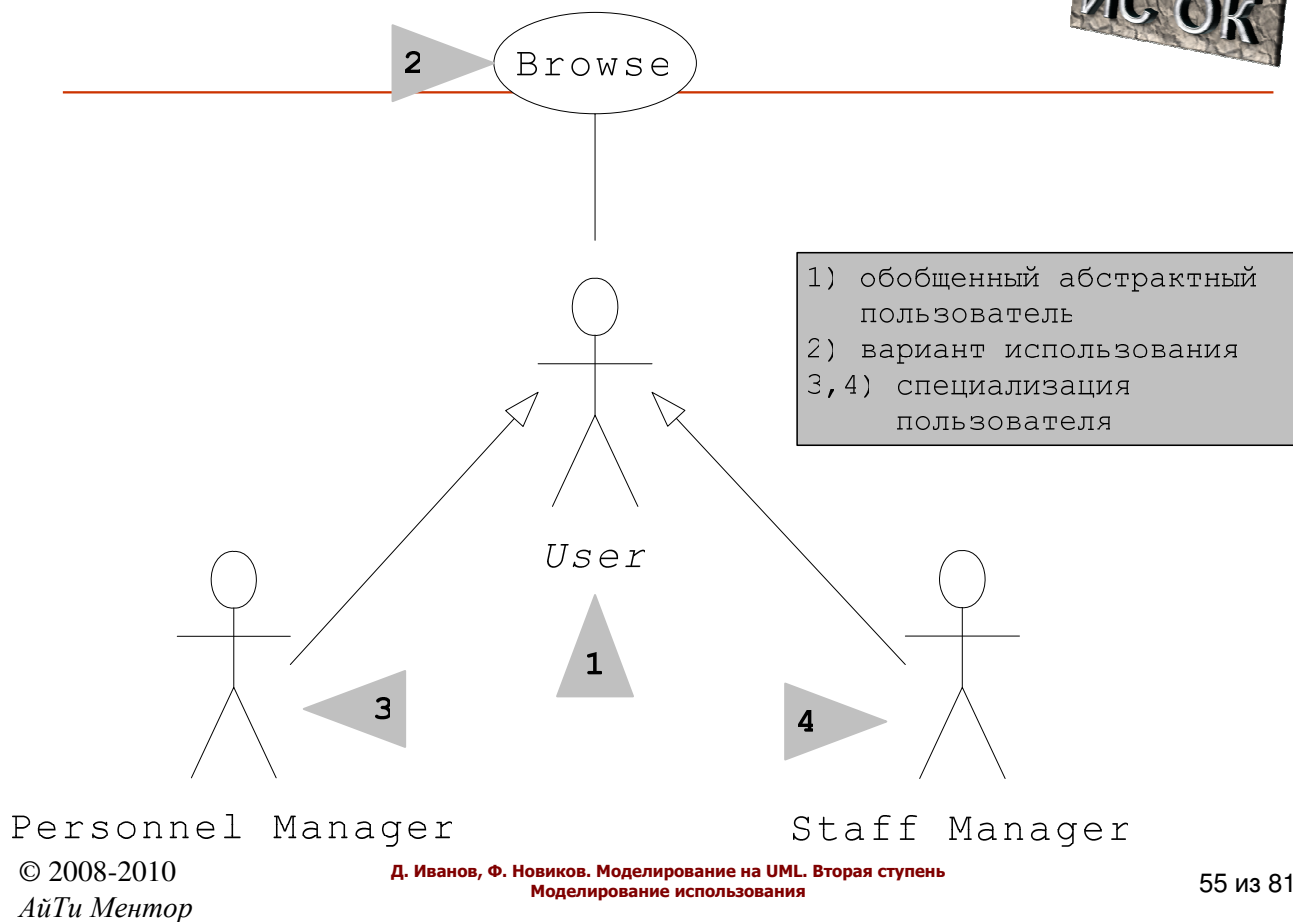
Отношения между элементами диаграммы использования

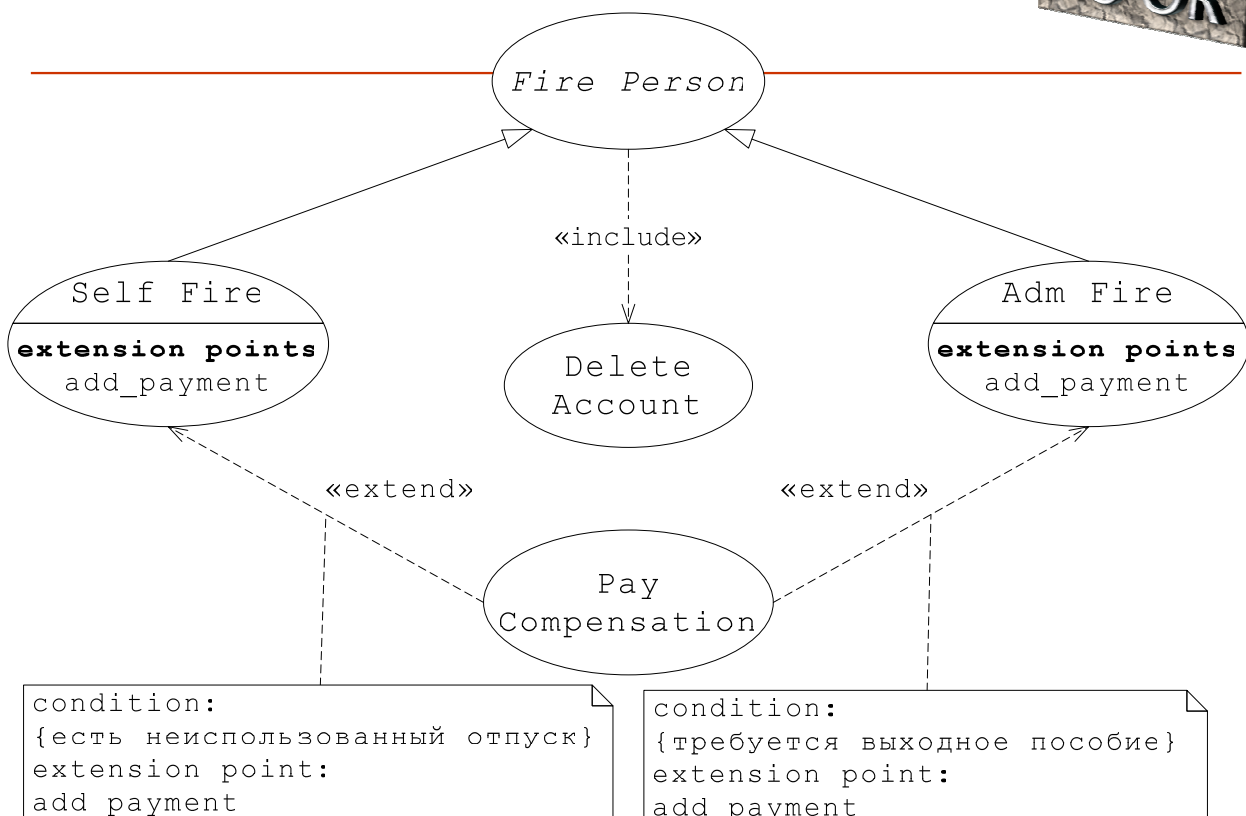
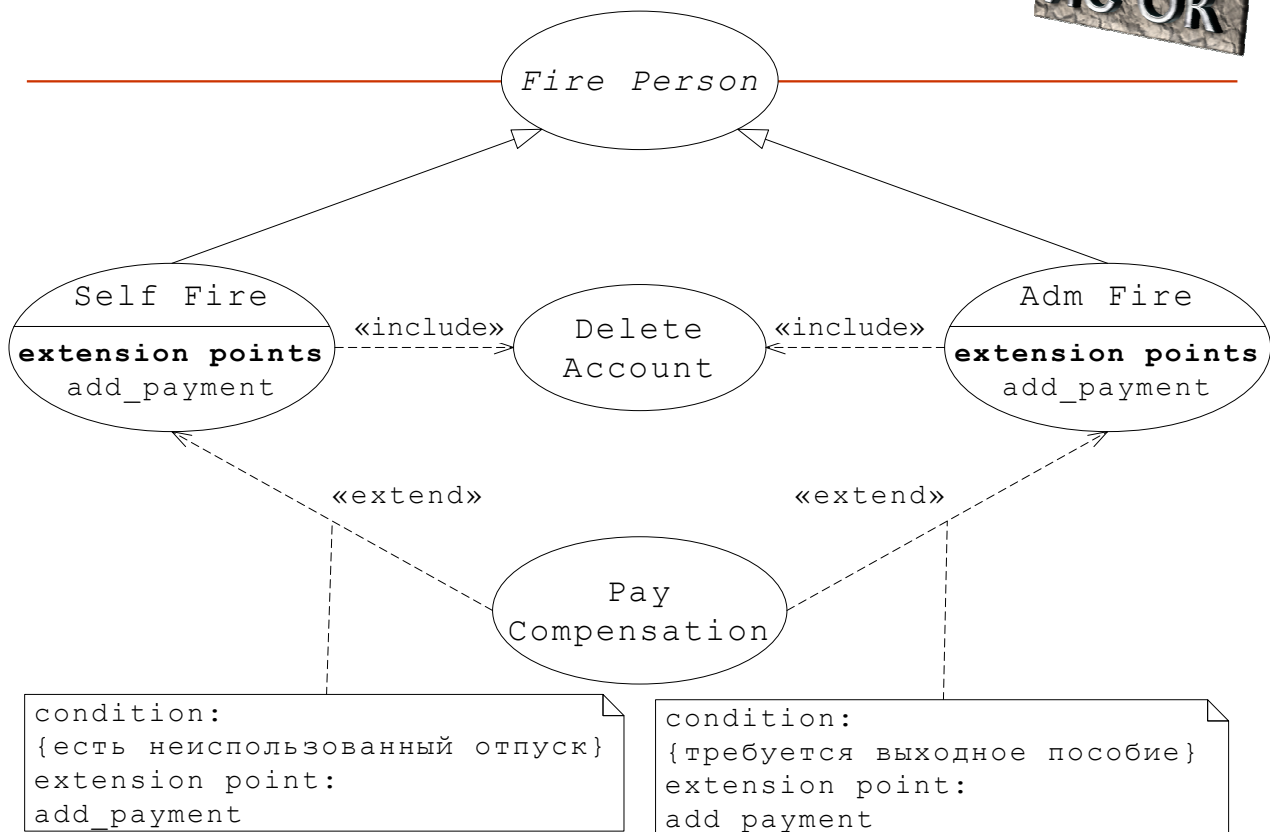
- Ассоциация между действующим лицом и вариантом использования
- Обобщение между действующими лицами
- Обобщение между вариантами использования
- Зависимость между вариантами использования



Иерархия категорий пользователей ИС ОК (обобщение)

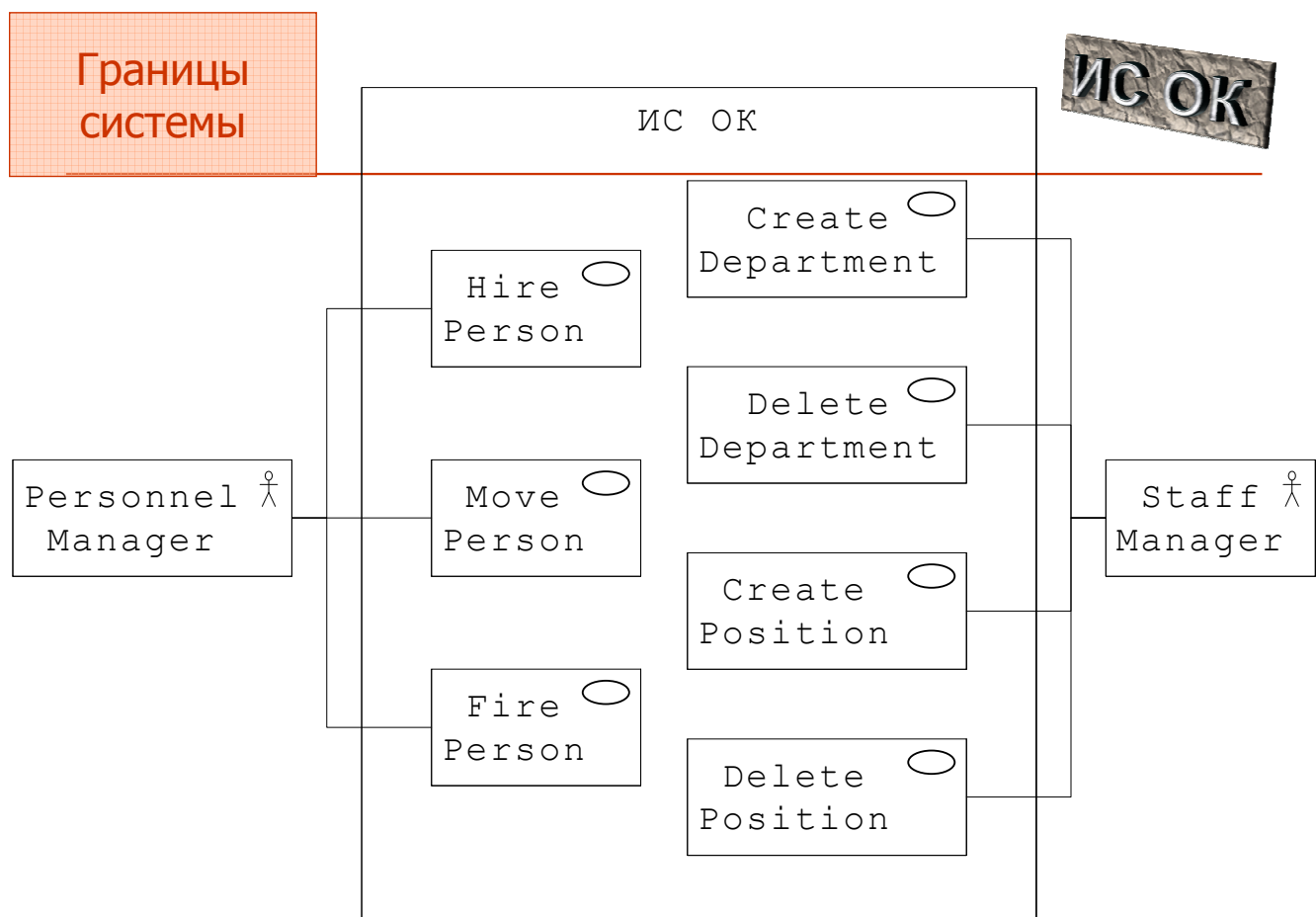






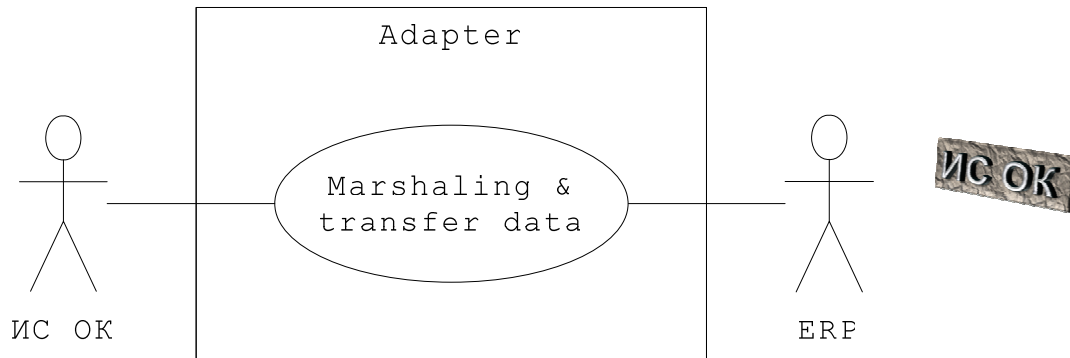
Границы системы

- Модель использования:
 - **внутренняя** моделируемая система — варианты использования
 - **внешнее** окружение — действующие лица
 - связь между моделируемой системой и внешним окружением
- Если нужно отделить применяется *графический комментарий* – **границы системы (subject)**



Применение моделей использования

- В качестве действующих лиц и вариантов использования могут выступать **любые сущности**, для которых можно определить функциональные или не функциональные **требования**



- Пример: основная система (ИС ОК) как действующее лицо для подсистемы (Adapter)

4. Выявление и анализ требований

Фаза анализа требований – первая и самая важная:

Если программисты знают **ЧТО** нужно сделать – они это сделают

Если требования не определены – успеха **НЕ** будет

Что такое требования к ПО (i)

1. Простое решение. Понятие «требование» не определяется, полагаются на здравый смысл

2. Обычное решение. Дать **какое-нибудь** определение

- Например: *Требования – это высокоуровневые обобщенные утверждения о функциональных возможностях и ограничениях системы*

Риск: разные бизнес-цели, опыт и квалификация у заказчика и разработчика → ошибки в определении требований

Что такое требования к ПО (ii)

3. Стандартное решение

- Стандарты IEEE:
 - "требование" апеллирует либо к пользователям, либо к формальным документам
- Российский стандарт ГОСТ 12207:
 - "требование" определяется перечислением всех видов требований

4. Сложное решение

- Проводится формализация предметной области и требования формулируются как формальные математически строгие утверждения – **очень точно, но очень дорого**

Что такое требования к ПО (iii)

5. Компромиссное решение

- Использовать все приёмы "по месту"
- Применять UML:
 - Математическая модель – с переводом в язык OCL
 - Функциональные требования – **модель ИСПОЛЬЗОВАНИЯ**
 - Псевдо требование «Сделайте мне красиво» – можно, напечатав крупным шрифтом, повесить на стену в комнате разработчиков

Источники требований

- **Бизнес-требования** (требования заказчиков) – определяются целями проекта
- **Пользовательские требования** (требования пользователей) – что они смогут делать с помощью системы
- **Системные требования** (требования разработчиков) – определяют характеристики системы

Виды требований

- **Функциональные требования** – что должна делать система (и чего она **не** должна делать)
- **Требования к качеству** – качественные характеристики системы
- **Параметрические требования** – количественные характеристики системы
- **Требования к модели** – характеристики модели (а не моделируемой системы)
- **Специальные требования** – не зависящие от системы требования, определяющиеся законами, стандартами и т.д.

Пример: требования к системе отсылки сообщений (sms)

	Бизнес-требования	Пользовательские требования	Системные требования
Функциональные требования	Ввод текста сообщения	Подбор слов по словарю	Словарь заданного языка
Требования к качеству	Nokia, Siemens, Alcatel		Независимый программный интерфейс
Параметрические требования	Длина сообщения ограничена объемом имеющейся памяти		Доступ к специальным интерфейсам
Требования к модели			Повторное использование
Специальные требования	Государственный язык страны продаж	Запрещена ненормативная лексика	Словарь ненормативной лексики

Применение UML

Модель использования UML, дополненная естественным языком и формальным языком типа OCL, – наилучшее средство выявления, анализа и документирования требований

- Функциональные требования – модель использования UML
- Требования к качеству и параметрические требования – примечания UML
- Удовлетворение требований к модели напрямую связано с квалификацией архитекторов

5. Реализация вариантов использования

- Реализация варианта использования — это описание всех или некоторых **сценариев**, составляющих вариант использования
- Метод описания — указание **алгоритма**
 - Текстовые сценарии Программы на псевдокоде
 - Диаграммы деятельности Диаграммы взаимодействия
- Переход от анализа и постановки задачи к решению (проектированию)

Текстовые описания

Увольнение по собственному желанию

1. Сотрудник пишет заявление
2. Начальник подписывает заявление *(а если нет?..)*
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

Минус: неточность, которая незаметна

Программы на псевдокоде

Use case Self Fire

Получить заявление

add_payment:

Pay Compensation

(Self Fire, add_payment)

Include Delete Account

Обновить информацию в
базе данных

Use case Adm Fire

Получить приказ

add_payment:

Pay Compensation

(Adm Fire, add_payment)

Include Delete Account

Обновить информацию в
базе данных

Use case Pay Compensation

if (from Self Fire)

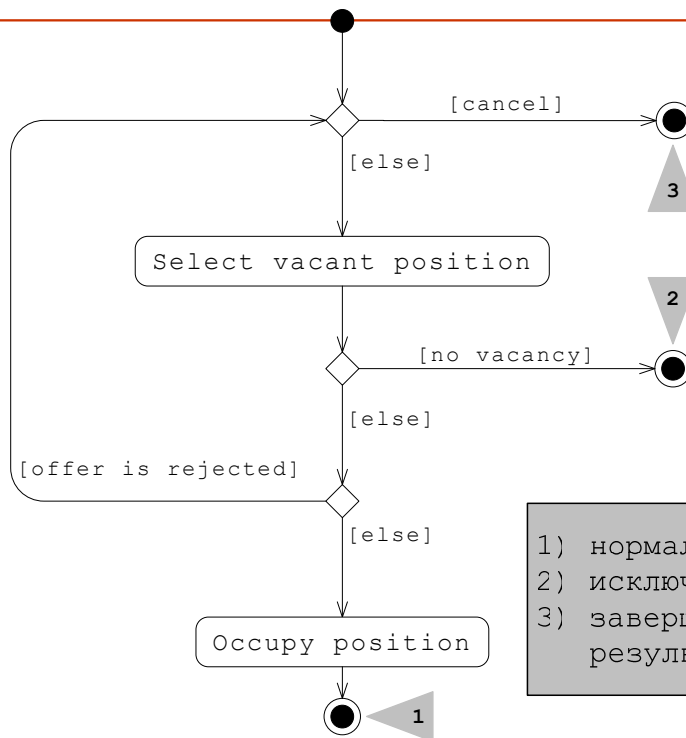
начислить за неиспользованный отпуск

else if (from Adm Fire)

начислить выходное пособие

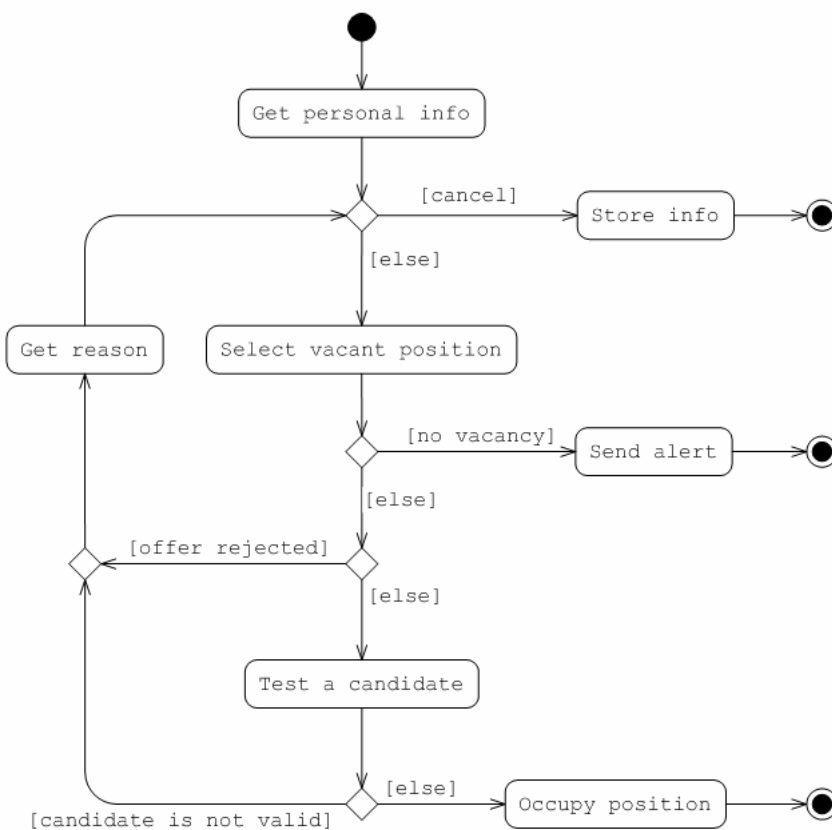
Минус: не подлежит повторному
использованию

activity Прием сотрудника на работу (Hire Person)



- 1) нормальное завершение
- 2) исключительная ситуация
- 3) завершение без видимого результата

activity Прием сотрудника на работу. версия 2 (Hire Person)



Усовершенствование
реализации

Реализация диаграммами взаимодействия

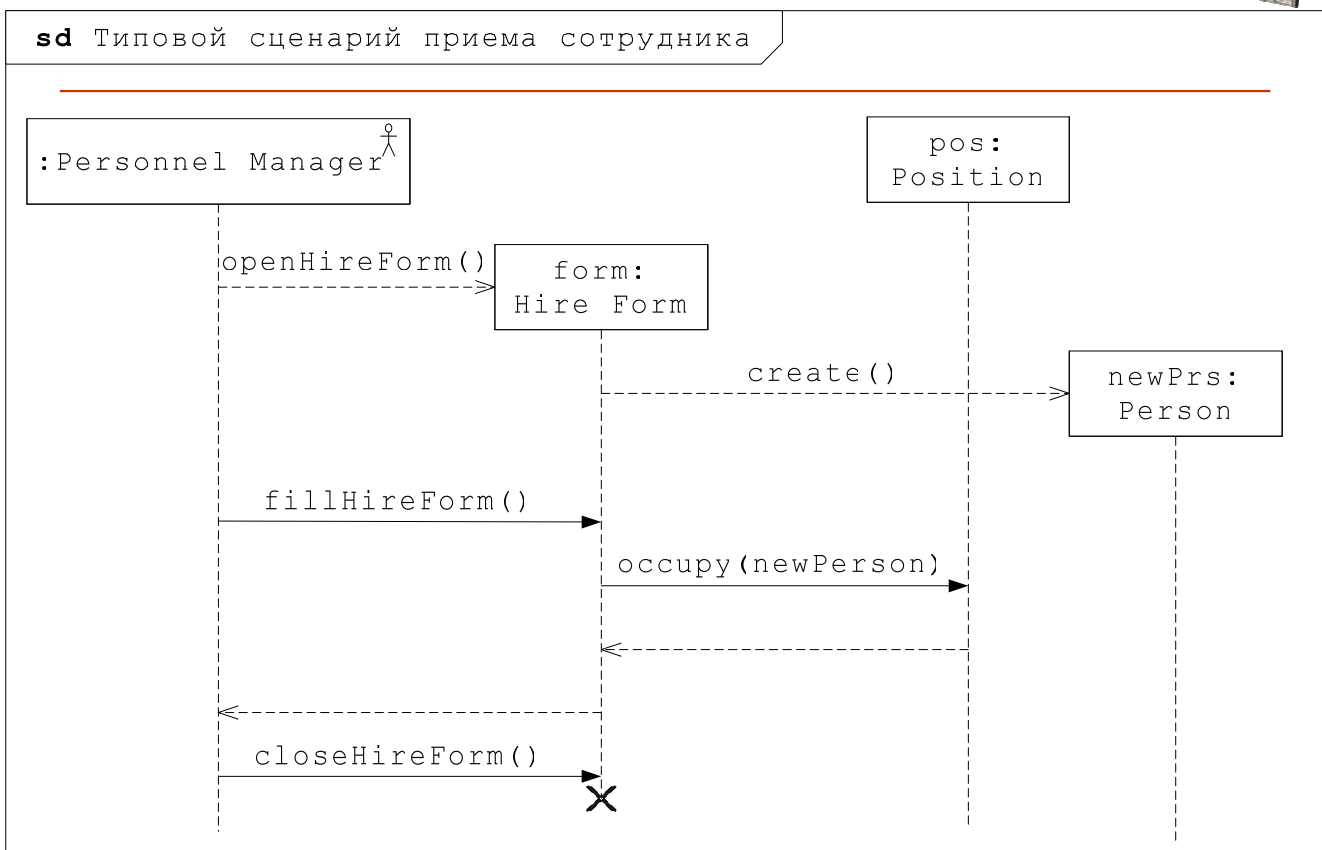
- Прямо ведет к объектной модели
- Целостность проектируемой системы
- Возможность автоматизированного построения прототипов

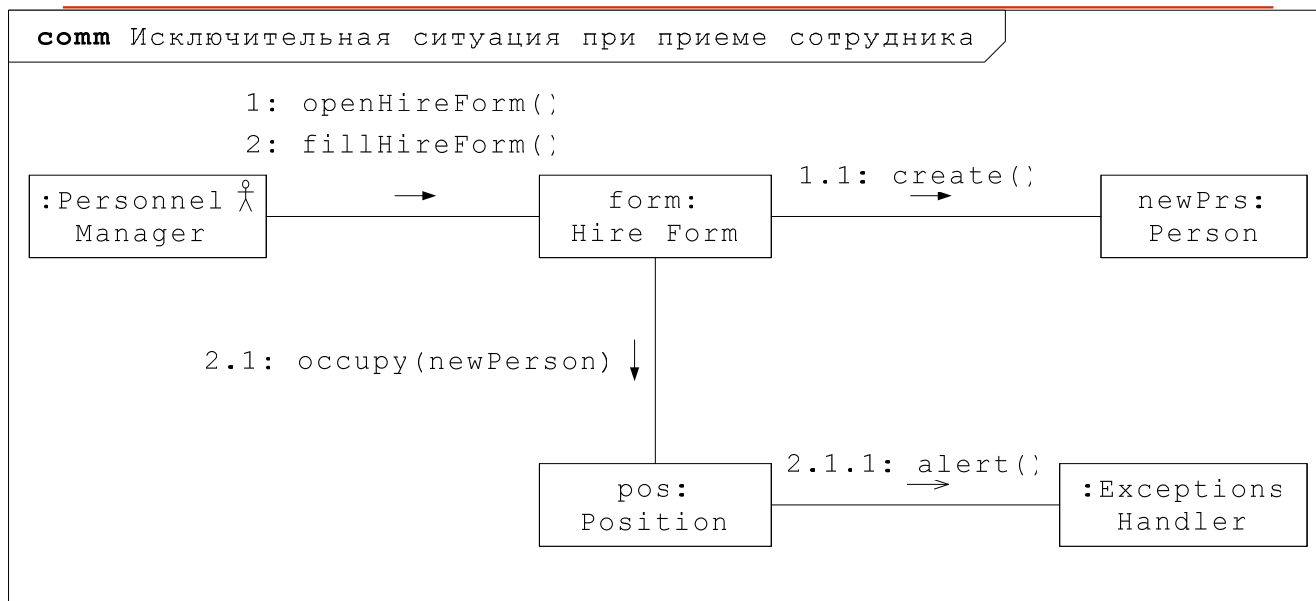
НО

- Позволяют реализовать только **ОДИН** сценарий – нужно **МНОГО** диаграмм
- Сложная и непривычная нотация

Диаграмма последовательности для типового сценария

ИС ОК





необходимость включения еще одного класса

Сравнение способов реализации вариантов использования (i)

- Текстовые описания
 - Всем понятно, привычно и удобно
 - Должно 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 (ii)

- 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

Полезные советы

- Строить диаграммы взаимодействия для всех **ТИПОВЫХ** сценариев вариантов использования
 - Хотя бы для всех вариантов использования, выбранных для реализации в первой версии системы
- Сопоставлять набор выявленных классов со словарем предметной области
 - Если пересечение незначительно, то нужно приостановить проектирование и вернуться на фазу анализа



6. Выводы

- Составление диаграмм использования — первый шаг моделирования
- Основное назначение — показать, что делает система во внешнем мире
- Не зависит от программной реализации системы (от структуры классов, модулей и компонентов)
- Идентификация действующих лиц и вариантов использования — ключ к успеху
- Способ реализации — дело вкуса