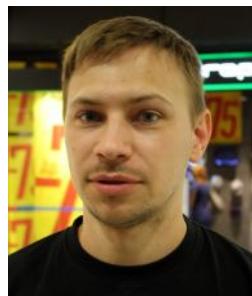


Моделирование на UML. Вторая ступень

Моделирование поведения



Иванов Д.Ю., Новиков Ф.А.



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. Модели поведения
2. Диаграмма автомата
3. Диаграмма деятельности
4. Диаграммы взаимодействия
5. Моделирование параллелизма
6. Выводы

1. Модели поведения

1.1 Конечные автоматы

1.2 Почему автоматы интересны?

1.3 Сети Петри

1.4 Средства моделирования поведения

Модели поведения

- Модель поведения (behavior model): описание алгоритма работы системы — как работает система?
- Модель поведения должна быть:
 - детальной
 - компактной и обозримой
 - не должна зависеть от особенностей реализации
 - средства должны быть знакомы и привычны большинству пользователей
- *Модели поведения в UML кооптированы*

Конечные автоматы

Конечный автомат (Мили)

- Входной алфавит: $A = \{ a_1, \dots, a_n \}$
- Алфавит состояний: $Q = \{ q_1, \dots, q_m \}$
- Выходной алфавит: $B = \{ b_1, \dots, b_k \}$
- Функция переходов: $\delta : A \times Q \rightarrow Q$
- Функция выходов: $\lambda : A \times Q \rightarrow B$

Дополнительные соглашения

- Автоматное поведение: последовательность пар «стимул — реакция»
- Инициальный автомат — с выделенным начальным состоянием
- Автомат Мура — функция **пометок** зависит только от состояния $\mu : Q \rightarrow B$
- Заключительные состояния

Модификации конечных автоматов

1. Распознаватель:

- Допускающие состояния

2. Сети взаимодействующих конечных автоматов:

- Параллельные — общий вход
- Последовательные — выход одного автомата является входом другого
- Петля обратной связи — выход одного автомата является входом этого же автомата

3. Недетерминированный автомат:

$$\delta : A \times Q \rightarrow 2^Q$$

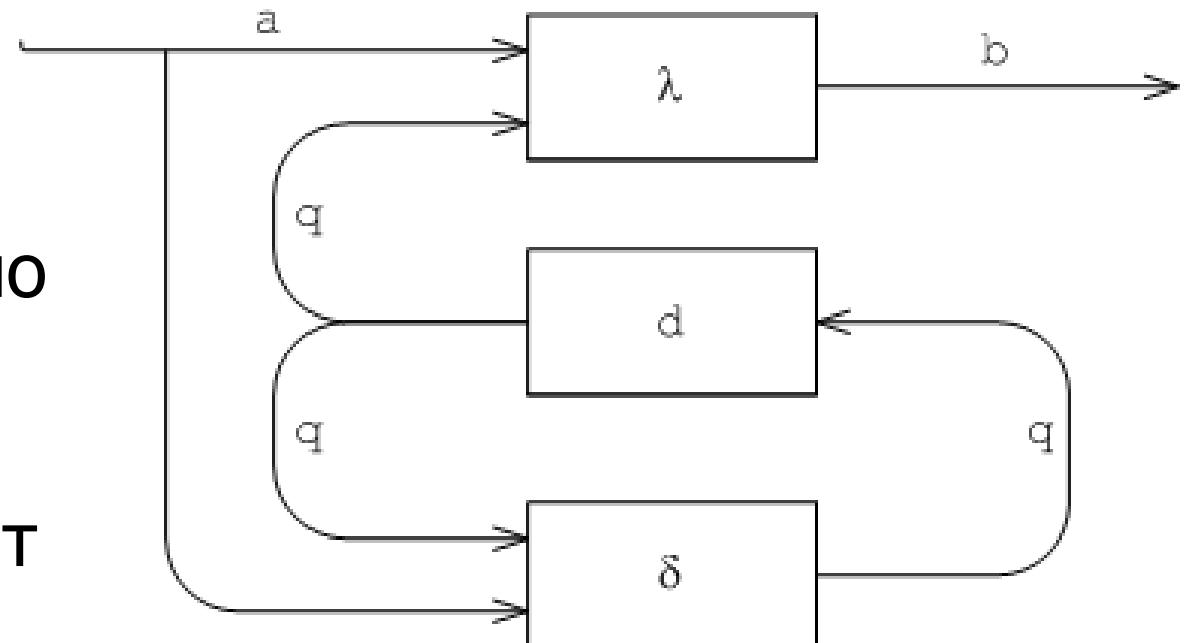
- Переходы (выходы) — многозначные отношения

Сеть интерпретатора конечного автомата

Любой конечный автомат — это сеть:

- комбинационные автоматы
- задержки

И наоборот: для любой сети можно построить эквивалентный конечный автомат



Почему автоматы интересны? (i)

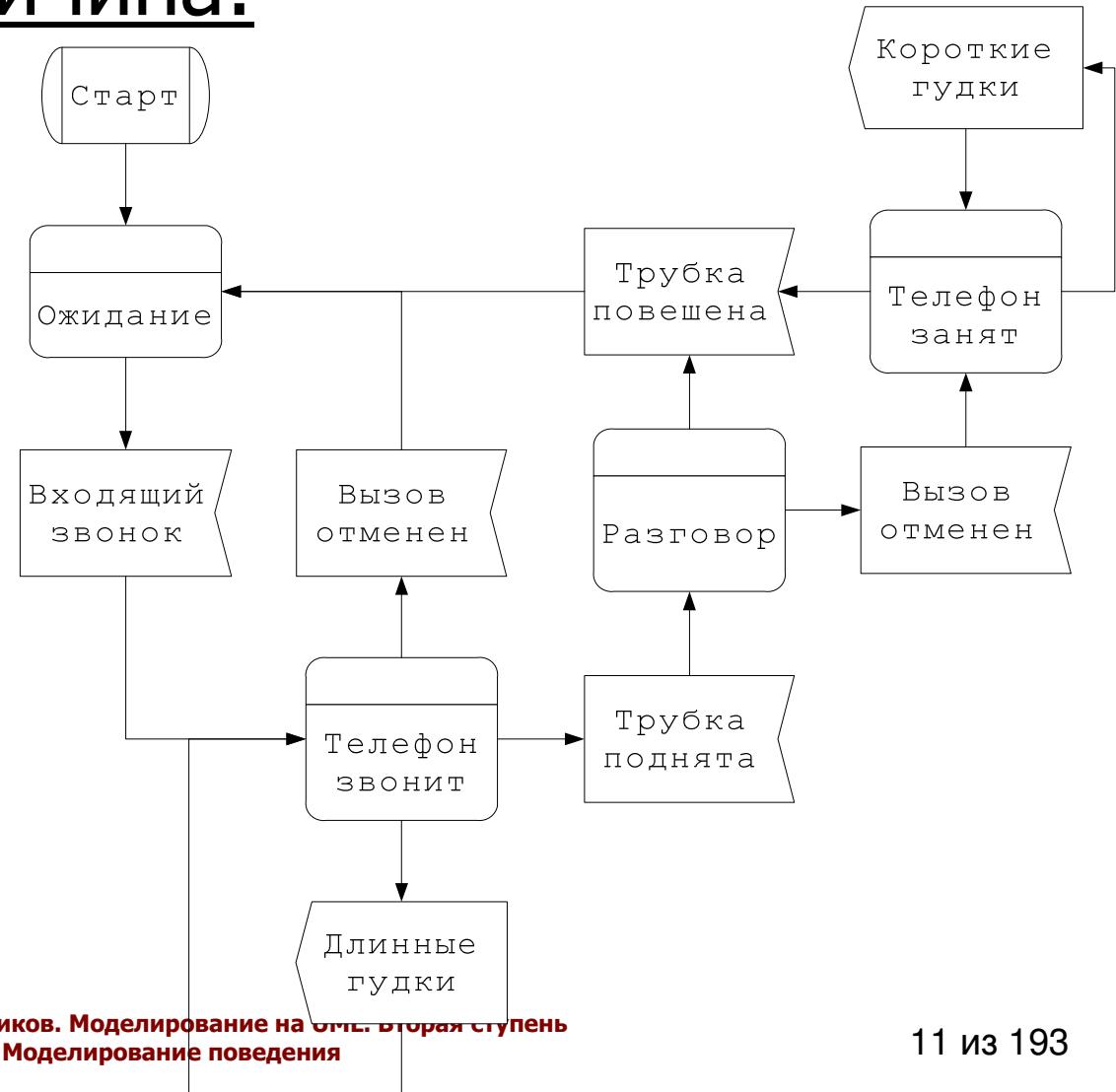
Теоретическая причина:

- Теорема Райса: все нетривиальные свойства вычислимых функций алгоритмически неразрешимы
 - проблема эквивалентности
 - проблема остановки ...
- Известны частные случаи (**автоматы**), для которых есть **эффективные** разрешающие алгоритмы

Почему автоматы интересны? (ii)

Историческая причина:

- Язык SDL
 - Z100, ITU-T
- Разработаны нотации:
 - таблицы
 - диаграммы



Нотации конечных автоматов (i)

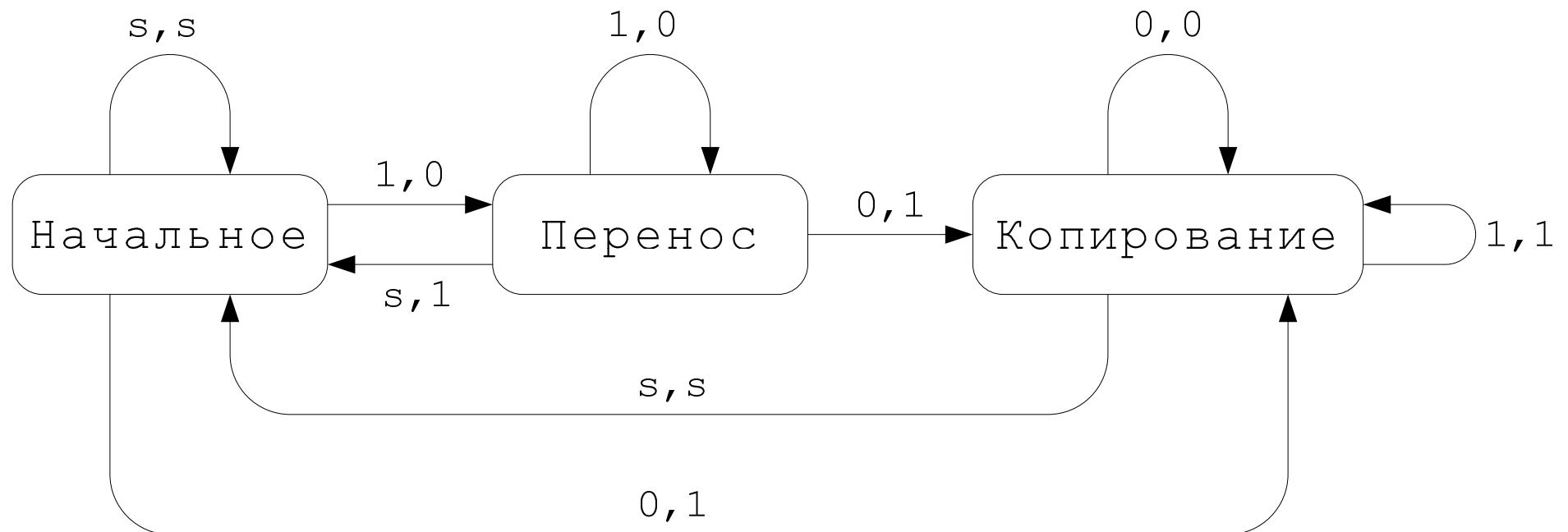
- Таблица конечного автомата

	0	1	S
начальное	копирование, 1	Перенос, 0	начальное, S
перенос	копирование, 1	Перенос, 0	начальное, 1
копирование	копирование, 0	копирование, 1	начальное, S

- Разряды поступают, начиная с младшего x++

Нотации конечных автоматов (ii)

- Диаграмма состояний-переходов



Почему автоматы интересны? (iii)

Практическая причина:

- **Процедура реакции** — функция переходов с побочным эффектом — алгоритмическая полнота
- Легко программируются:

```
state := 1
while state ≠ k
    stimulus := get()
    стимулы
    switch state
        case 1
            switch stimulus
                case A
                    . . .
                . . .
            . . .
        . . .
    . . .
```

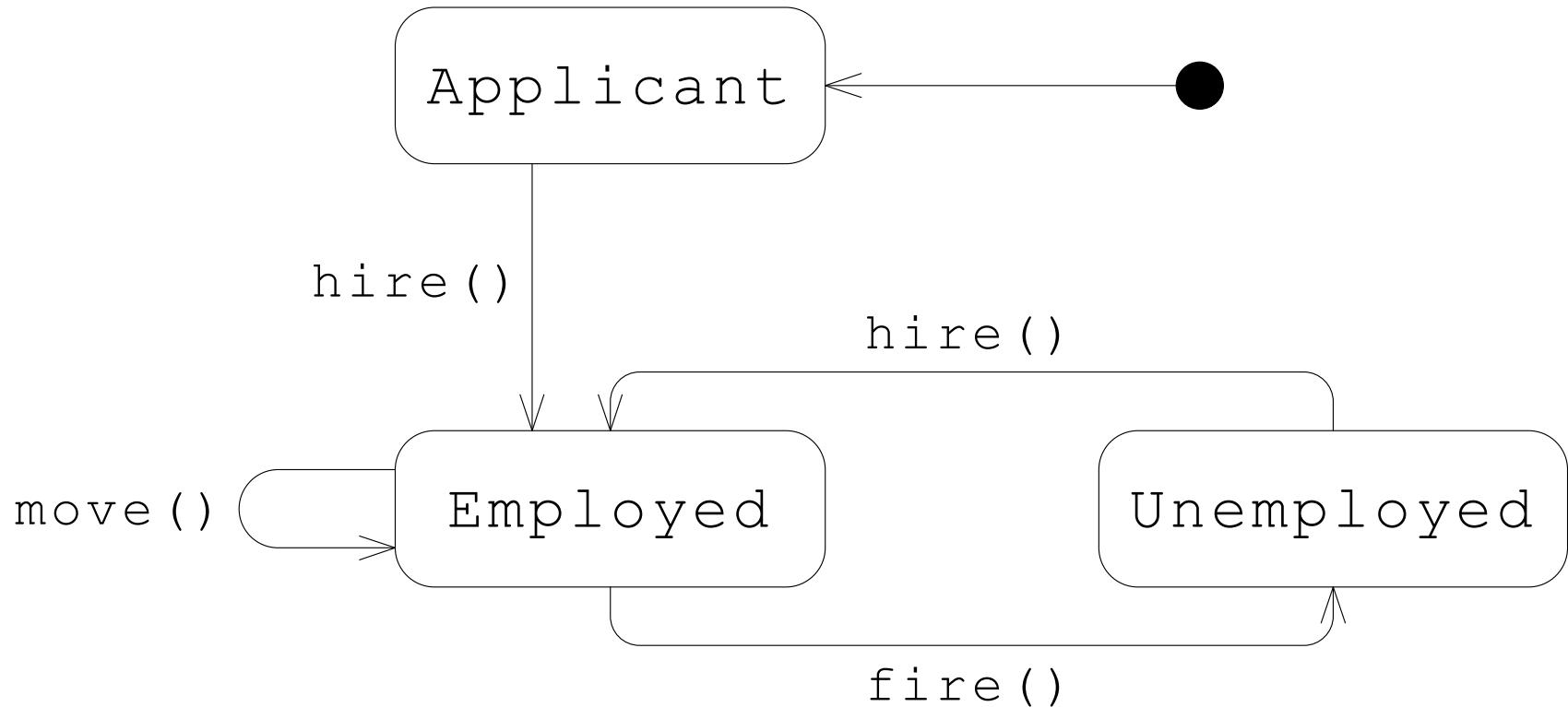
1 — начальное состояние
k — заключительное
A, B, ... — входные

Жизненный цикл сотрудника (i)

	Прием <i>hire</i>	Перевод <i>move</i>	Увольнение <i>fire</i>
Кандидат <i>Applicant</i>	Принять(), В штате	Ошибка(), Кандидат	Ошибка(), Кандидат
В штате <i>Employed</i>	Ошибка(), В штате	Перевести(), В штате	Уволить(), Уволен
Уволен <i>Unemployed</i>	Принять(), В штате	Ошибка(), Уволен	Ошибка(), Уволен

В каждой ячейке процедура реакции и новое состояние

Жизненный цикл сотрудника (ii)



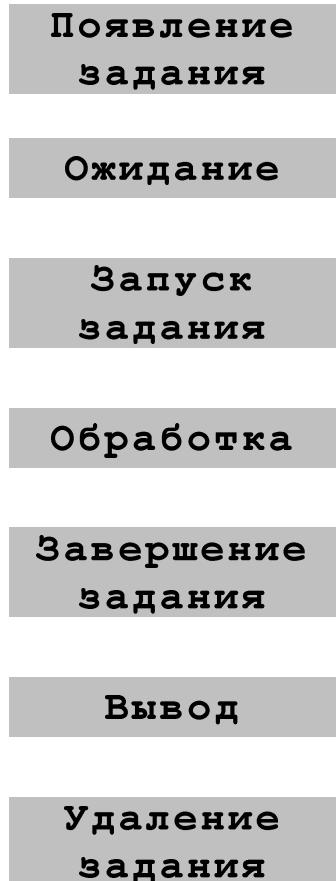
Не все реакции определены

Сети Петри

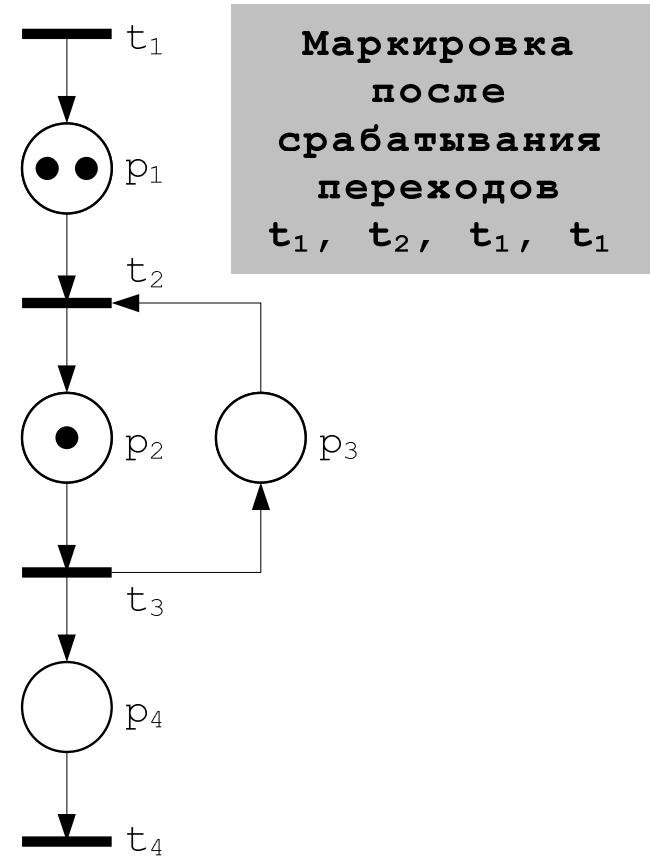
Сеть Петри — ориентированный двудольный граф:

- Вершины — **позиции и переходы**
- **Маркировка** — сопоставляет каждой позиции неотрицательное целое число
- Переход имеет **входные и выходные** позиции
- **Разрешенный** переход — все его входные позиции не пусты (> 0)
- Любой разрешенный переход может **сработать** — меняются маркировки:
все входящие --, все исходящие ++

Сеть Петри



Процесс свободен



Средства моделирования поведения в UML (i)

1. Явное выделение состояний

- Жизненный цикл (lifecycle): текущее поведение объекта определяется его историей
 - диаграммы автомата

2. Поток управления и поток данных

- Поток управления: последовательность выполнения операторов
- Поток данных: связь выходов со входами
 - диаграммы деятельности

Средства моделирования поведения в UML (ii)

3. Последовательность сообщений

- Диаграммы взаимодействия
(диаграммы коммуникации и диаграммы последовательности, диаграммы синхронизации, обзорные диаграммы взаимодействия)

4. Параллельное поведение

- Специальные конструкции на диаграммах автомата, деятельности, взаимодействия

1. Диаграммы автомата

= Диаграммы состояний в UML 1

- Сущности — состояния
 - простые (simple), составные (composite)
 - специальные (pseudo), ссылочные (submachine)
- Отношения — переходы: простые / составные
 - исходное состояние (source), событие перехода (trigger event), сторожевое условие (guard), действие на переходе (effect), целевое состояние (target)

Совокупность состояний и переходов между ними
образует **конечный автомат**

Дэвид Харел

Дэвид Харел, (David Harel) (р. 1950) — профессор информатики в Вейсманновском институте в Израиле

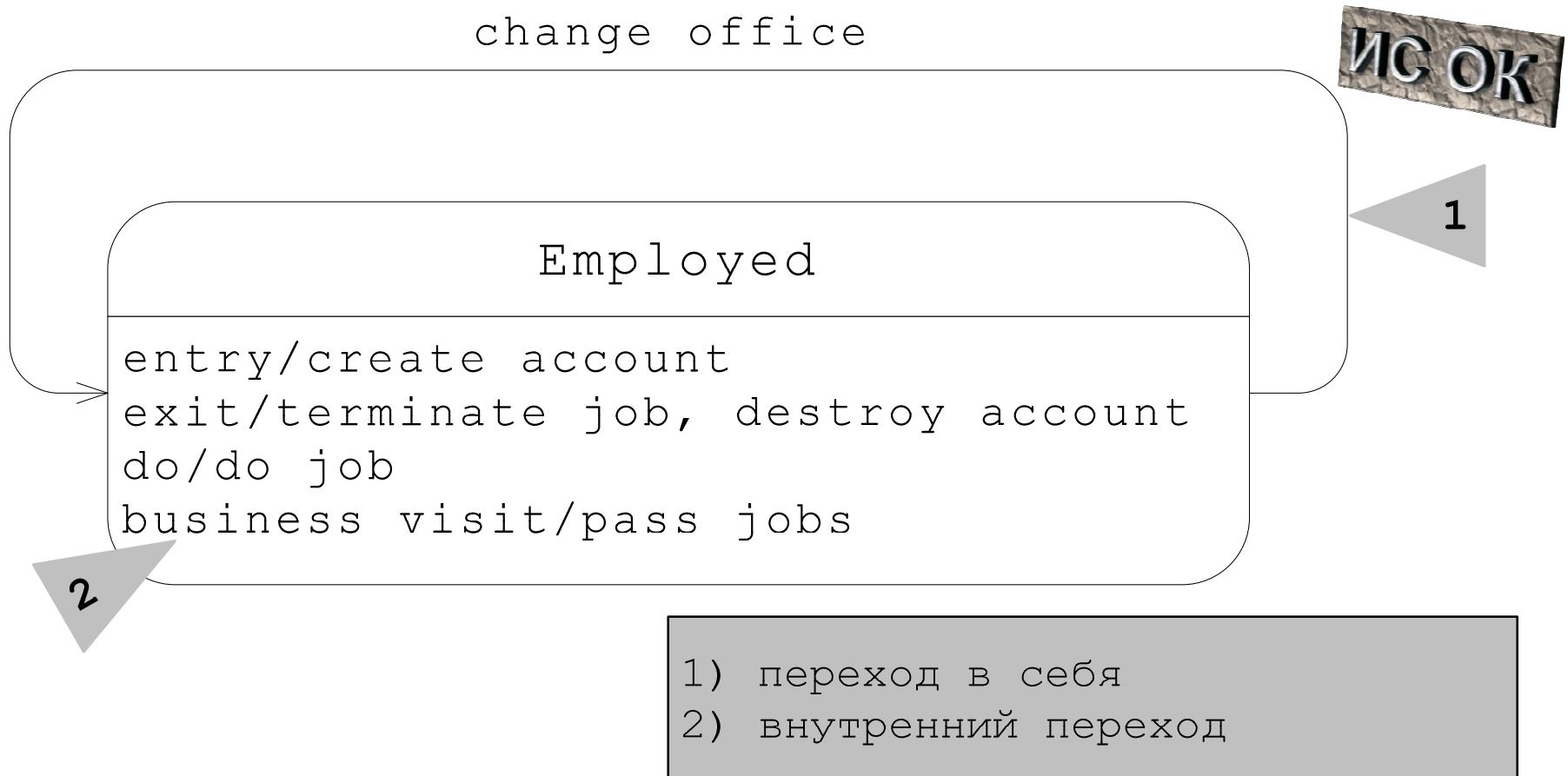
- Динамическая логика, теория вычислимости, технологии программирования
- В 1984 году предложил диаграммы состояний (statecharts), которые вошли в язык UML
- Лауреат ACM Software System Award в 2007 году



Простое состояние

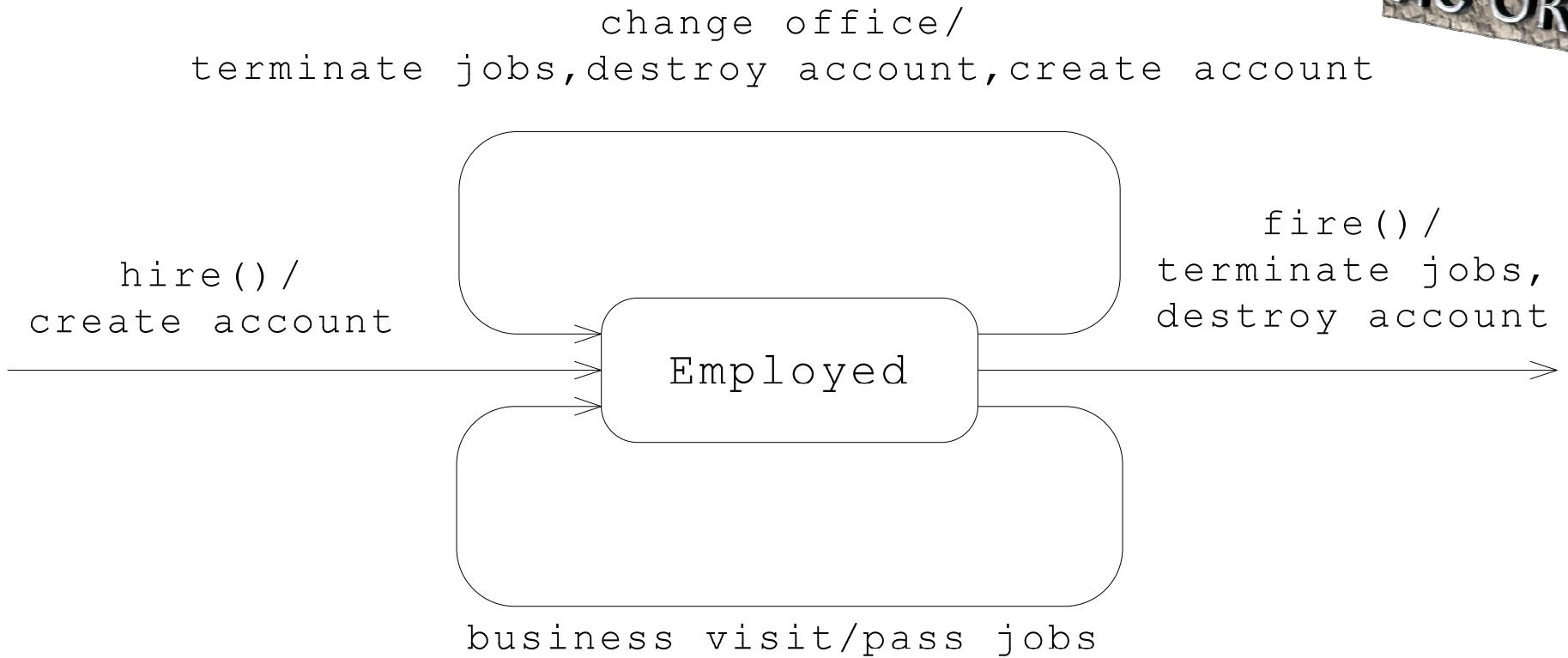
- Имя (name)
- Действие при входе (entry action): entry
- Действие при выходе (exit action): exit
- Множество внутренних переходов (internal transitions): событие / реакция
- Внутренняя деятельность (do activity): do
- Множество отложенных событий (defer events): defer

Простое состояние сотрудника (i)



- встречи с заказчиками вне офиса / переезд в другой офис

Простое состояние сотрудника (ii)

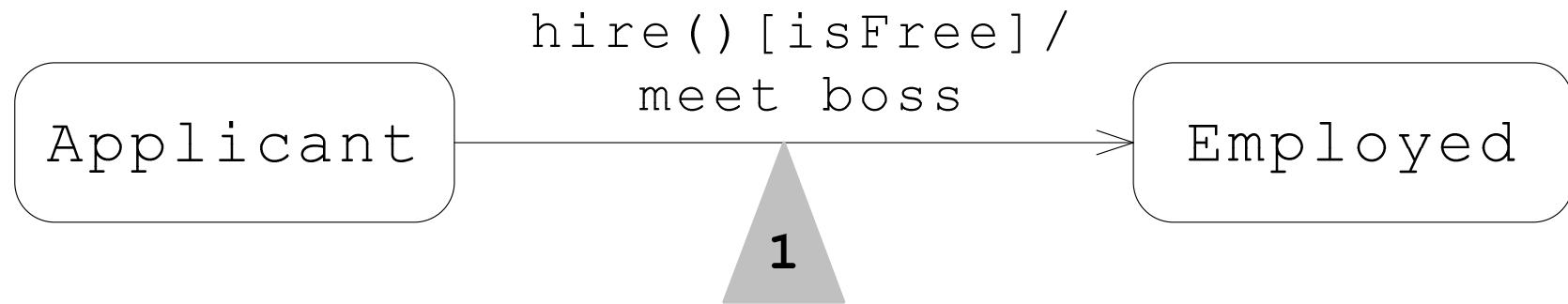


Вариант без дополнений

Простой переход

ис ок

Событие [Сторожевое условие] / Действие



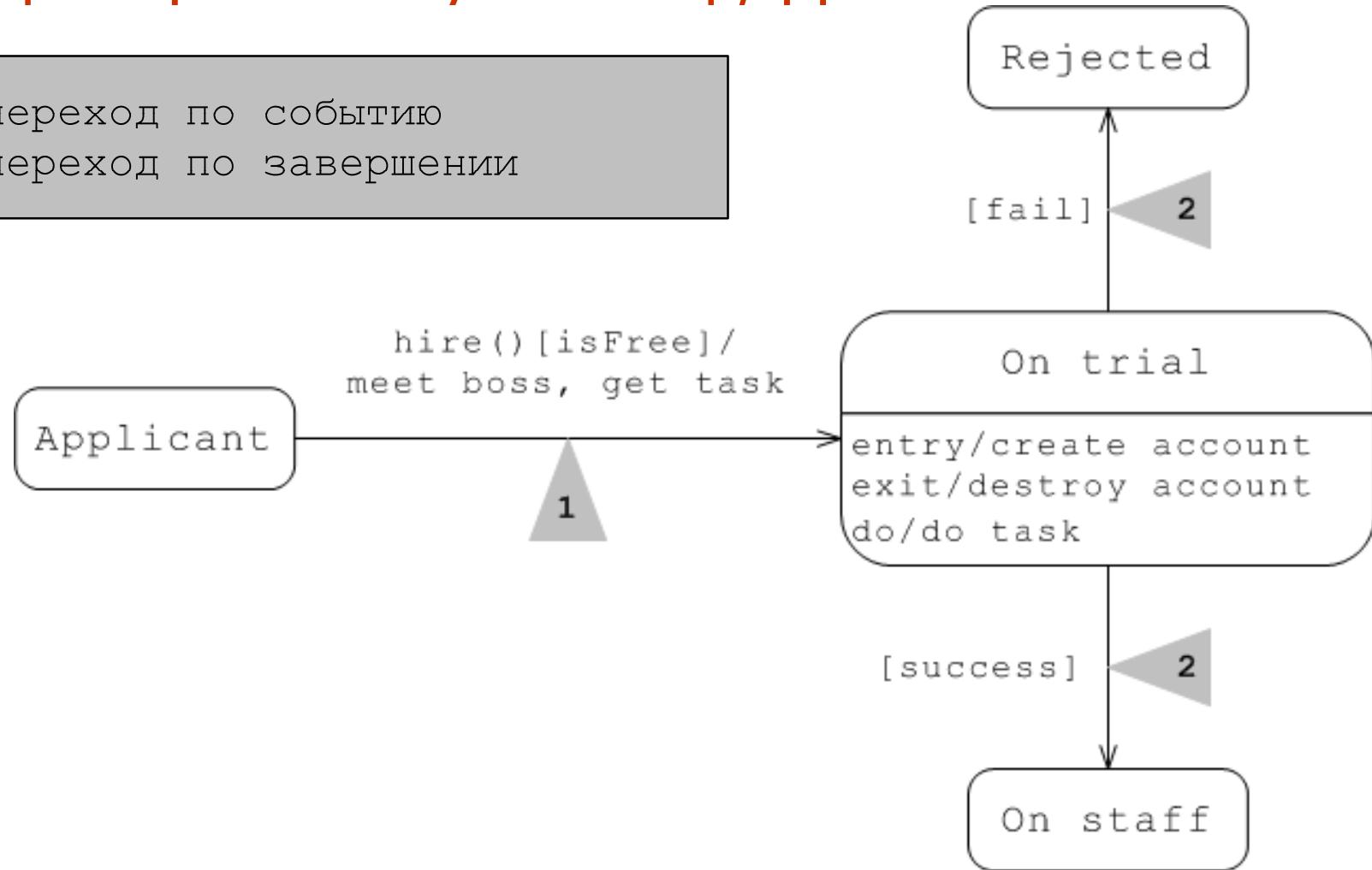
- Возникает **событие** `hire()` → переход **возбуждается** → проверяется **сторожевое условие** `isFree`
 - Если выполнено → переход **срабатывает** и выполняется **действие** `meet boss`
 - Если не выполнено → переход **не** срабатывает
 - Если ни один из возбужденных переходов не срабатывает → событие **теряется**

Переход по завершении

ис ок

Г Сторожевое условие 1 / Действие

- 1) переход по событию
- 2) переход по завершении



Действие на переходе

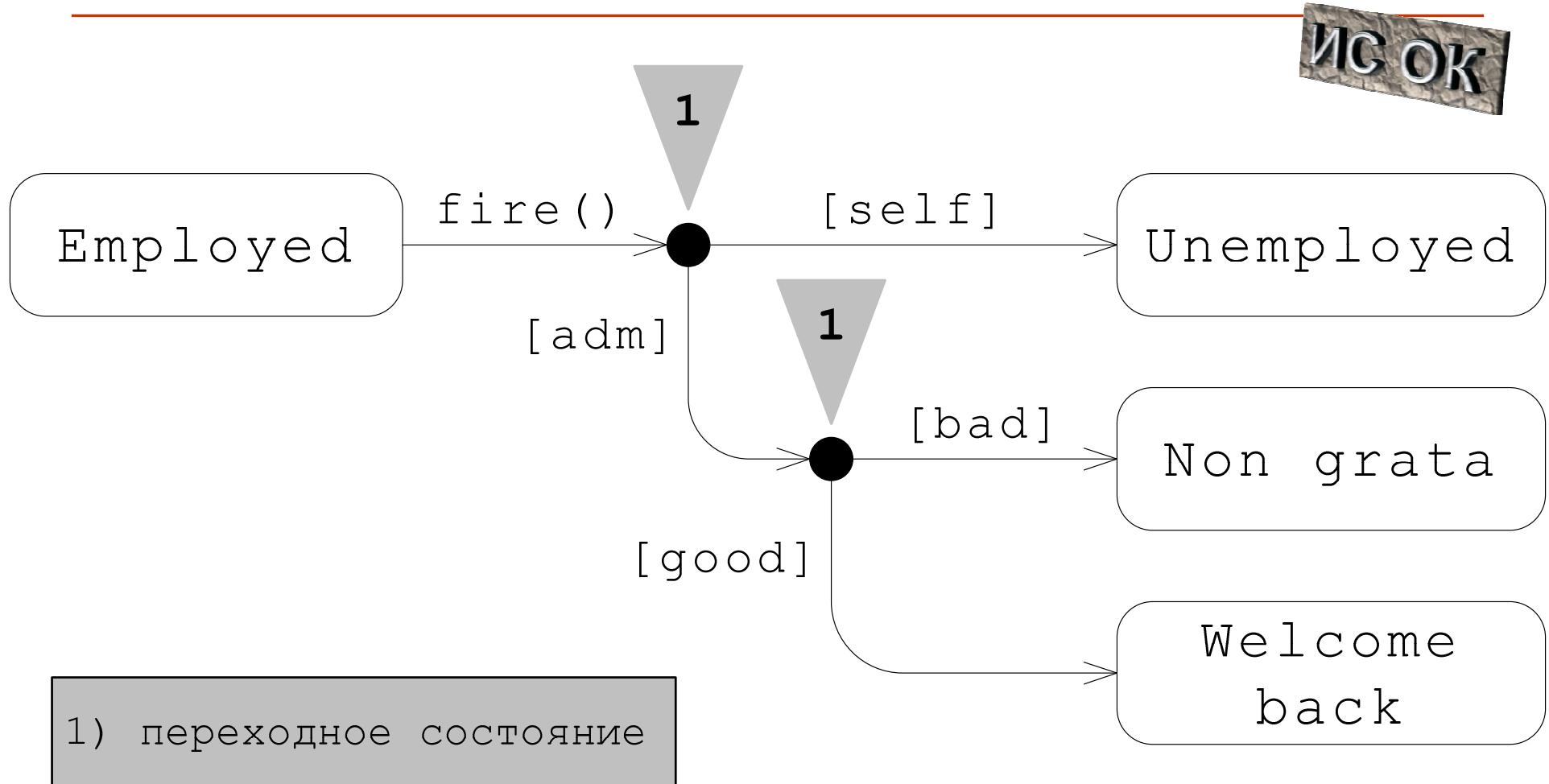
Действие — вычисление, чье время выполнения пренебрежимо мало

- Атомарное и непрерываемое
 - Если происходит событие, система его задерживает до окончания действия
- Безальтернативное и завершаемое
 - Раз начавшись, выполняется до конца
- Последовательность действий также является действием

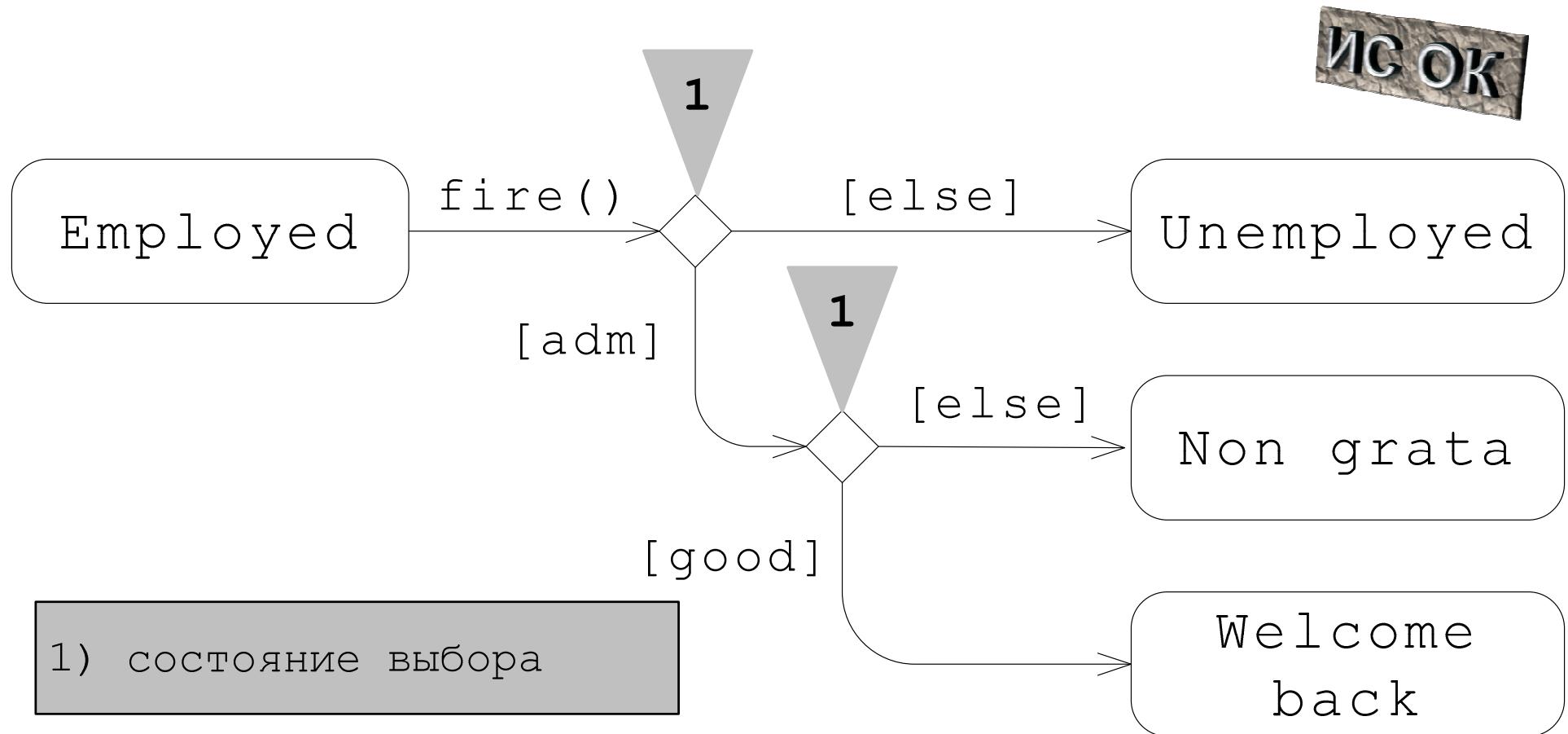
Сегментированные переходы (i)

- Сегменты перехода — это части, на которые может быть разбита линия перехода
- Разбивающие элементы:
 - Переходное состояние (junction state)
 - Состояние выбора (choice)
 - Действия посылки и приема сигнала
- Дерево сегментированных переходов :
 - Корневой сегмент из исходного состояния
 - Листовые сегменты в целевые состояния
- В UML 2 м.б. несколько исходных состояний, т.е. не дерево, а решетка

Сегментированные переходы (ii)



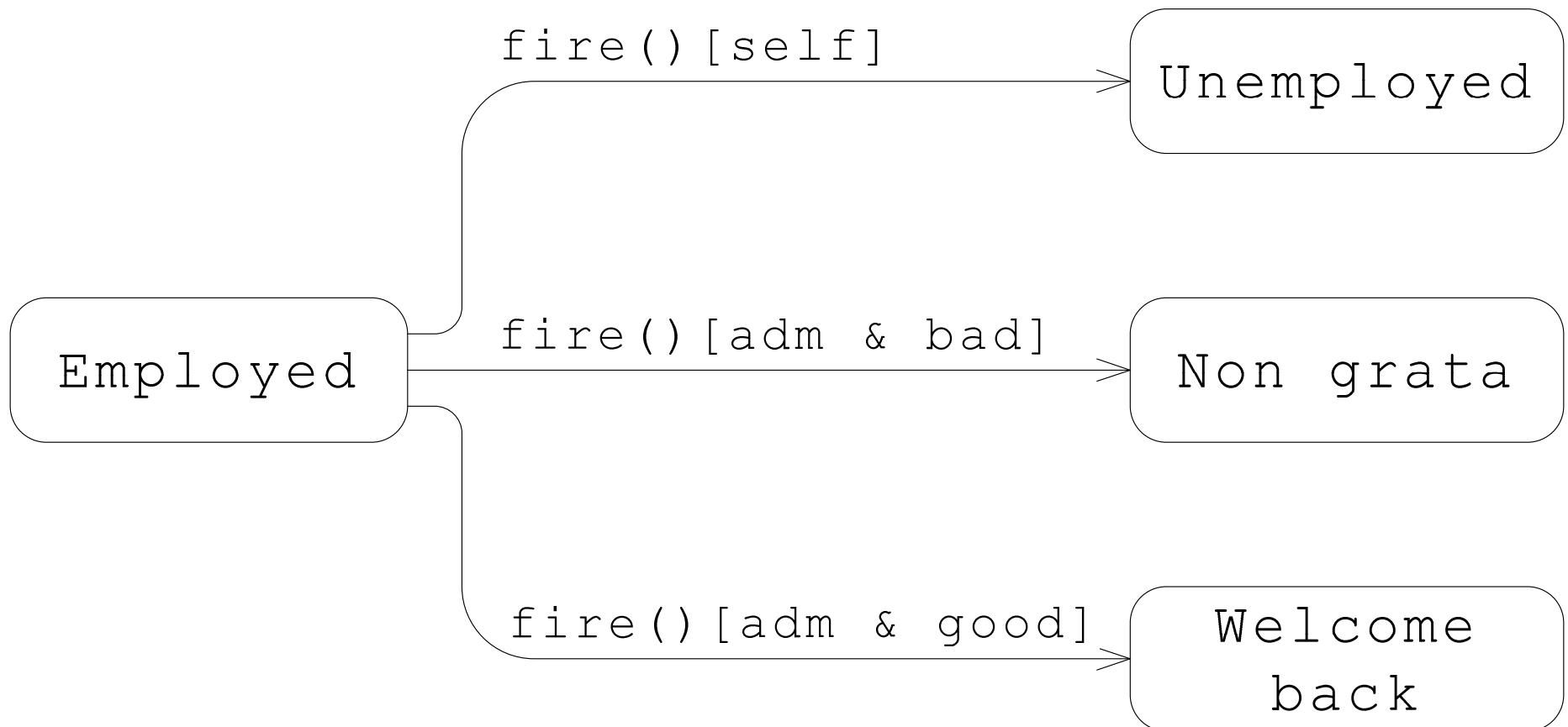
Сегментированные переходы (iii)



Использование предиката **else**

Сегментированные переходы (iv)

Эквивалентная модель



Классификация состояний (i)

- Составные состояния (composite state)
 - Последовательные (sequential / non-orthogonal state)
 - Параллельные (ортогональные) (concurrent / orthogonal state)
- Дополнительные состояния
 - Сылочное состояние (submachine state),
 - Состояние "заглушка" (stub state) – UML 1

Классификация состояний (ii)

- Специальные состояния (pseudo state)

- Начальное (initial)
- Заключительное (final)
- Слияние (join)
- Развилка (fork)
- Переходное (junction)
- Выбор (choice)
- Поверхностное историческое (shallow history)

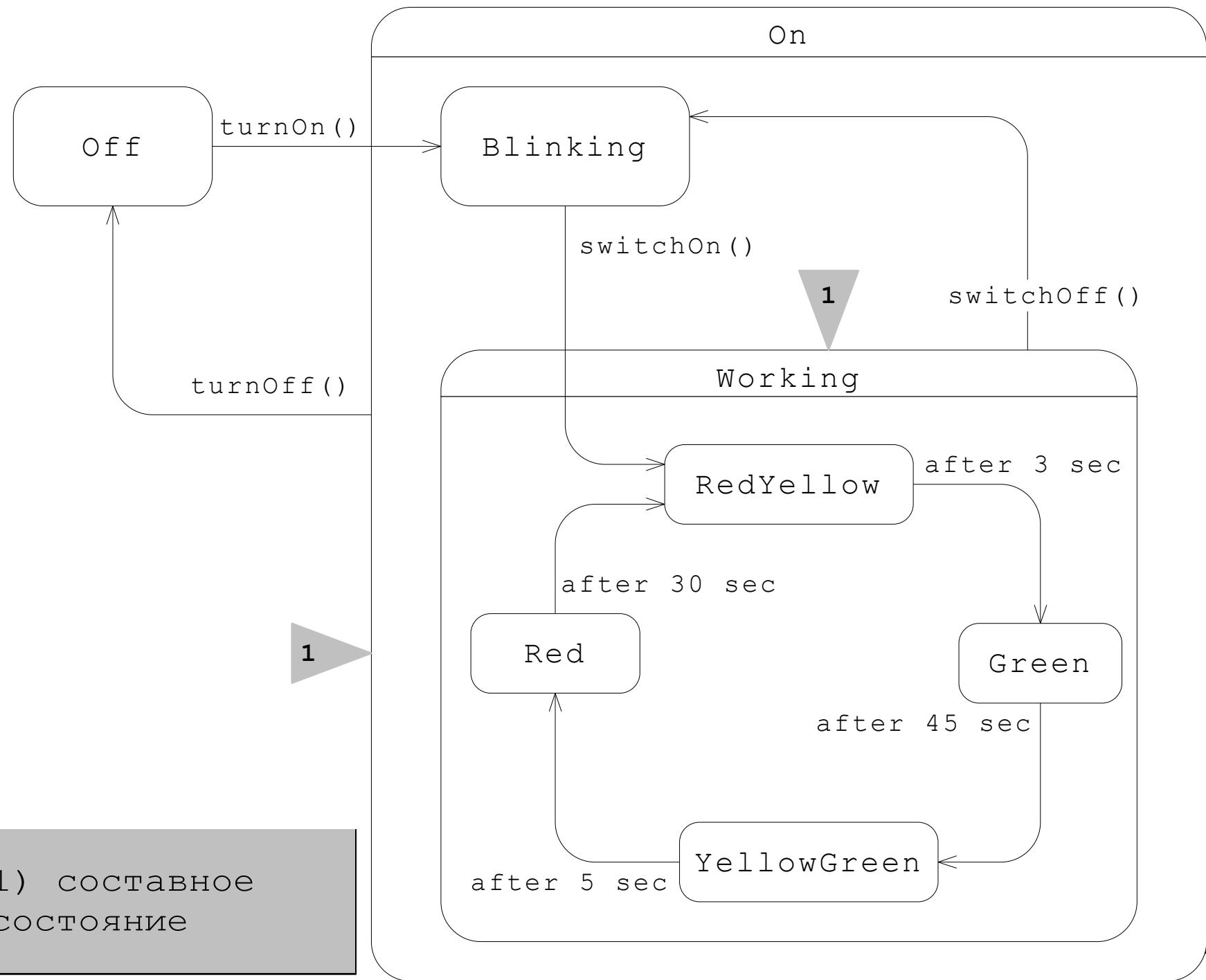
- Глубинное историческое (deep history)
- Синхронизирующее (synch) – UML 1
- Останов (terminate)
- «Точка входа» (entry point) – UML 2
- «Точка выхода» (exit point) – UML 2

Составные состояния

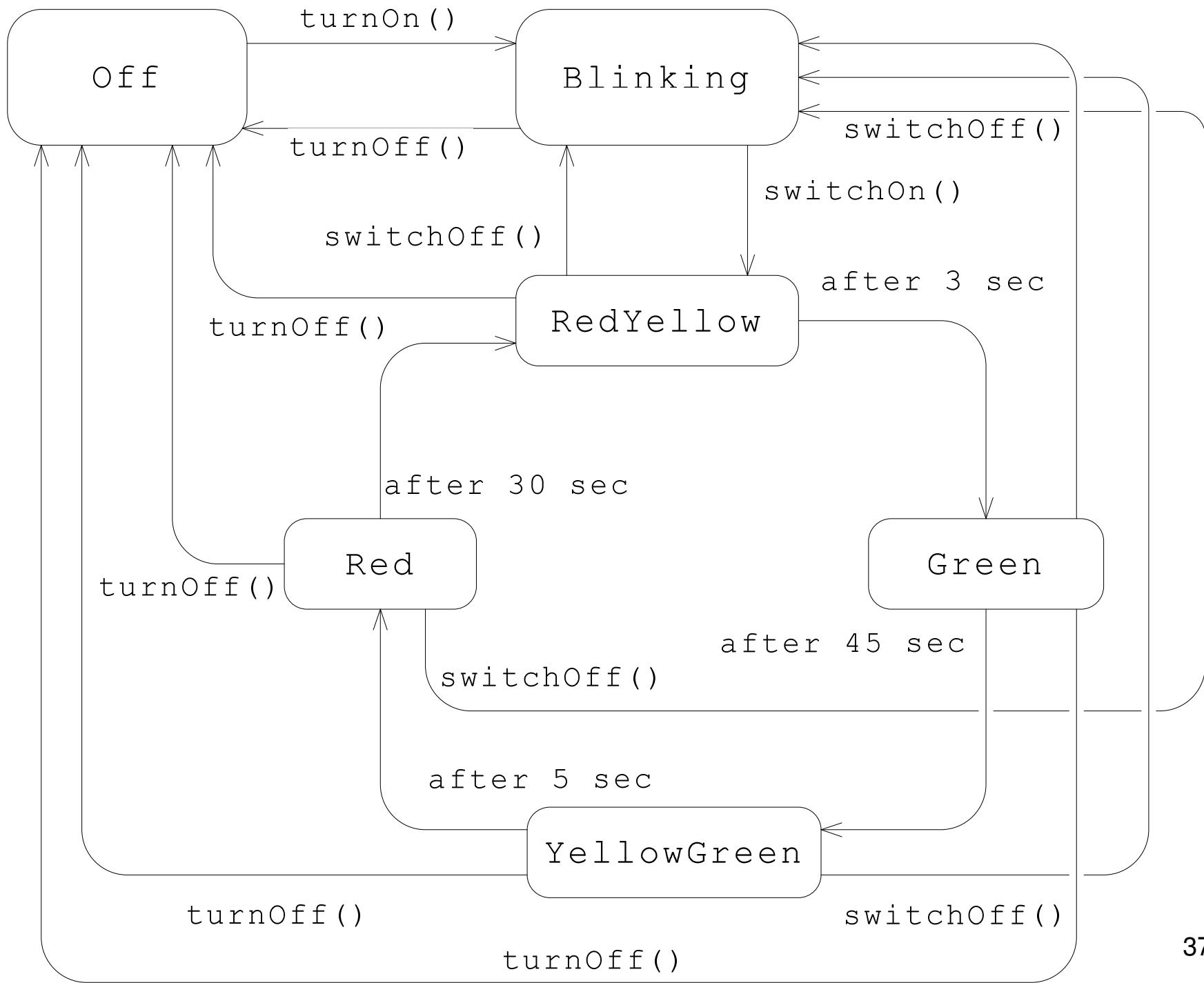
Составное состояние — это состояние, в которое вложена машина состояний

- Вложена одна машина — последовательное состояние
- Несколько — параллельное состояние
- Глубина вложенности не ограничена

Составные состояния: светофор



Светофор без составных состояний

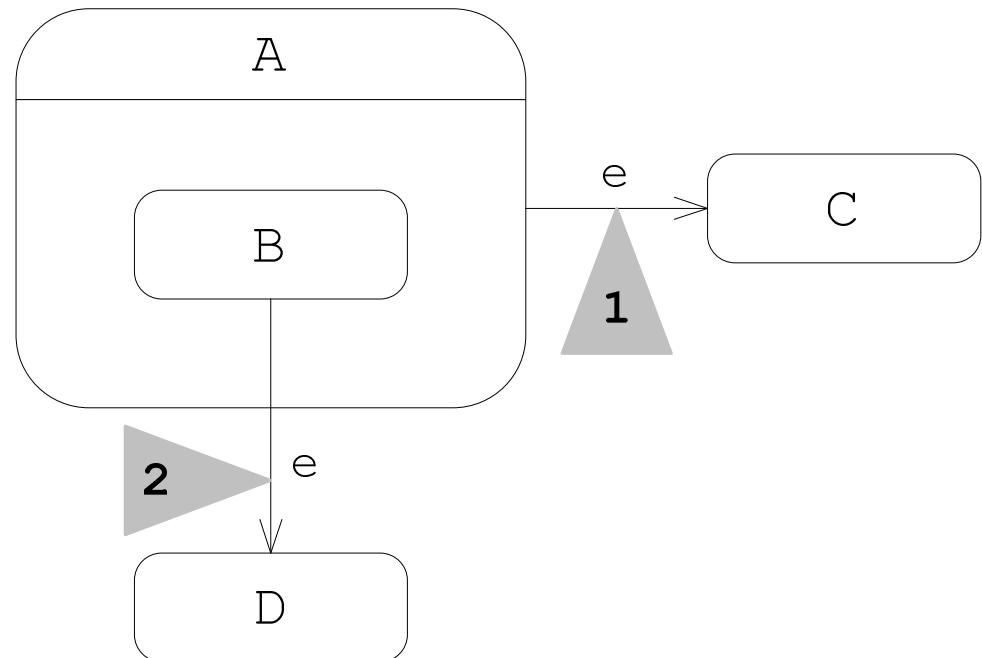


Переход из составного состояния

- Переход из составного состояния наследуется всеми вложенными состояниями

- Конфликт переходов:
переход из составного состояния конфликтует с переходом из вложенного в него состояния

- 1) глобальный переход
- 2) локальный переход



Преимущество имеет
локально определенный переход

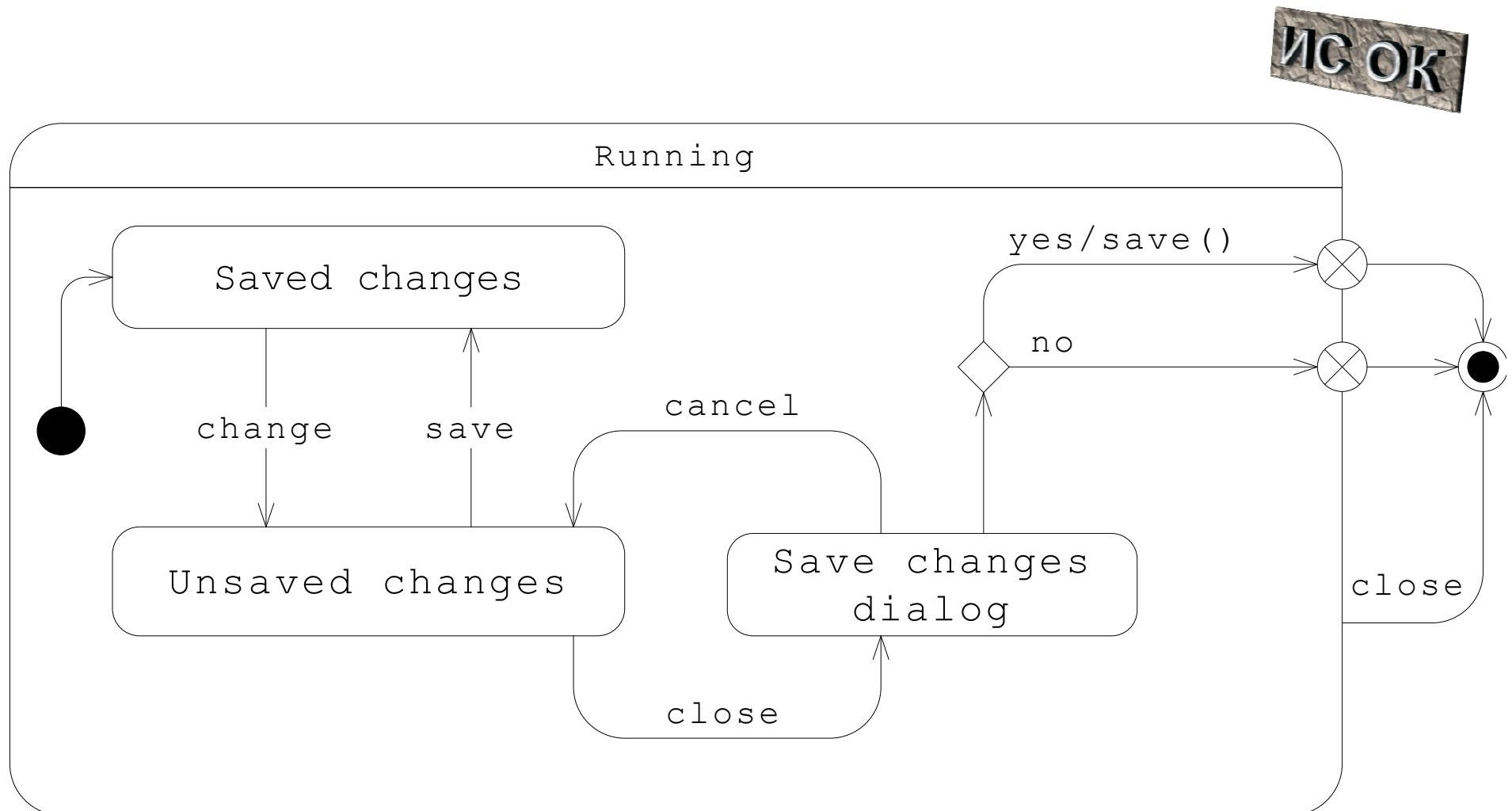
Локальное переопределение перехода (i)

ИС ОК

- В любой момент времени работа информационной системы отдела кадров может быть завершена
- Если при этом имеются данные, требующие сохранения, то решение о том, сохранять данные или нет, должен принимать пользователь



Локальное переопределение перехода (ii)



Некорректные дополнительные действия на переходах

ИС ОК

- ИС ОК должна сохранять для каждой проведенной сессии имя пользователя и продолжительность сессии

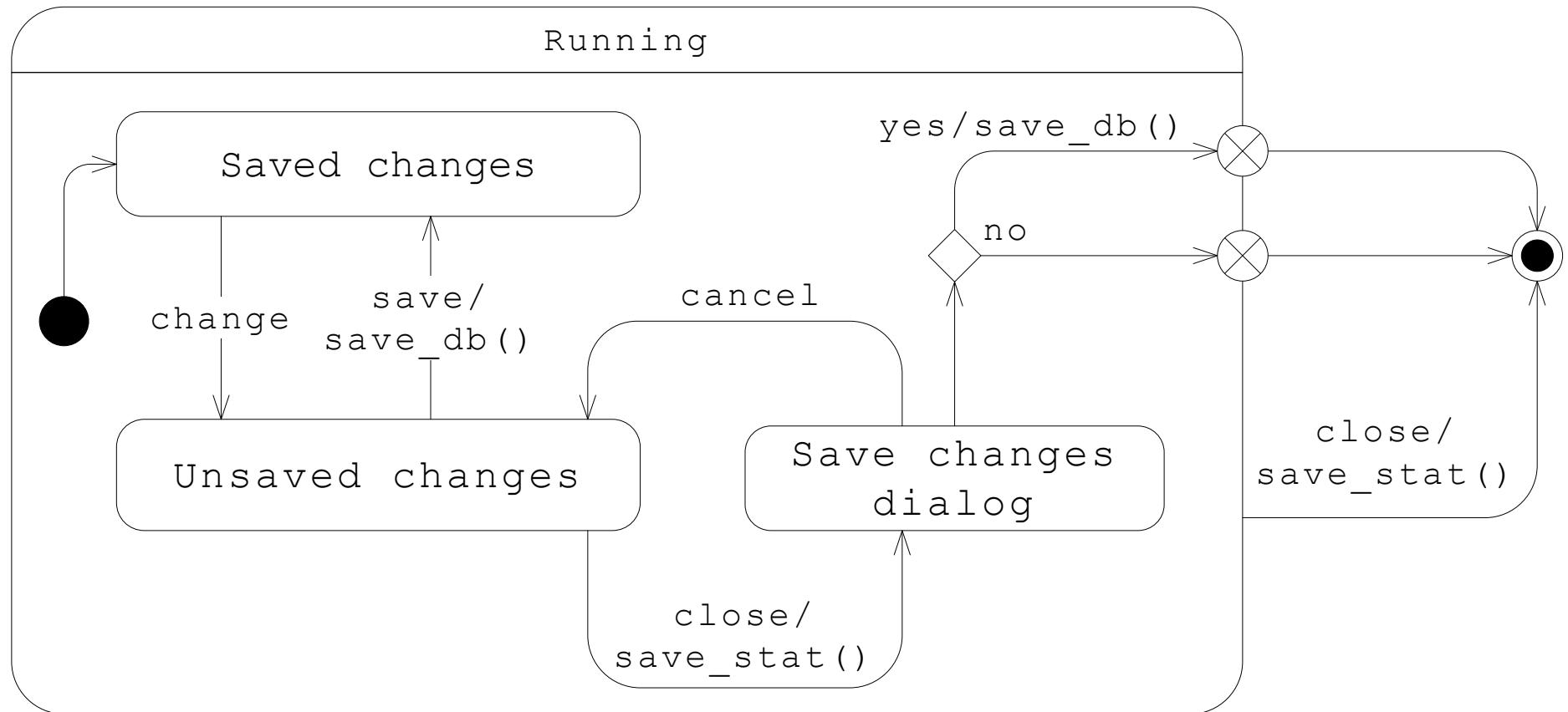
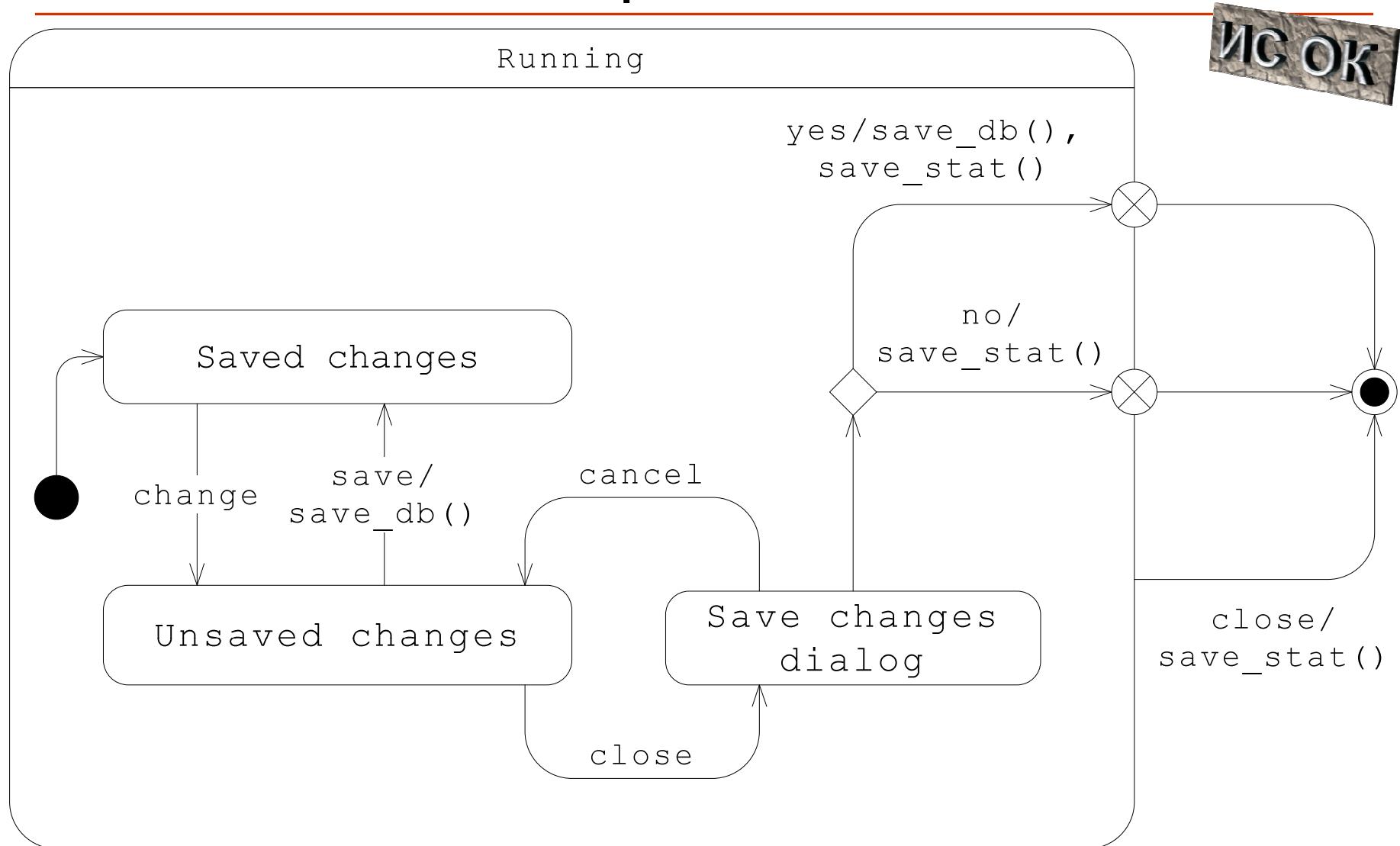


Диаграмма ошибочна!

Корректные дополнительные действия на переходах



Специальные состояния (i)

- НЕ может быть текущим активным состоянием автомата
- **Начальное состояние** — это специальное состояние, соответствующее ситуации, когда машина состояний **еще не работает**
 - Имеет: исходящий переход
 - НЕ имеет: действие на входе, выходе, внутренняя активность, событие перехода
 - Может иметь: действие на переходе

Специальные состояния (ii)

- **Заключительное состояние** — когда машина состояний **уже** не работает
 - Имеет: входящий переход
 - НЕ имеет: действия на входе, выходе, внутренняя активность
- На диаграмме: столько заключительных состояний, сколько семантически различных вариантов завершения работы (*авторский совет по стилю моделирования*)

Переходы между составными состояниями

Вид перехода	Диаграмма перехода	Эквивалентная диаграмма
Переход в составное состояние	<pre> graph LR B((B)) -- "E[G]/D" --> A[A] A --> C((C)) subgraph A entry["entry/D1"] exit["exit/D2"] end </pre>	<pre> graph LR B((B)) -- "E[G]/D, D1" --> C((C)) </pre>
Переход из составного состояния	<pre> graph LR A[A] -- "E[G]/D" --> B((B)) subgraph A entry["entry/D1"] exit["exit/D2"] end </pre>	<pre> graph LR C((C)) -- "E[G]/D2, D" --> B((B)) </pre>
Переход по завершении	<pre> graph LR A[A] -- "E[G]/D" --> C((C)) subgraph A entry["entry/D1"] exit["exit/D2"] end </pre>	<pre> graph LR C((C)) -- "E[G]/D, D2" --> B((B)) </pre>

Историческое состояние (i)

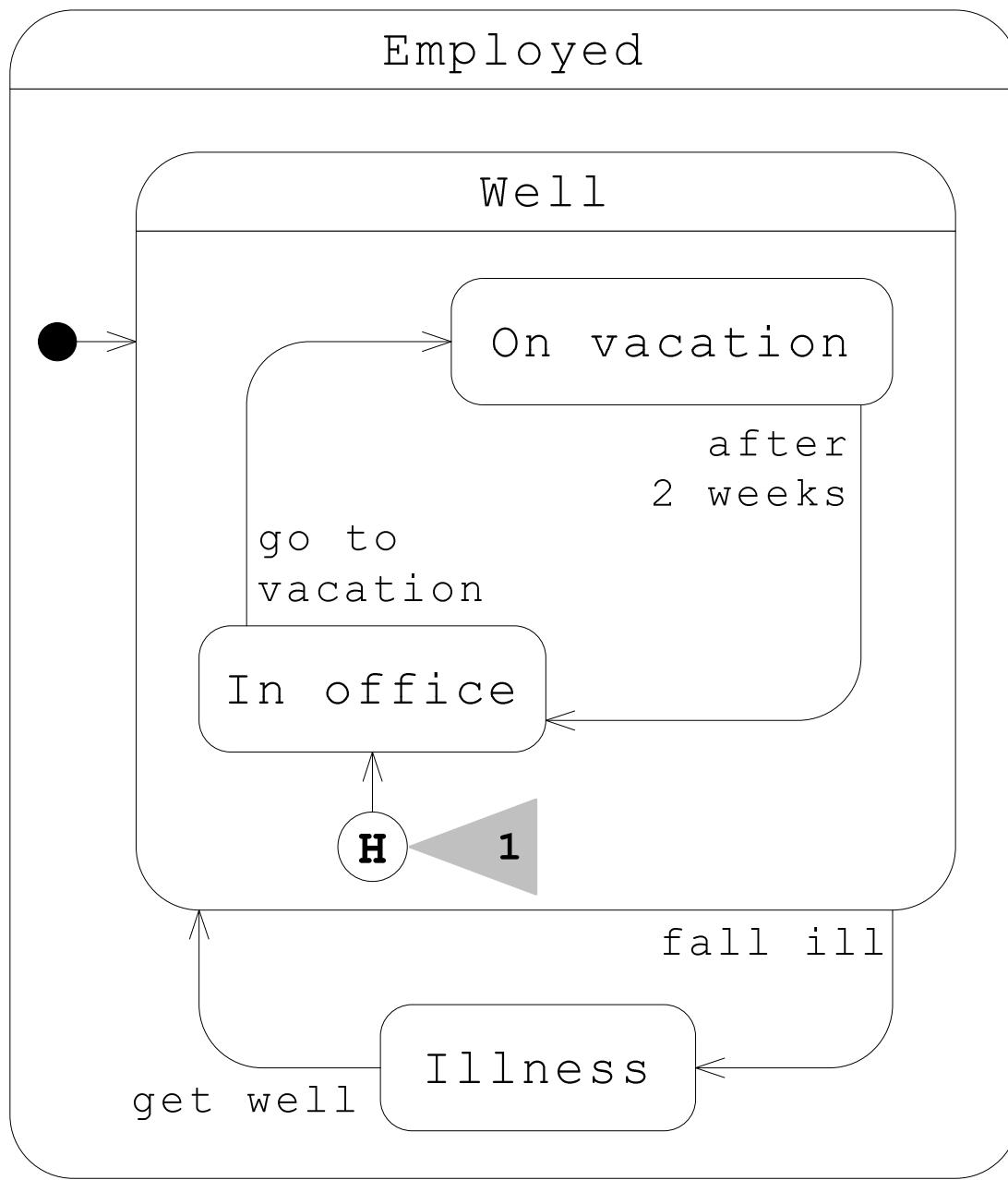
Историческое состояние — начальное состояние с дополнительной семантикой: автомат "продолжает начатое"

- Поверхностное историческое состояние (shallow history state) — помнит, какое состояние было активным на том же уровне вложенности
- Глубинное историческое состояние (deep history state) — помнит состояние и на вложенных уровнях

Историческое состояние (ii)

исок

- Состояния сотрудника: в офисе, в отпуске, на больничном
- Если сотрудник заболел в отпуске, то отпуск прерывается, а по выздоровлении возобновляется



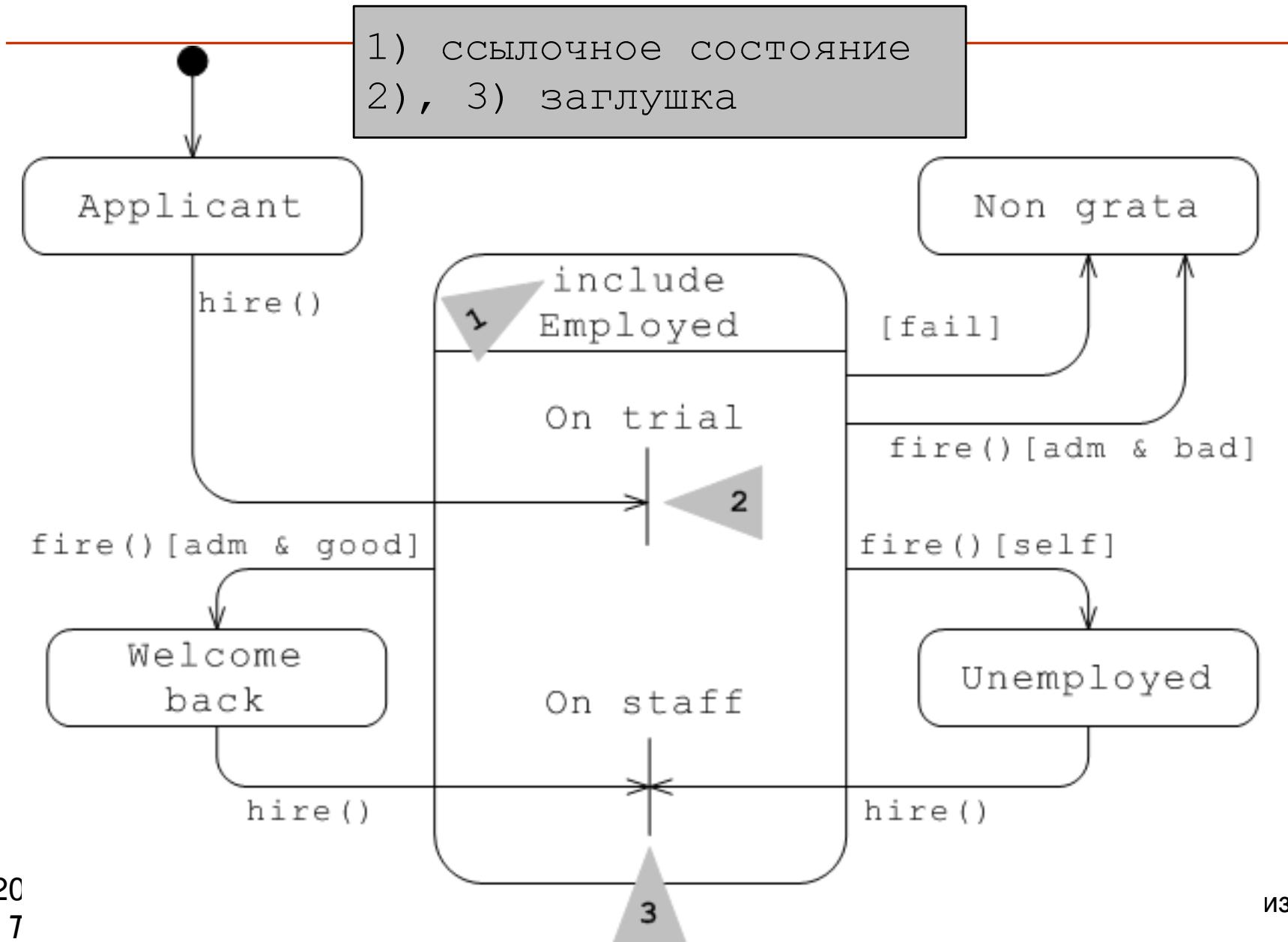
Вложенные машины состояний

- Для описания действительно сложного поведения:
- UML 1: **ссылочное состояние и состояния заглушки**
- UML 2: **вложенный автомат с точками входа и выхода**
- Критерий Дейкстры: «правильно написанный модуль помещается на одной странице»

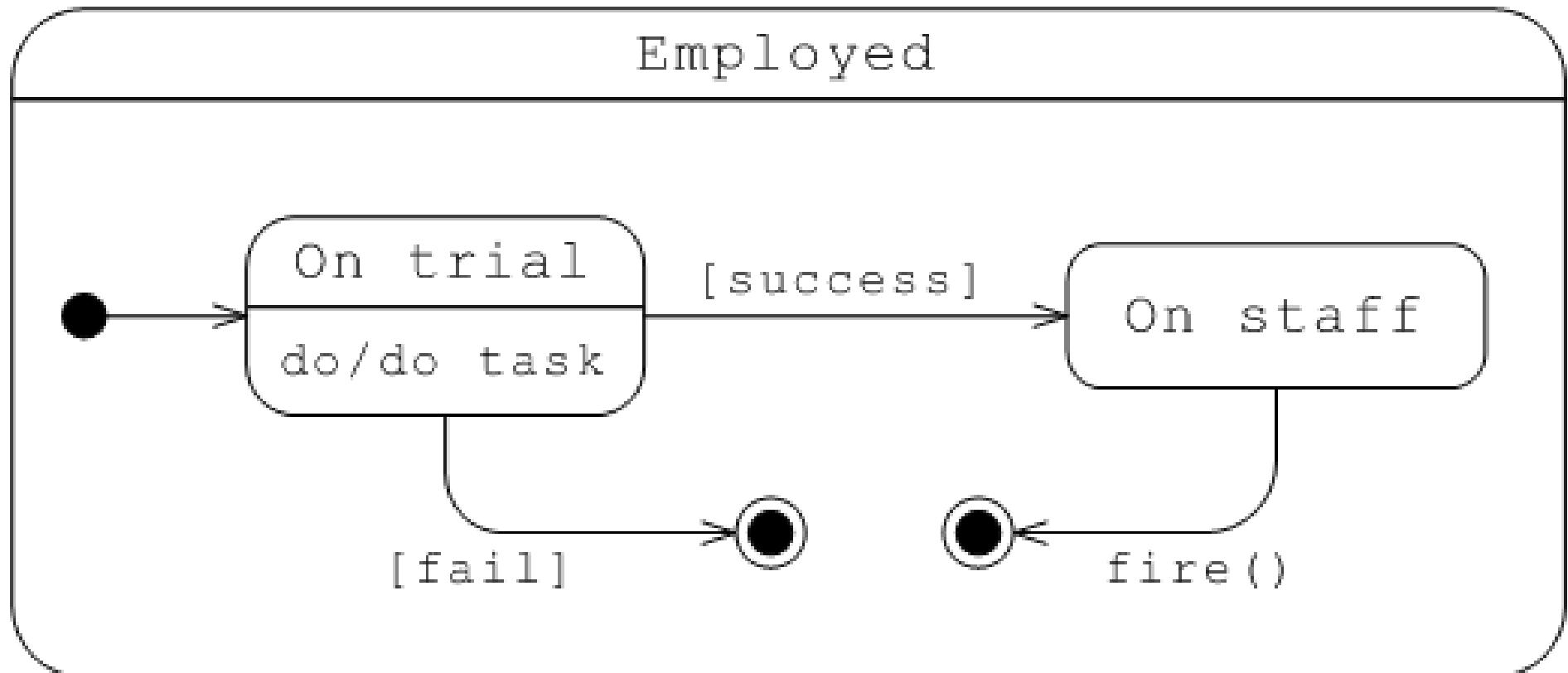
Сылочное состояние и заглушка (i)

- Сылочное состояние (submachine state) include — обозначает вложенный в него автомат
- Состояние заглушки (stub state) — обозначает в сылочном состоянии некоторое вложенное состояние того составного состояния, на которое делается ссылка

Сылочное состояние и заглушка (ii)

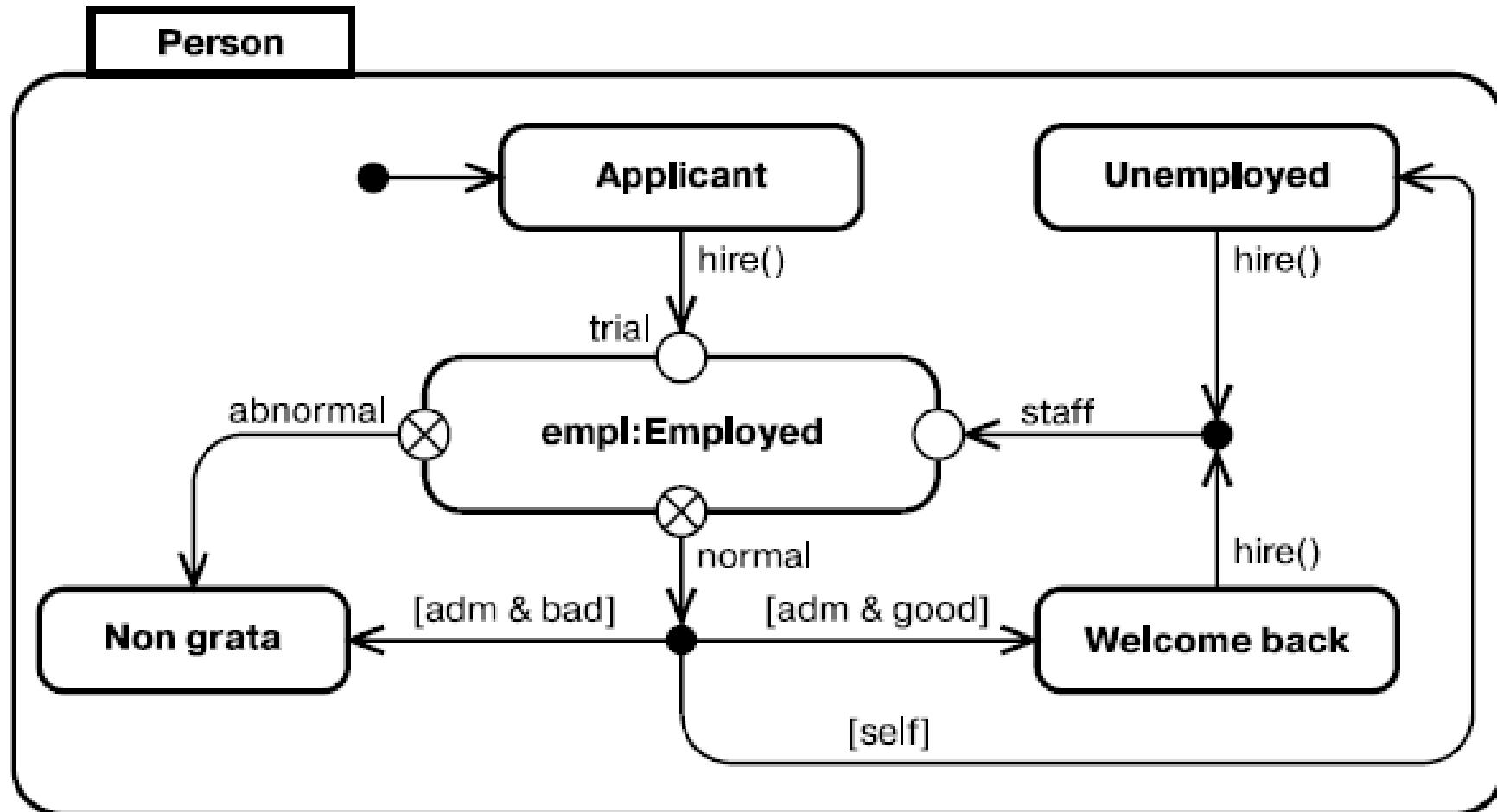


Раскрытие ссылочного состояния



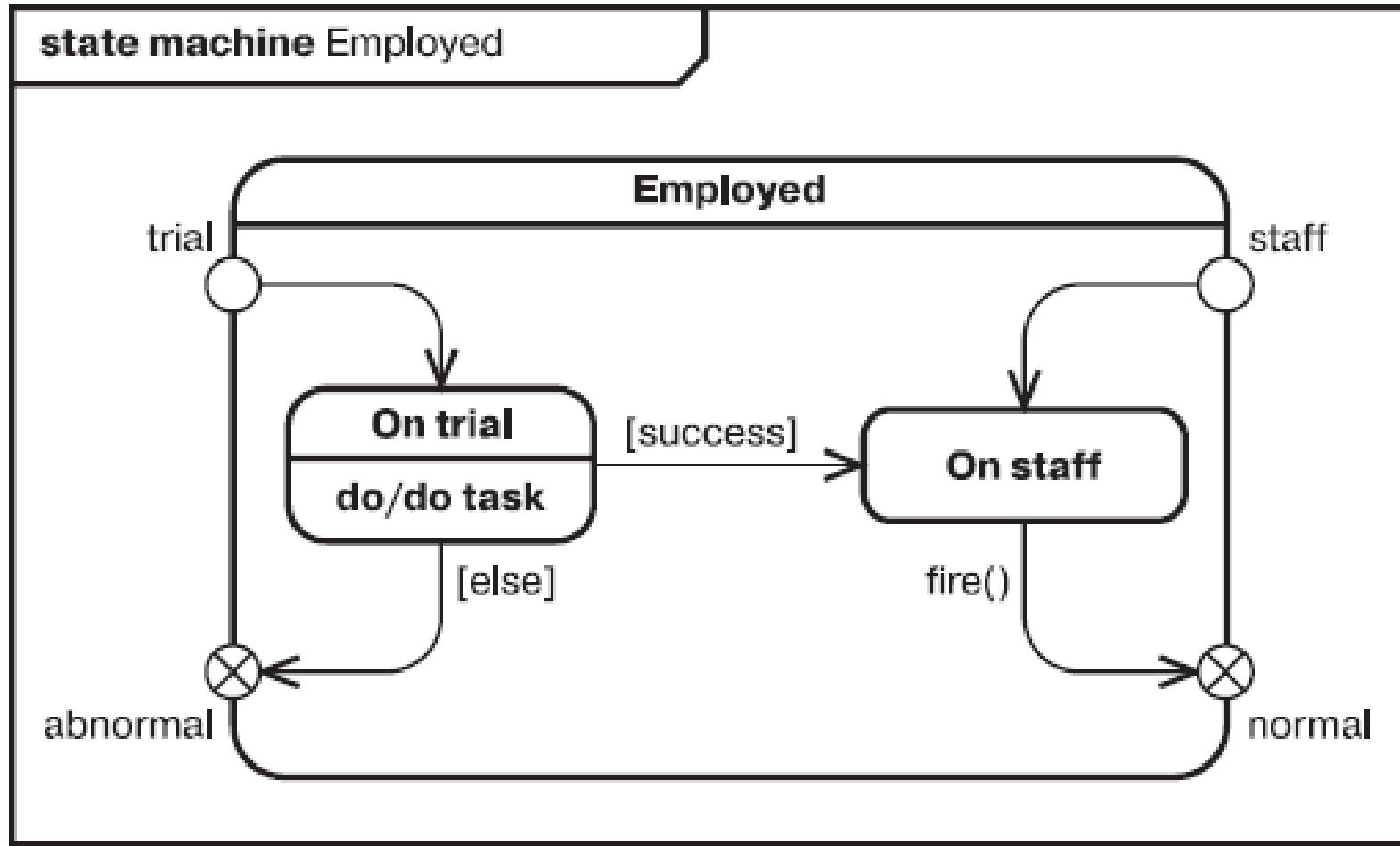
Вложенный конечный автомат

- Диаграмма верхнего уровня: вложенный автомат как простое состояние



Раскрытие вложенного автомата

- Диаграмма нижнего уровня: вложенный автомат раскрывается как составное состояние



События

- Переход может быть нагружен: событием перехода, сторожевым условием и действиями на переходе
- События в UML:
 - событие вызова (call event)
 - событие сигнала (signal event)
 - событие таймера (time event)
 - событие изменения (change event)

Событие вызова (i)

Событие вызова — это событие,
возникающее при вызове метода класса

- Класс должен иметь соответствующую **операцию**
- Может иметь **аргументы**, как всякая операция

Событие вызова (ii)

Операции, определенные в классах:

Person

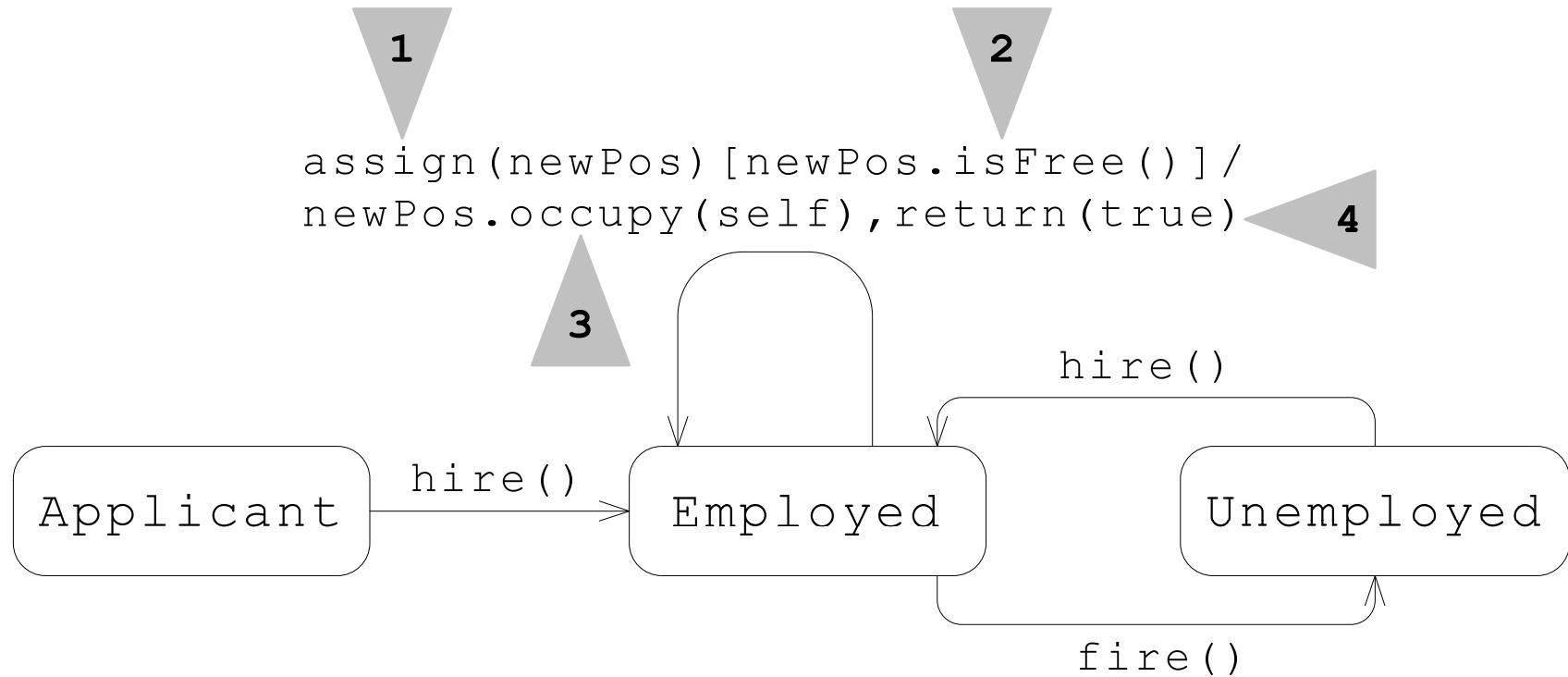
```
+assign (newPos: Position) : Boolean  
+hasPosition () : Boolean
```

Position

```
+occupy (person: Person) : Boolean  
+isFree () : Boolean
```

ИС ОК

Событие вызова (iii)

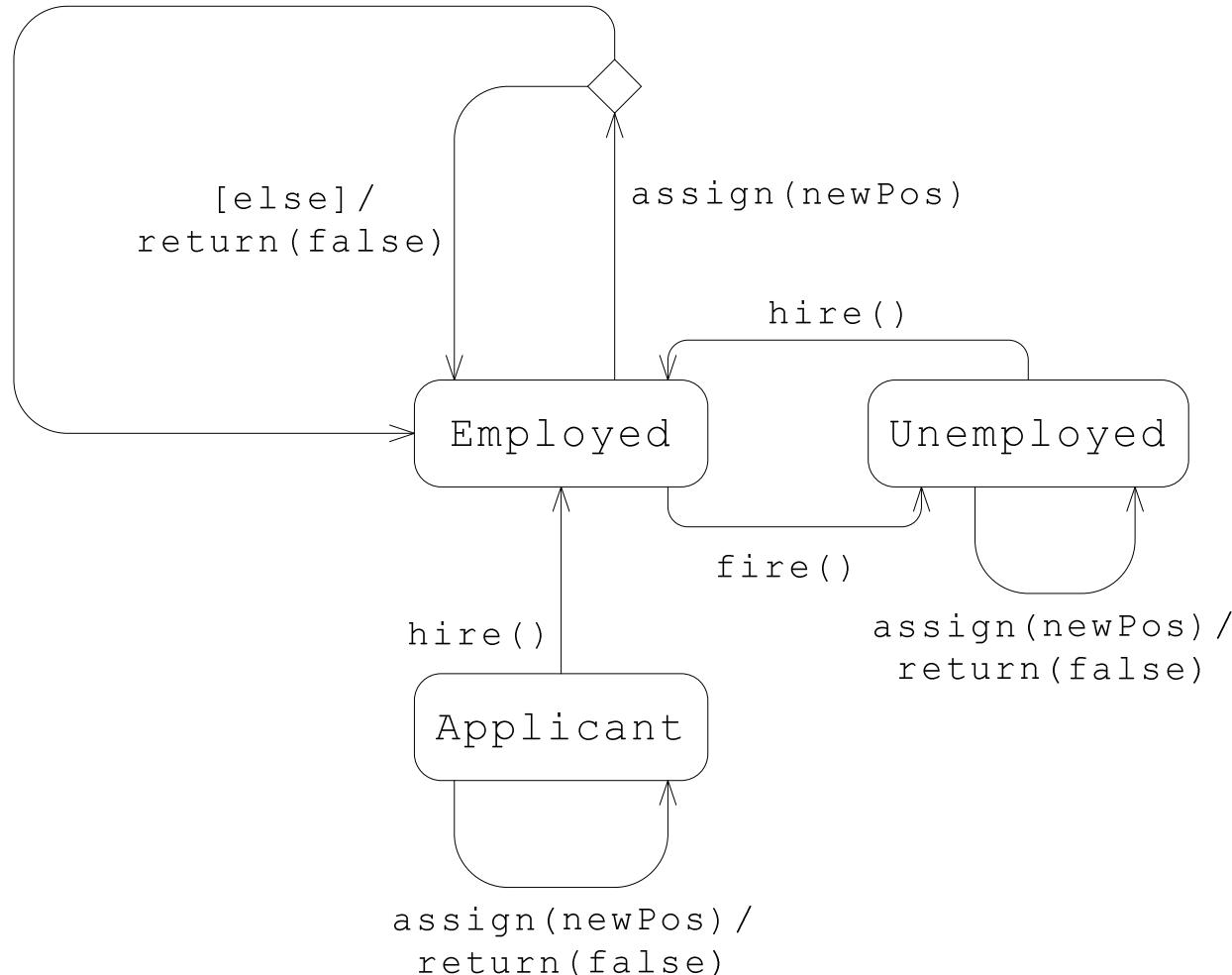


- | | |
|------------------------|--------------------------|
| 1) событие на переходе | 3) действие на переходе |
| 2) сторожевое условие | 4) возвращаемое значение |

Событие вызова (iv)



```
[newPos.isFree()]/  
newPos.occupy(self), return(true)
```

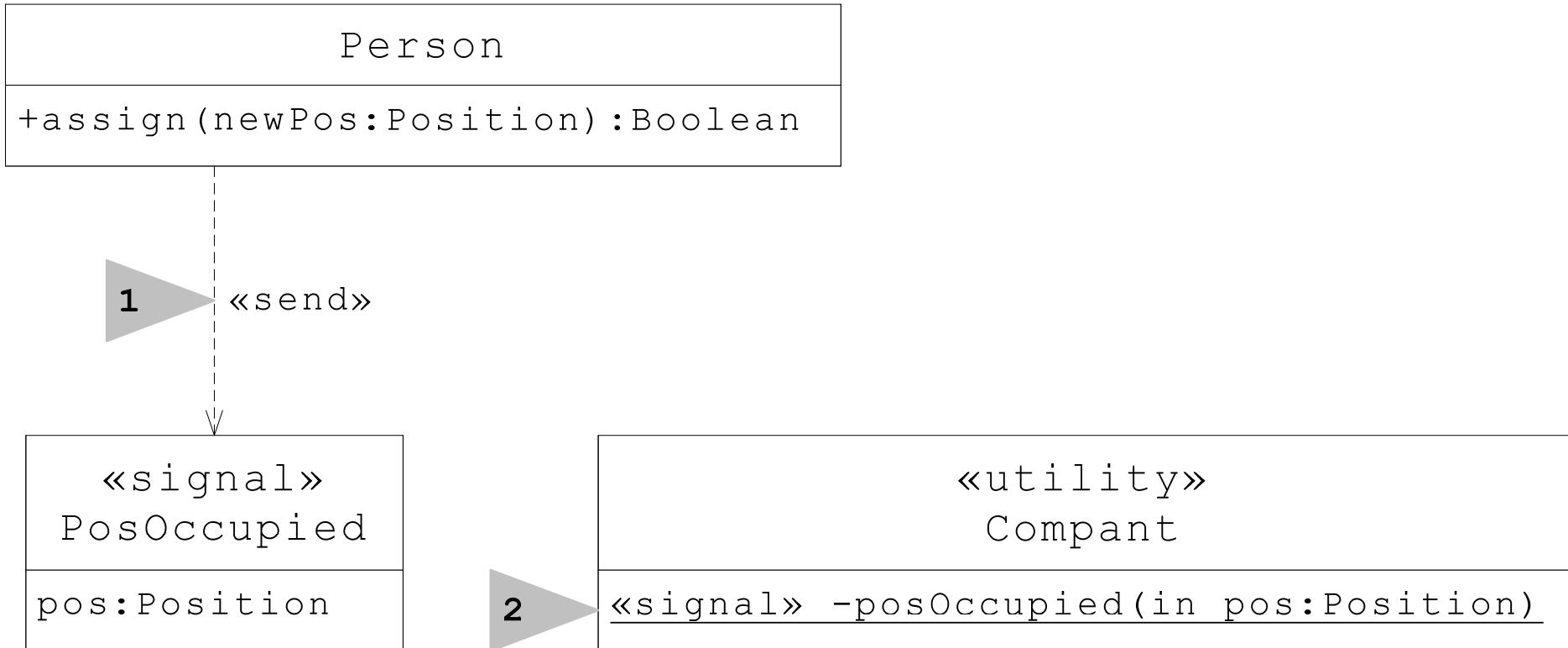


Событие сигнала

- Событие сигнала — возникает при посылке сигнала
- Сигнал — это именованный объект, который создается другим объектом (отправителем) и обрабатывается третьим объектом (получателем)
 - В UML 1: экземпляр класса со стереотипом «signal»
 - В UML 2: самостоятельный классификатор
- Атрибуты = параметры сигнала
- Операция (по умолчанию): имя: send, параметр: множество объектов-получателей

Описание сигнала на диаграмме классов

ис ок

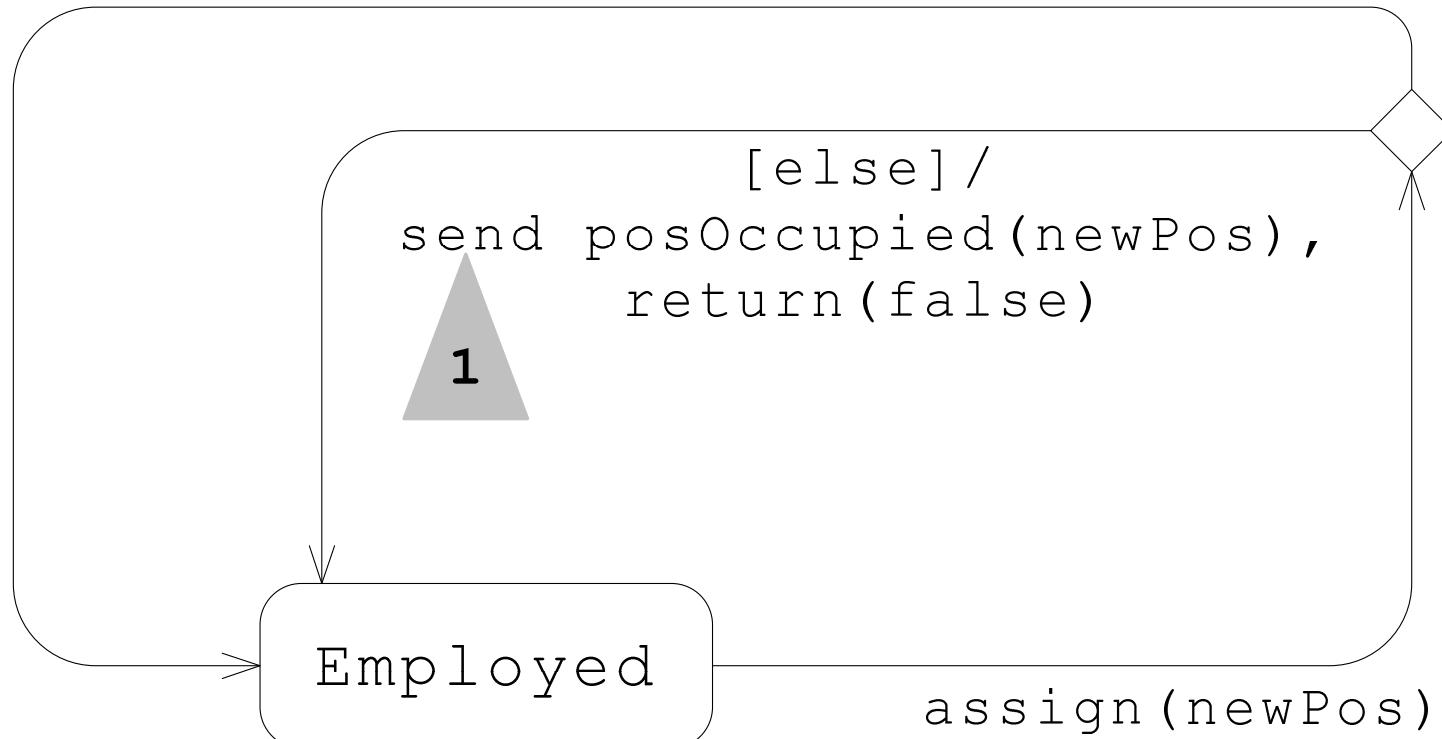


- 1) кто посылает сигнал
- 2) где обрабатывается сигнал

ис ок

Посылка сигнала

```
[newPos.isFree() ] /  
newPos.occupy(self) , return(true)
```

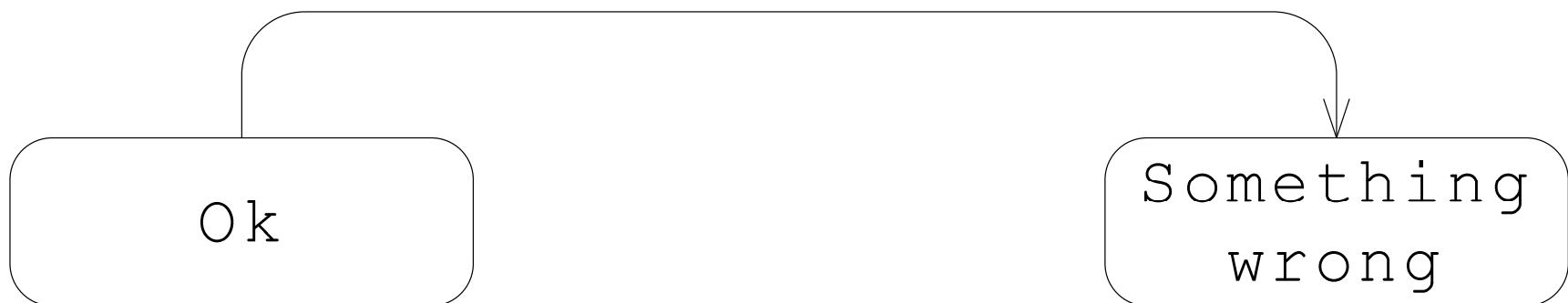


1) посылка сигнала

Переход по событию сигнала

ис OK

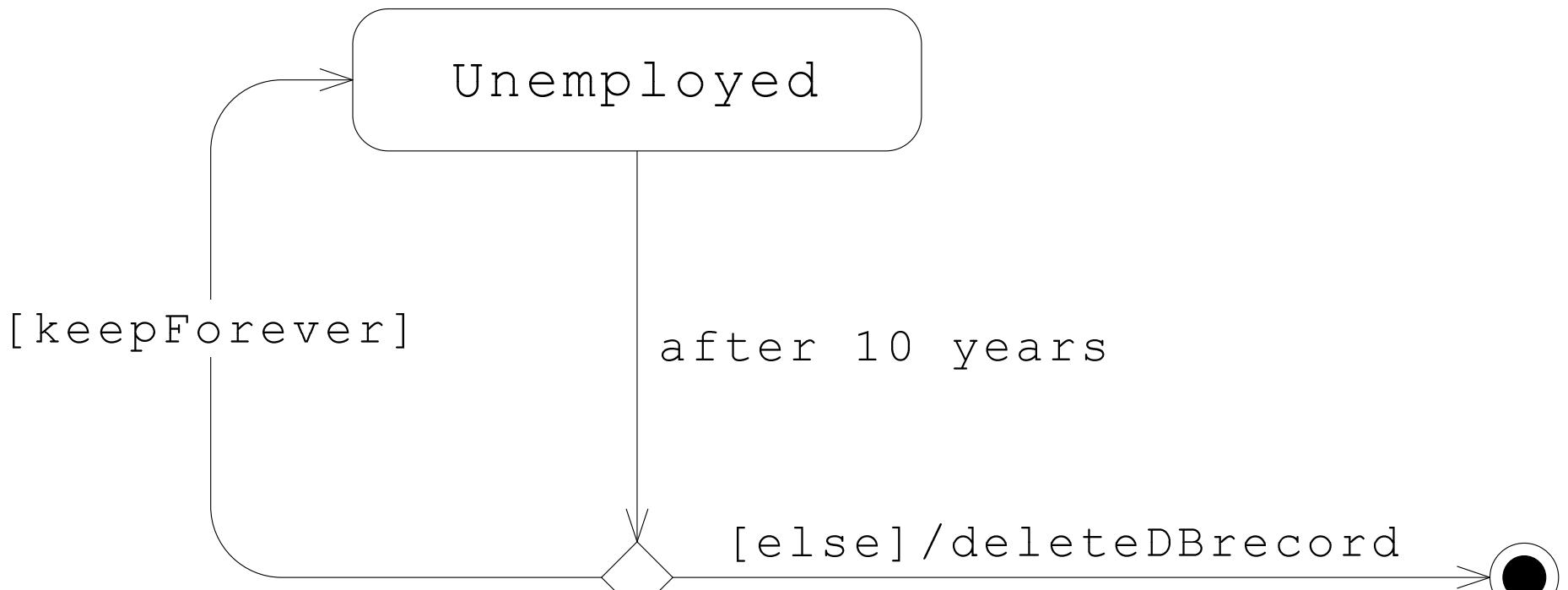
posOccupied(pos) / putLog



Переход по событию сигнала между состояниями класса Company

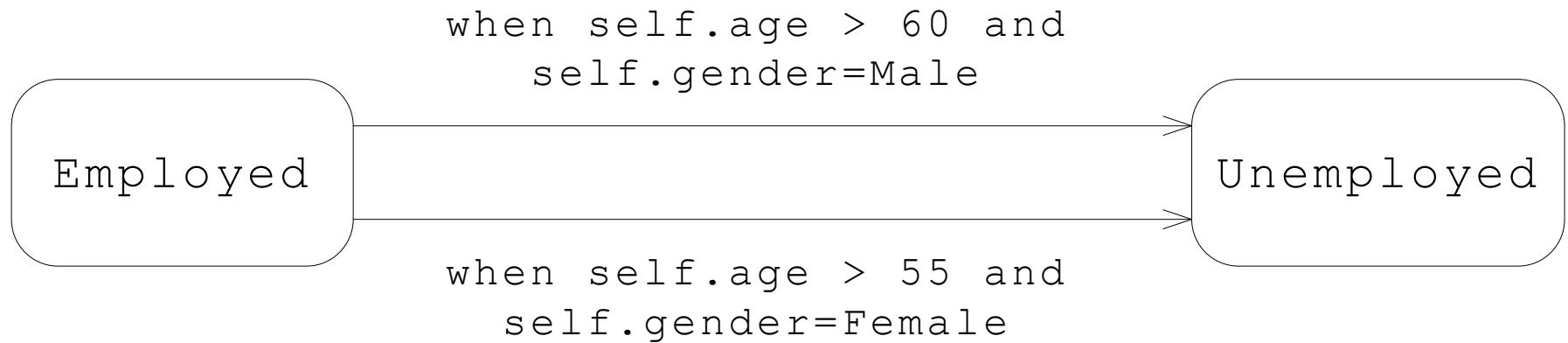
Событие таймера

Событие таймера `after` — возникает, когда истек заданный интервал времени с момента попадания автомата в данное состояние



Событие изменения

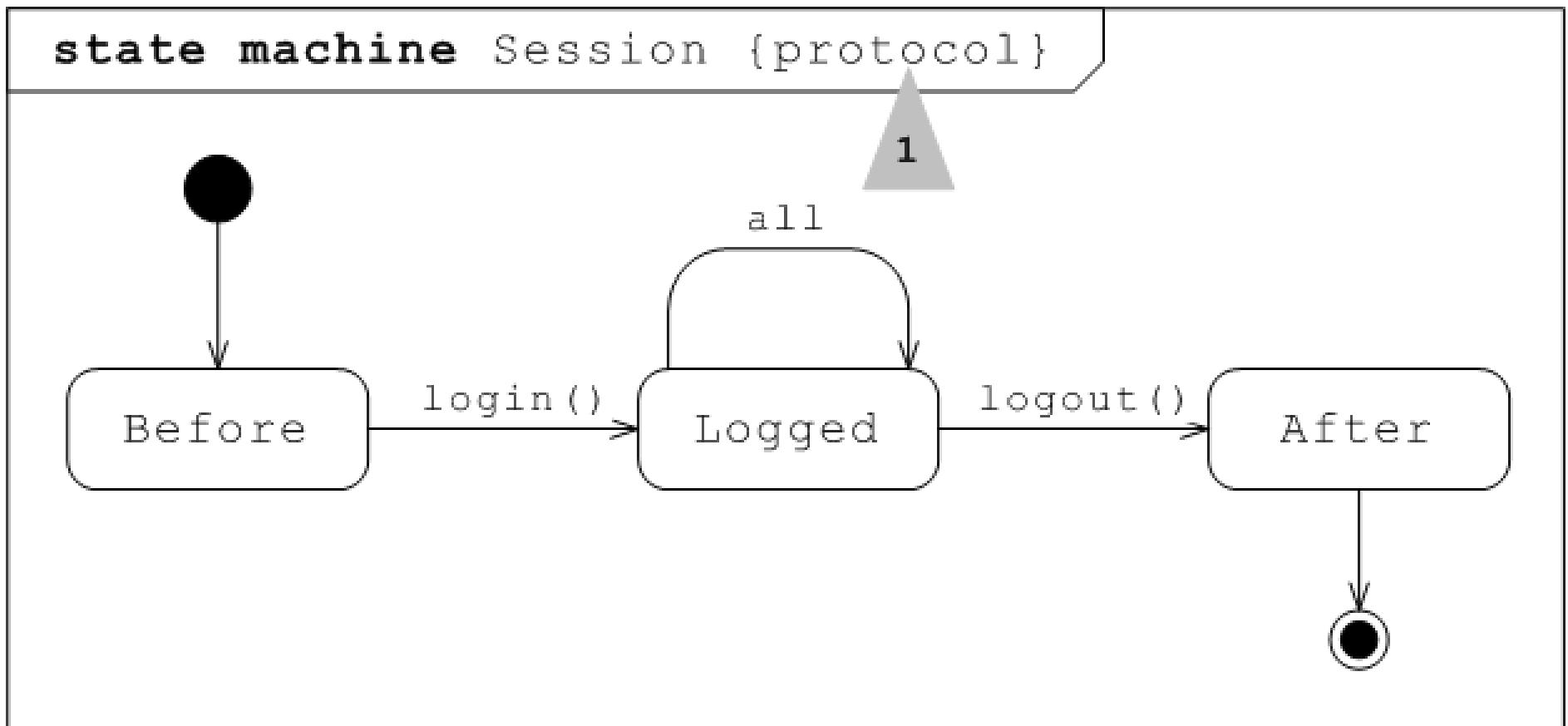
Событие изменения when — возникает, когда некоторое логическое условие становится истинным, будучи до этого ложным



Протокольный автомат (i)

Протокольный автомат protocol =
Протокольная машина состояний
(protocol state machine) —
это машина состояний,
предназначенная для задания
допустимых последовательностей
вызовов и сигналов

Протокольный автомат (ii)



2. Диаграммы деятельности

- Вторичны по отношению к диаграммам состояний
- + Дополнительные средства
- Область применения – расплывчата
- Сущность: состояние деятельности
- Отношение: переход по завершении

3.1 Действие и деятельность

3.2 Граф деятельности. Узлы управления

3.3 Дорожки и разбиения

3.4 Траектория объекта и поток данных

3.5 Отправка, прием сигналов и работа с таймером

3.6 Сравнение способов описания поведения

3.7 Прерывания и исключения

3.8 Структурные узлы деятельности

3.9 Применение диаграмм деятельности

Действия в UML 1

Тип действия	Ключевое слово	Описание
Присваивание значения	=	Присваивание значения атрибуту
Вызов операции	Call	Вызов операции заданного объекта с заданными аргументами
Создание объекта	Create	Создание и инициализация нового объекта
Уничтожение объекта	Destroy	Уничтожение объекта и всех его составляющих
Возврат значения	Return	Возврат значения в точку вызова операции
Посылка сигнала	Send	Создание и отправка нового сигнала
Прекращение выполнения	Terminate	Прекращение работы машины состояний объекта и его уничтожение
Не интерпретируемое действие	любой текст	Любое действие, не определенное в UML
Повторитель	* [повторитель]	Предписывает выполнить действие несколько раз

Действия в UML 1.5 – 2.x

- Действия применимы к классификаторам
- Позволяют манипулировать экземплярами классификаторов
 - читать и записывать значения атрибутов
 - вызывать методы классификаторов
- Имеют присущие им наборы входных и выходных параметров = **контакты** (pin)
- Действия атомарны, непрерываемы, безусловны и завершаемы
- Определено более 50 действий = полная операционная семантика → **Executable UML**

Группы действий в UML 2

Группа действий	Прототип в UML 1	Описание
Структурные действия	=	Запись или чтение значения атрибута объекта, параметров операций и др.
Действия по вызову и возврату	Call Return	Действия на вызывающей и вызываемой сторонах
Действия с классификаторами	Create Destroy	Создание и уничтожение объектов, получение экстента, классификация
Отправка и прием сигналов	Send	Создание нового сигнала и отправка, возбуждение и обработка исключений
Действия с временем	Нет прототипа	Действия для получения текущего времени и другие действия с таймером
Действие с заданной интерпретацией	Не интерпретирующее действие	Любое действие, определенное внешними по отношению к UML средствами

Деятельность

Деятельность — описание поведения в форме графа деятельности

Действие (action) \leftrightarrow деятельность (activity)

Характеристика	Действие	Деятельность
Внешнее событие	Не прерывает выполнения	Может прерваться и завершить выполнение
Завершаемость	Всегда завершается самостоятельно	Может продолжаться неограниченно долго
Внутренняя структура	Не моделируется в UML	Может быть раскрыта на отдельной диаграмме
Время выполнения	Пренебрежимо мало	Продолжительное

- Обобщающее понятие — **активность**

Граф деятельности

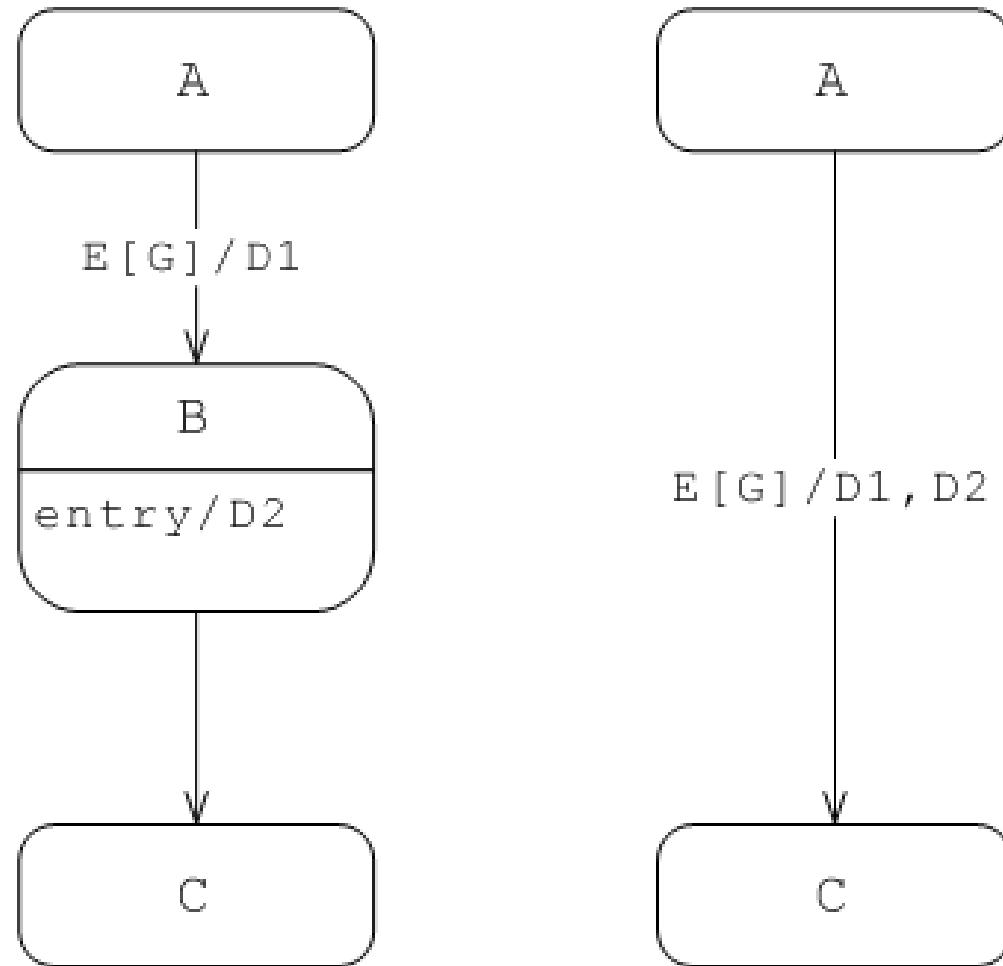
Граф деятельности (activity graph) — это граф, в котором сущности — действия или деятельности, а отношение — порядок их выполнения

- Нагруженный ориентированный граф
- Узлы:
 - узлы действий
 - узлы деятельности
 - узлы управления
 - узлы объектов
- Дуги — потоки управления или данных

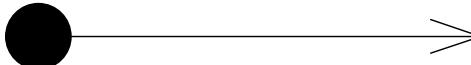
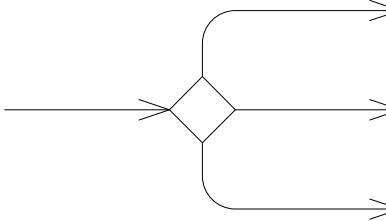
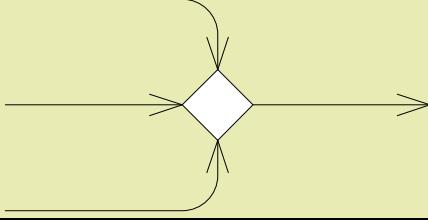
Диаграммы деятельности UML 1

- Состояние действия — это состояние, внутренняя активность которого является действием
- Состояние деятельности — это состояние, внутренняя активность которого является деятельностью
- Объект в состоянии — является аргументом и/или результатом работы некоторого действия или деятельности

Элиминация состояния действия



Узлы управления UML 1 (i)

Название	Изображение	Что обозначает
Начальное состояние (Initial node)		Начало деятельности
Заключительное состояние (Final node)		Завершение деятельности
Разветвление управления (decision node)		Начало альтернативных ветвей деятельности
Объединение управления (merge node)		Конец альтернативных ветвей деятельности

Узлы управления UML 1 (ii)

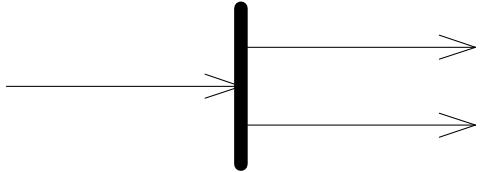
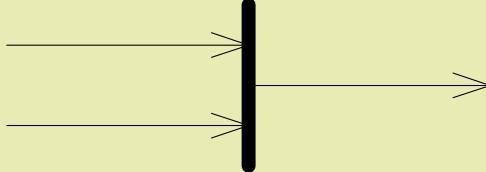
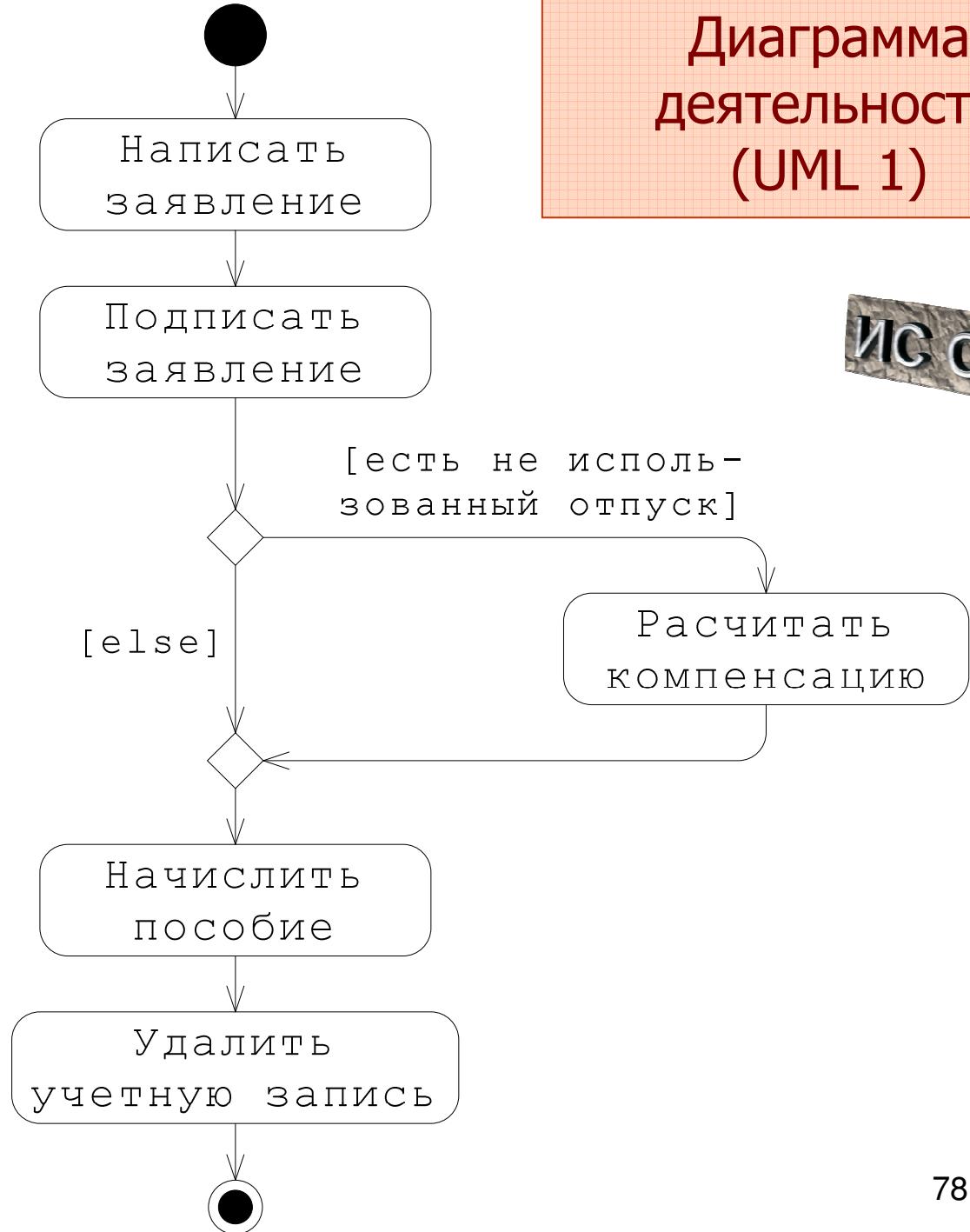
Название	Изображение	Что обозначает
Развилка управления (fork node)		Начало параллельных ветвей деятельности
Слияние управления (join node)		Конец параллельных ветвей деятельности
Посылка сигнала (send)		Действие посылки сигнала
Прием сигнала (accept)		Ожидание события прихода сигнала

Диаграмма деятельности (UML 1)

ИС ОК

Увольнение сотрудника как бизнес-процесс



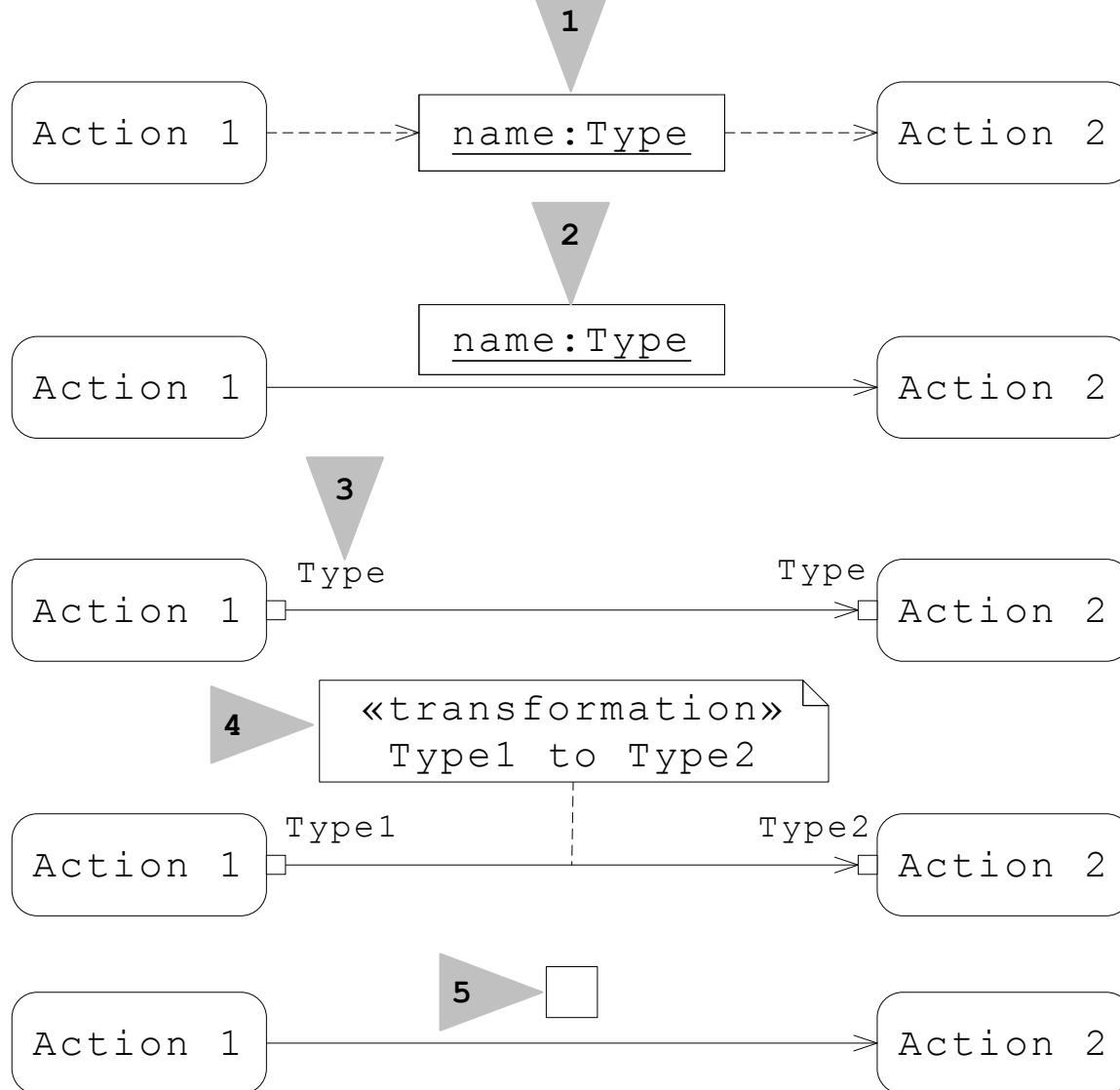
Граф деятельности в UML 2

- Семантика определена через сети Петри
- **Маркер** (token) — абстрактные конструкции:
 - Маркер управления (control flow token) — пустой маркер
 - Маркер данных (data flow token) — содержит ссылку на объект или структуру данных
- **Узлы:**
 - Узлы действий, узлы объектов и узлы управления
 - Области — составные узлы

Типы узлов в UML 2

- Узлы действий:
 - Входные и выходные контакты
- Узлы объектов:
 - Контакт (pin)
 - Параметр деятельности (activity parameter) — контакт на границе деятельности
 - Хранилище данных (data store) — аналог переменной
 - Центральный буфер (central buffer) — имеет несколько входящих и несколько исходящих дуг

Контакты в UML 2



**Контакт (pin) — это
указание аргумента
или результата
действия. Может
иметь тип и имя**

- 1, 2) объект-в-состоянии
UML 1
- 3) тоже но без имени
UML 2
- 4) приведение типов
UML 2
- 5) не специфицированные
данные UML 2

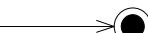
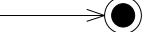
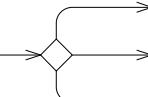
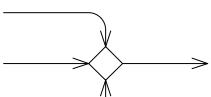
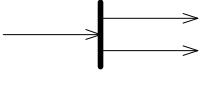
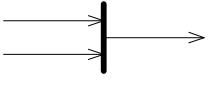
Узлы управления UML 2

Те же, что и в UML 1 +

Название	Изображение	Что обозначает
Заключительное состояние потока		Завершение <u>одного</u> потока управления или данных в деятельности
Комбинированное соединение/разветвление управления		Последовательность из узла соединения и узла разветвления
Комбинированное слияние/развилка управления		Последовательность из узла слияния и узла развилки
Сигнал таймера		Посылка сигнала через определенный промежуток времени

Правила для узлов (i)

Для узлов управления:

-  Начальное состояние создает один маркер управления и все исходящие дуги готовы передать этот маркер
-  Если хотя бы одна входящая дуга заключительного состояния потока готова передать маркер, то заключительное состояние потока поглощает этот маркер
-  Если хотя бы одна входящая дуга заключительного состояния деятельности готова передать маркер, то заключительное состояние деятельности поглощает все маркеры управления и все маркеры данных, завершая, тем самым, выполнение деятельности
-  Если единственная входящая дуга ветвления готова передать маркер (управления или данных), то те исходящие дуги ветвления, на которых сторожевые условия выполняются, готовы передать этот маркер
-  Если любая входящая дуга соединения готова передать маркер (управления или данных), то единственная исходящая дуга соединения готова передать этот маркер.
-  Если единственная входящая дуга разветки готова передать маркер (управления или данных), то все исходящие дуги готовы одновременно передать копии этого маркера.
-  Если все входящие дуги слияния готовы передать маркеры (управления или данных), то исходящая дуга готова передать маркер управления. Слияние обеспечивает синхронизацию потоков

Правила для узлов (ii)

Для узлов объектов:

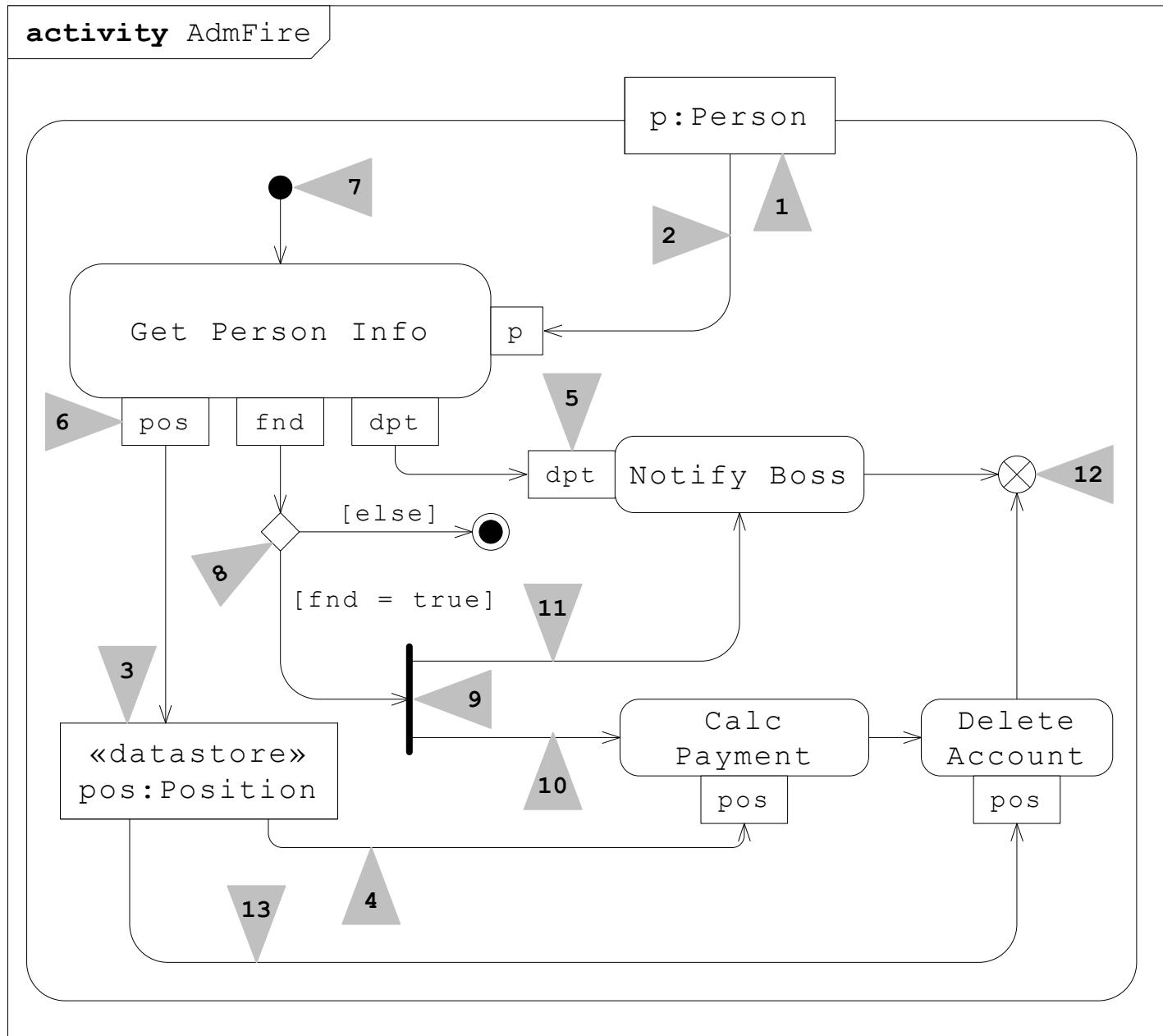
- Параметр деятельности создает один маркер данных и все исходящие дуги готовы передать этот маркер
- Хранилище данных поглощает один маркер данных и создает неограниченное количество его копий. Все выходные дуги хранилища всегда готовы передать копию хранимого маркера данных
- Центральный буфер перенаправляет маркеры данных, не создавая и не поглощая их. Как только входная любая входная дуга готова передать маркер, все выходные дуги готовы передать этот маркер
- Входной контакт действия поглощает маркер данных
- Выходной контакт действия создает маркер данных

Для контактов и действий

- Если все дуги данных, входящие во все входные контакты действия, готовы передать маркеры данных, и если есть входящие дуги управления и хотя бы одно готово передать маркер управления, то действие выполняется
- Если действие выполнено, то все дуги данных, выходящие из выходных контактов, готовы передать маркеры данных, и если есть исходящие ребра управления, то те исходящие дуги управления, на которых сторожевые условия выполняются, готовы передать маркер управления

Диаграмма деятельности UML 2

ис ок



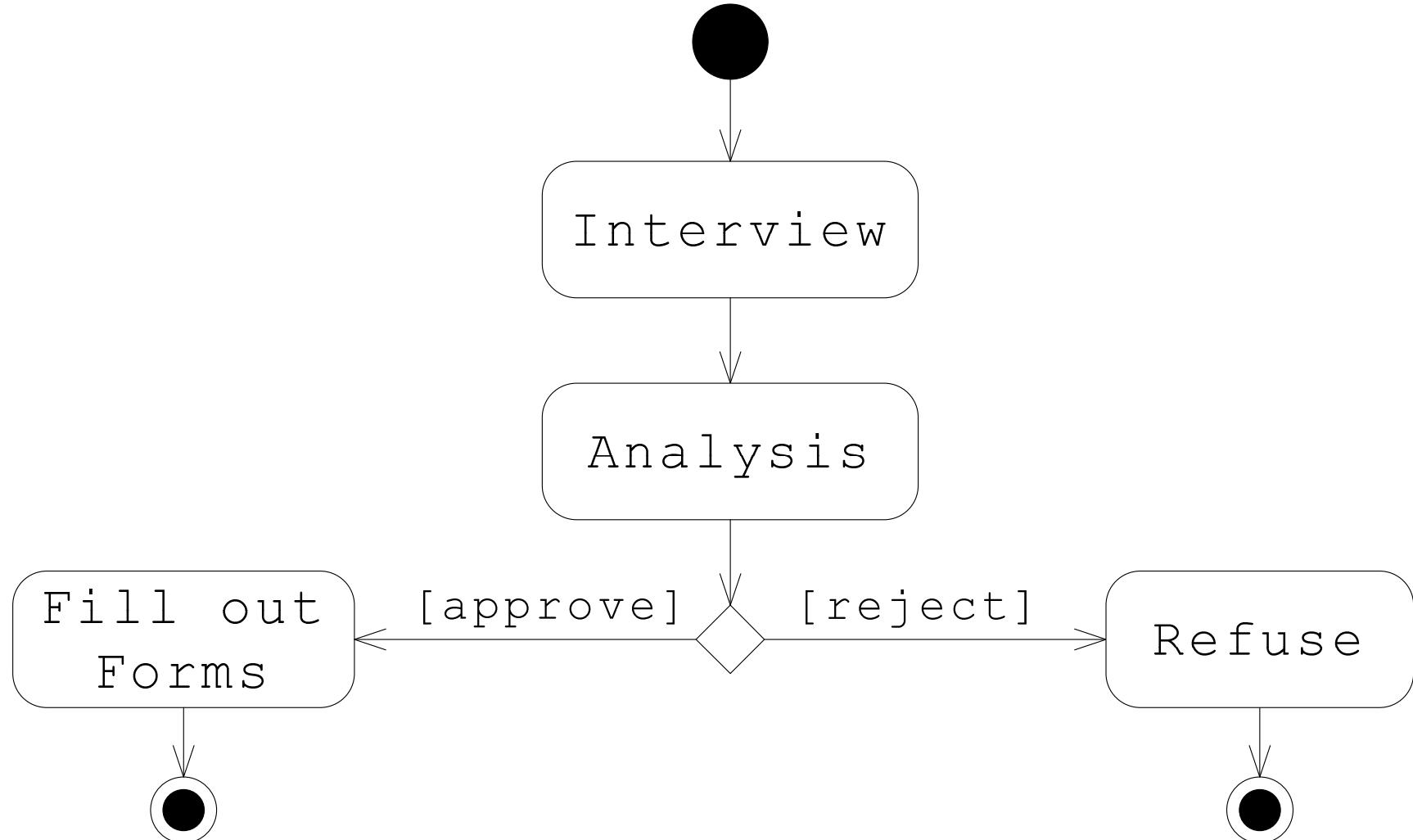
- 1) параметр деятельности
- 2, 4, 10, 11, 13) переход
- 3) хранилище данных
- 5) входной контакт действия
- 6) выходной контакт действия
- 7) начальное состояние
- 8) ветвление
- 9) разветвка
- 12) заключительное состояние потока

Дорожки и разбиения

Дорожка (swim lane) — в UML 1 это графический комментарий, позволяющее классифицировать по некоторому признаку сущности

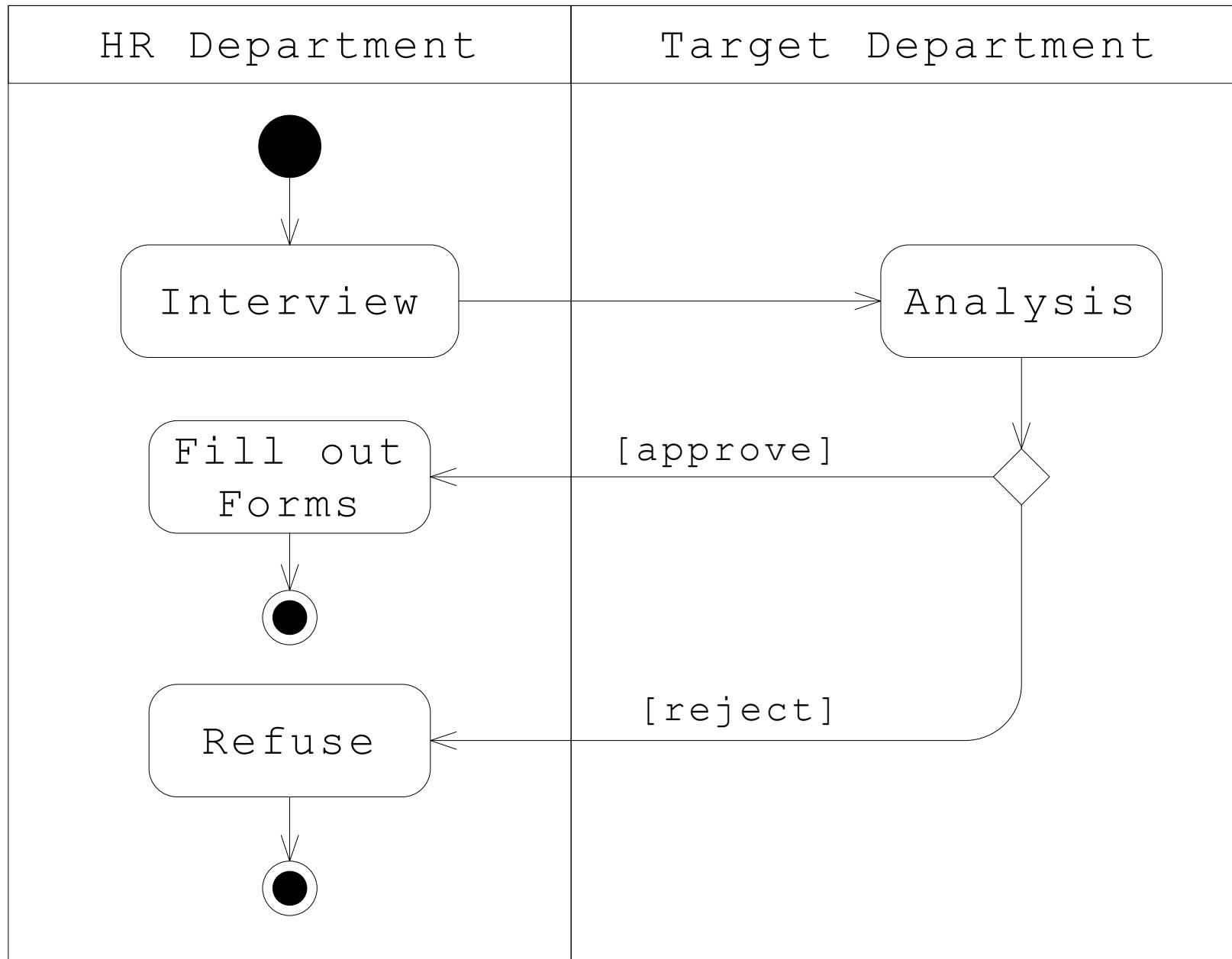
Разбиение (partition) — в UML 2 это разбиение в математическом смысле (то есть дизъюнктное покрытие) множества сущностей на диаграмме

Процесс найма на работу



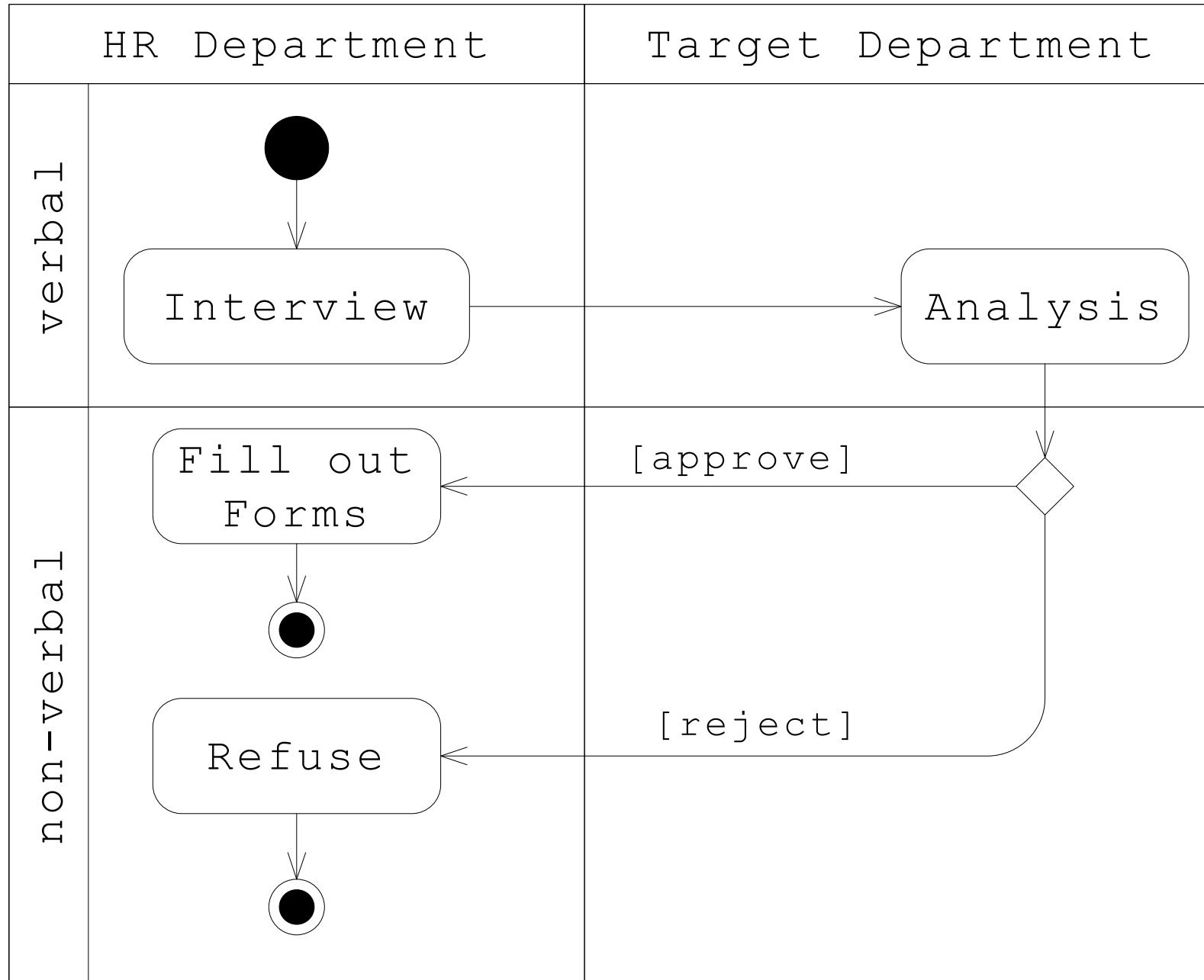
Дорожки

ИС ОК



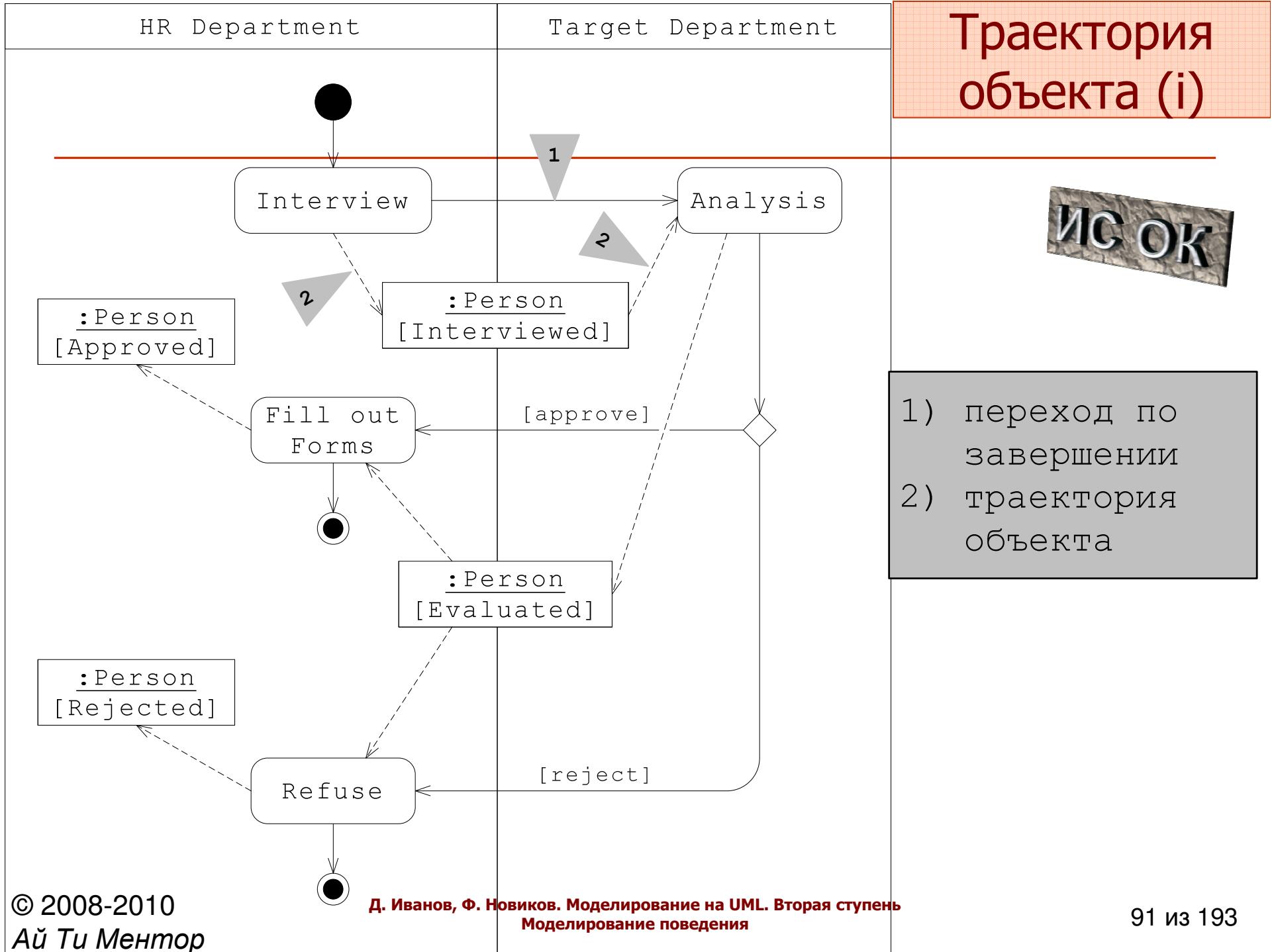
Ортогональные дорожки

исок



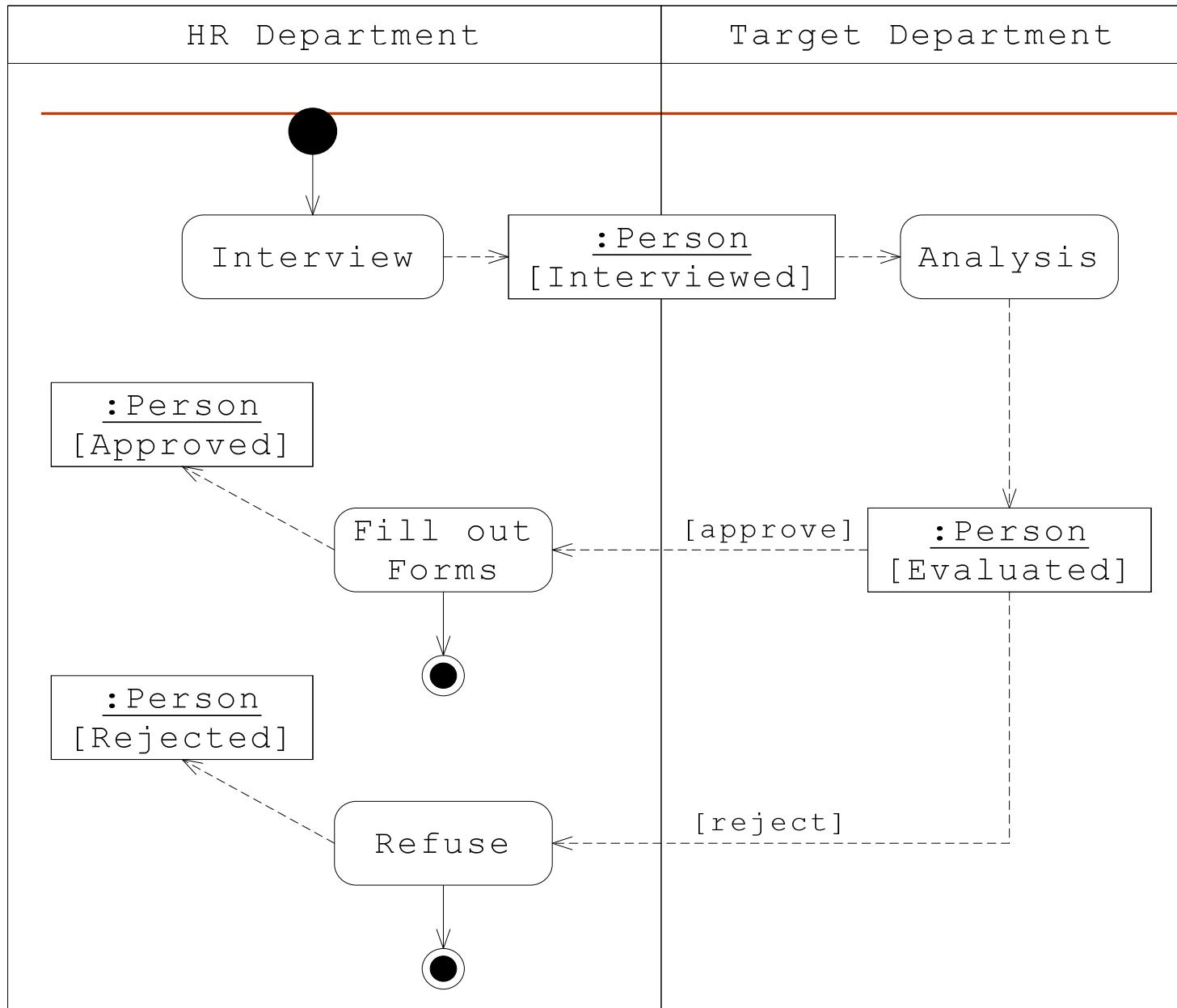
Траектория объекта и поток данных

- **Объект в состоянии** (object in state) — находится в определенном состоянии в данной точке вычислительного процесса
- **Траектория объекта** — это переход особого рода, исходным или целевым состоянием которого является объект в состоянии
 - от состояния деятельности к объекту в состоянии:
→ объект в данном состоянии является результатом деятельности
 - от объекта в состоянии к состоянию деятельности:
→ объект в данном состоянии является аргументом деятельности



Траектория объекта (ii)

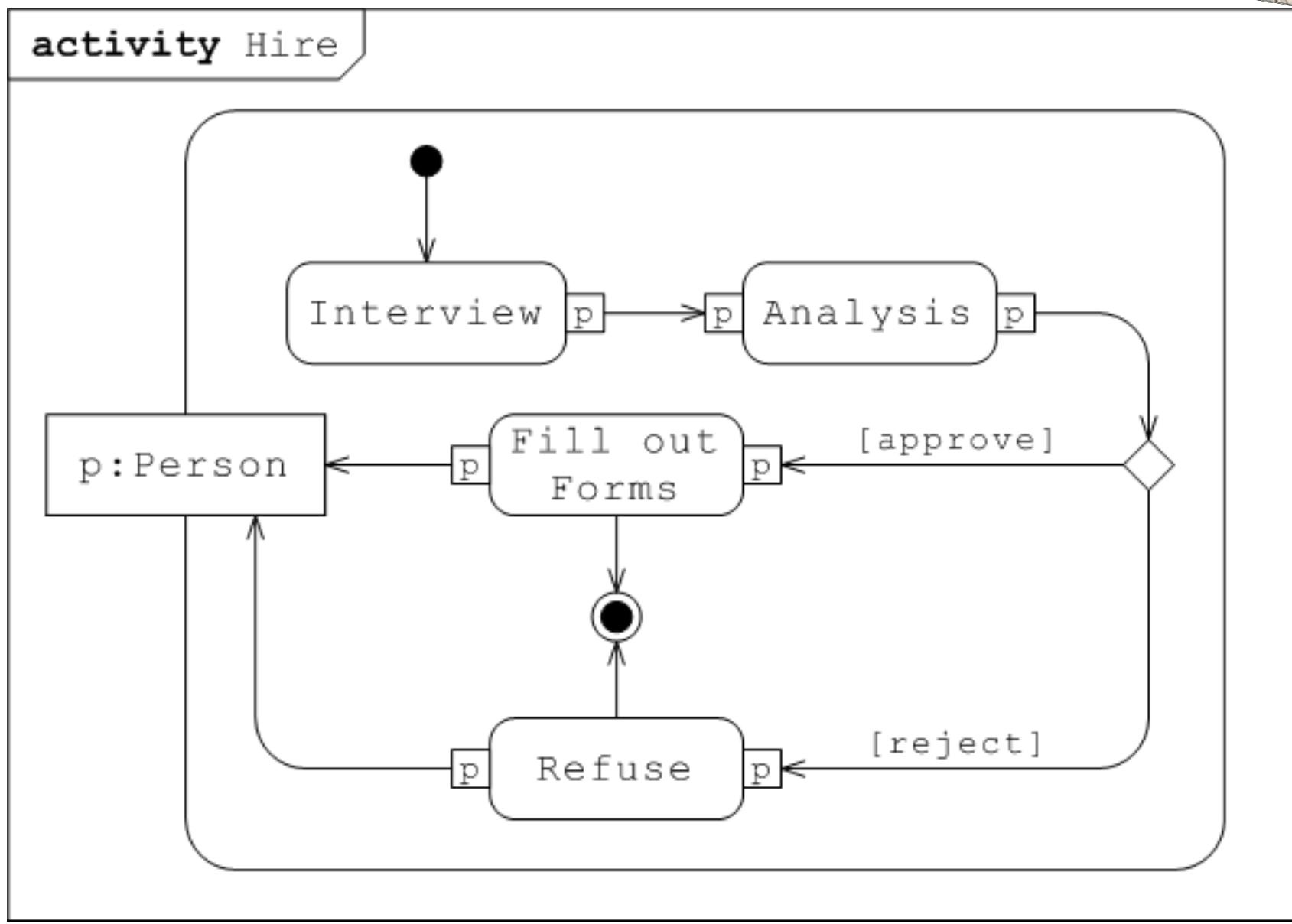
ИС ОК



ТРАКТОРИЯ ОБЪЕКТА

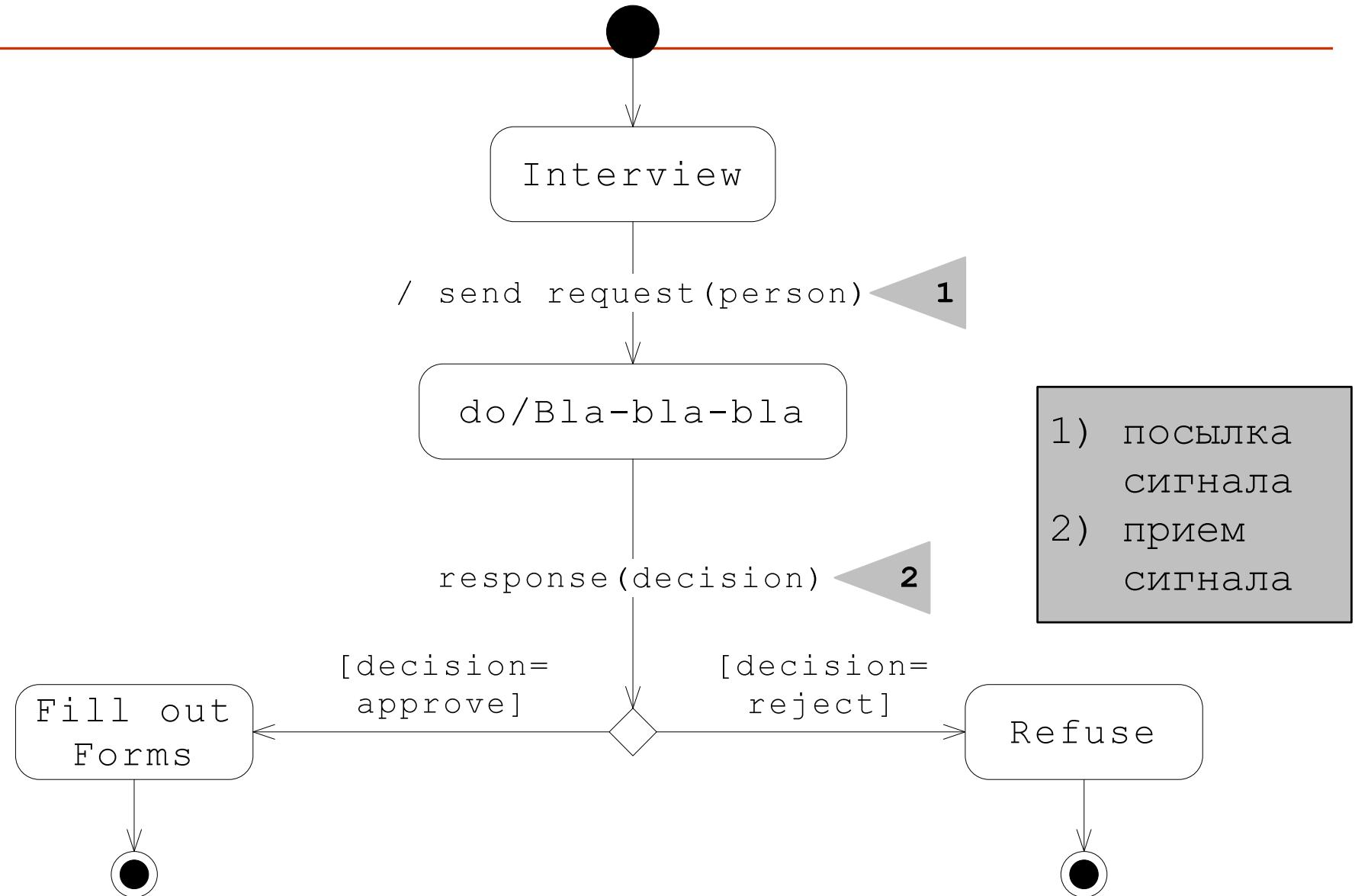
(iii)

ИС ОК



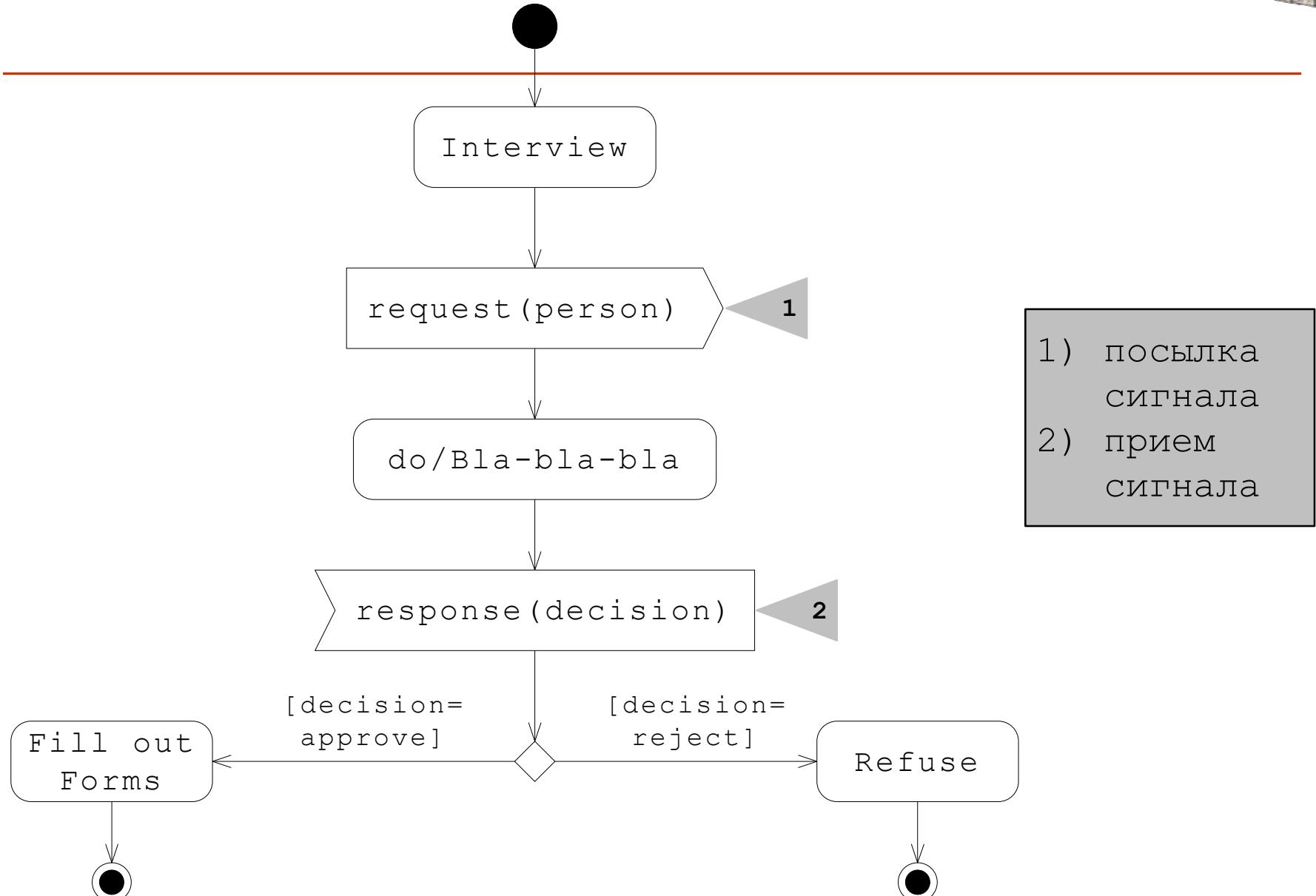
Асинхронный процесс принятия решения (i)

ис ок



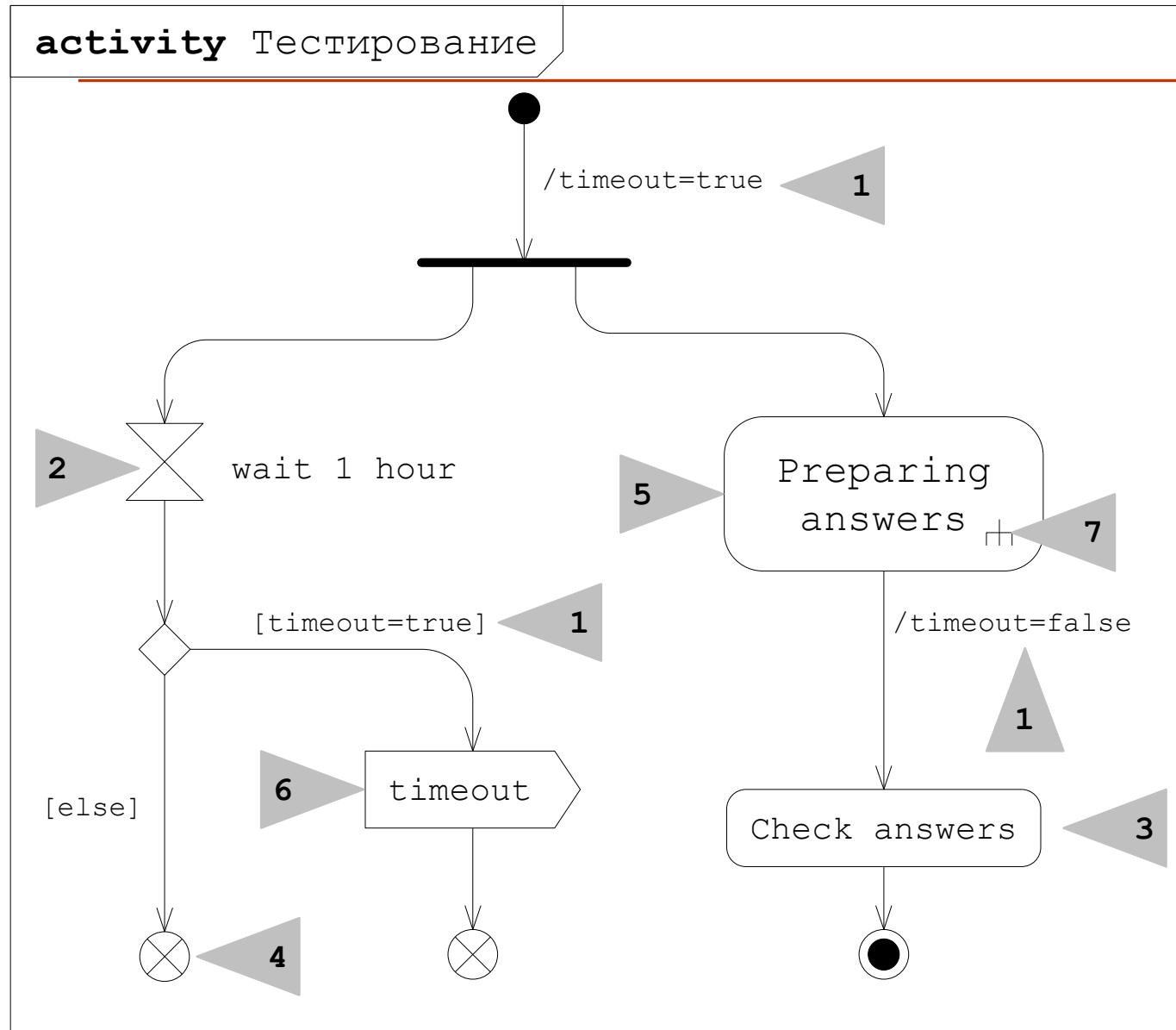
Асинхронный процесс принятия решения (ii)

ис ок



Прием сигнала от таймера

ис ок



Сравнение способов описания поведения

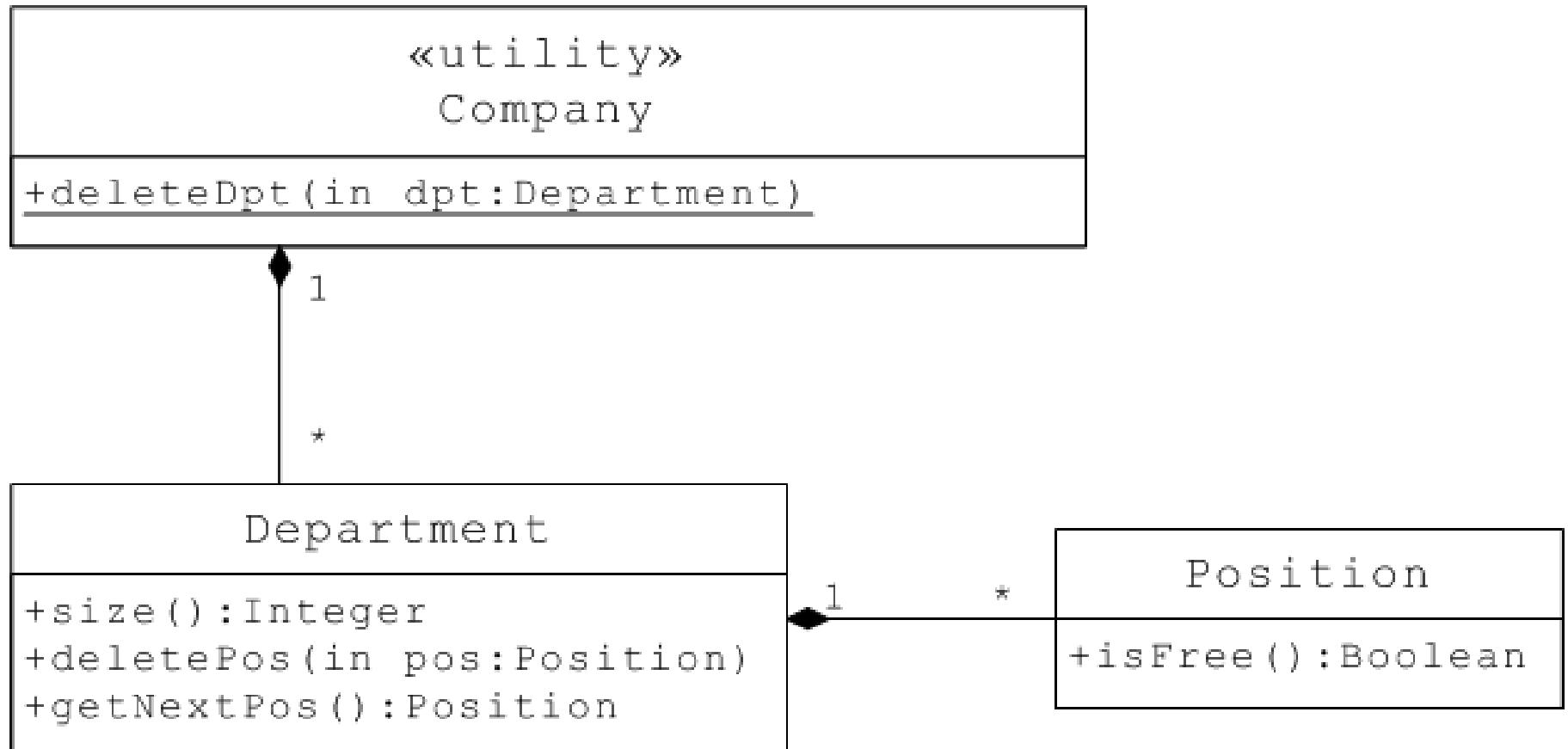
- Реализация операции `deleteDepartment`
 - При удалении подразделения, если нет “живых” людей, то удаляются все вакантные должности и подразделение. В противном случае – исключение

Способы описания:

1. Наивное программное решение
2. Эквивалентная блок-схема
3. Графический рефакторинг
4. Использование состояния с внутренней деятельностью
5. Автоматная программа



Некоторые классы

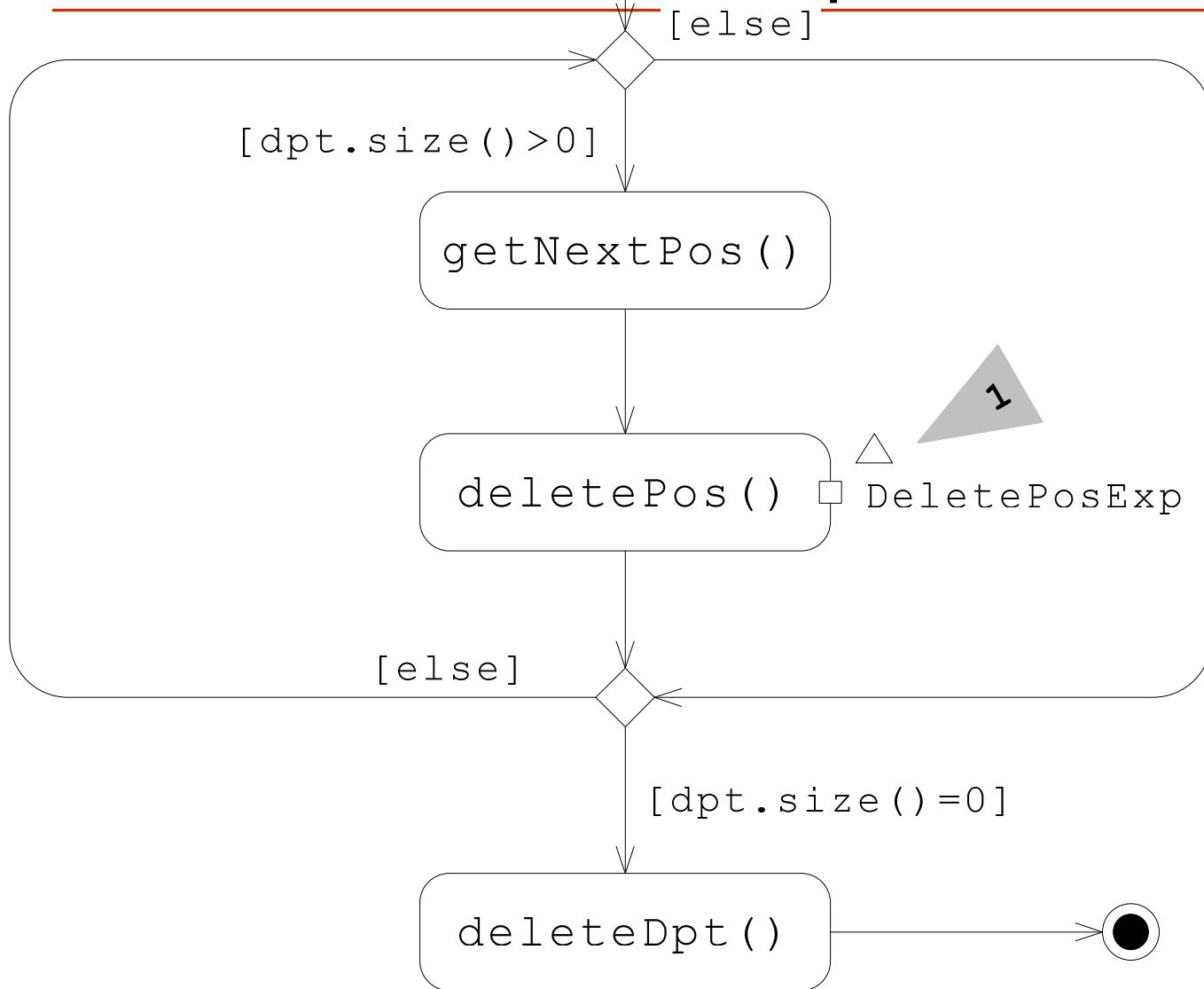


Наивное программное решение

```
if dpt.size() > 0
repeat
    pos := dpt.getNextPos()
    dpt.deletePos(pos)
        // возможна генерация
        // исключения DeletePosExpr
until dpt.size() = 0
deleteDpt(dpt)
```



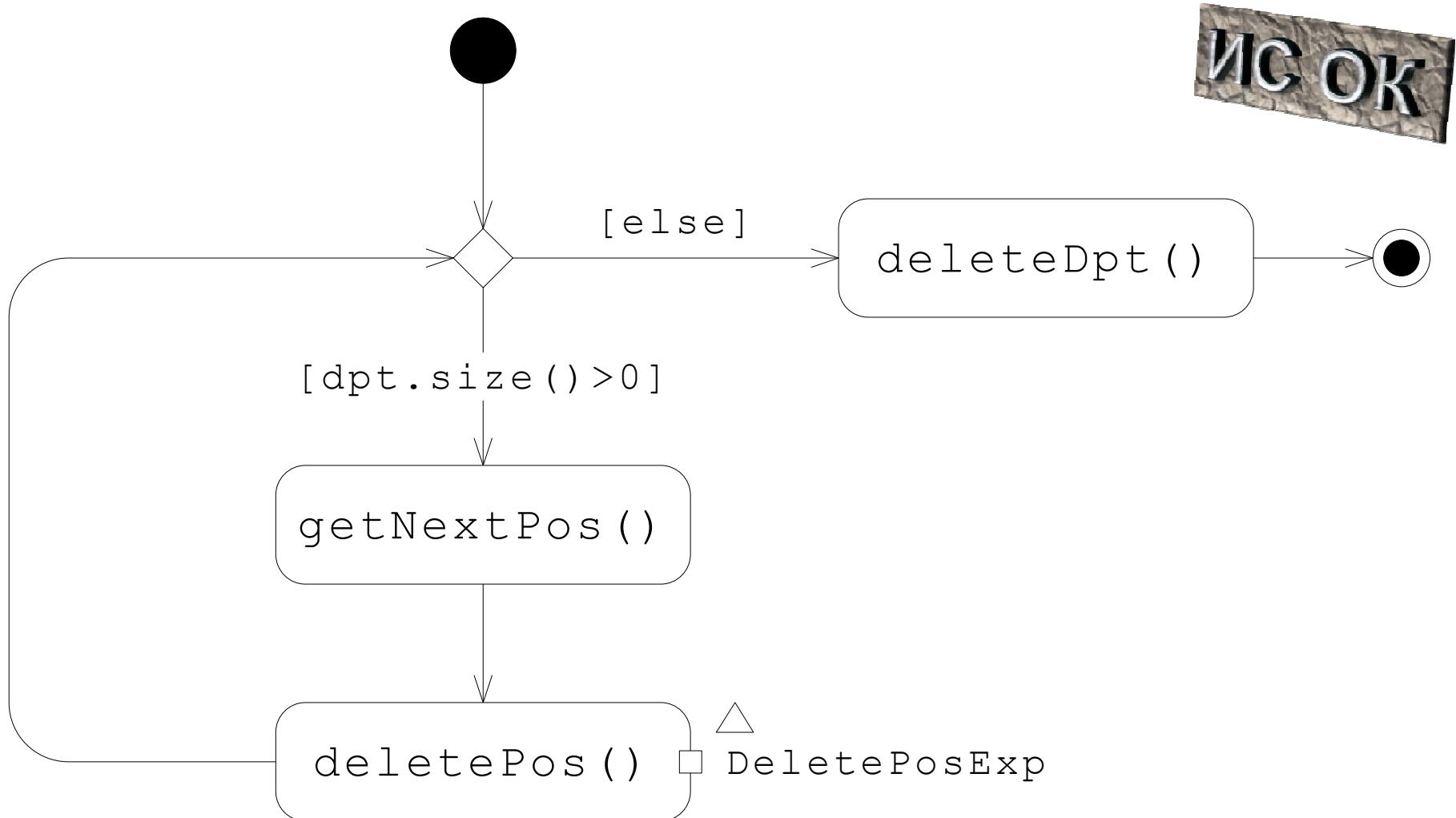
Эквивалентная диаграмма деятельности



ИС ОК

- 1) значок контакта исключения

Результат графического рефакторинга



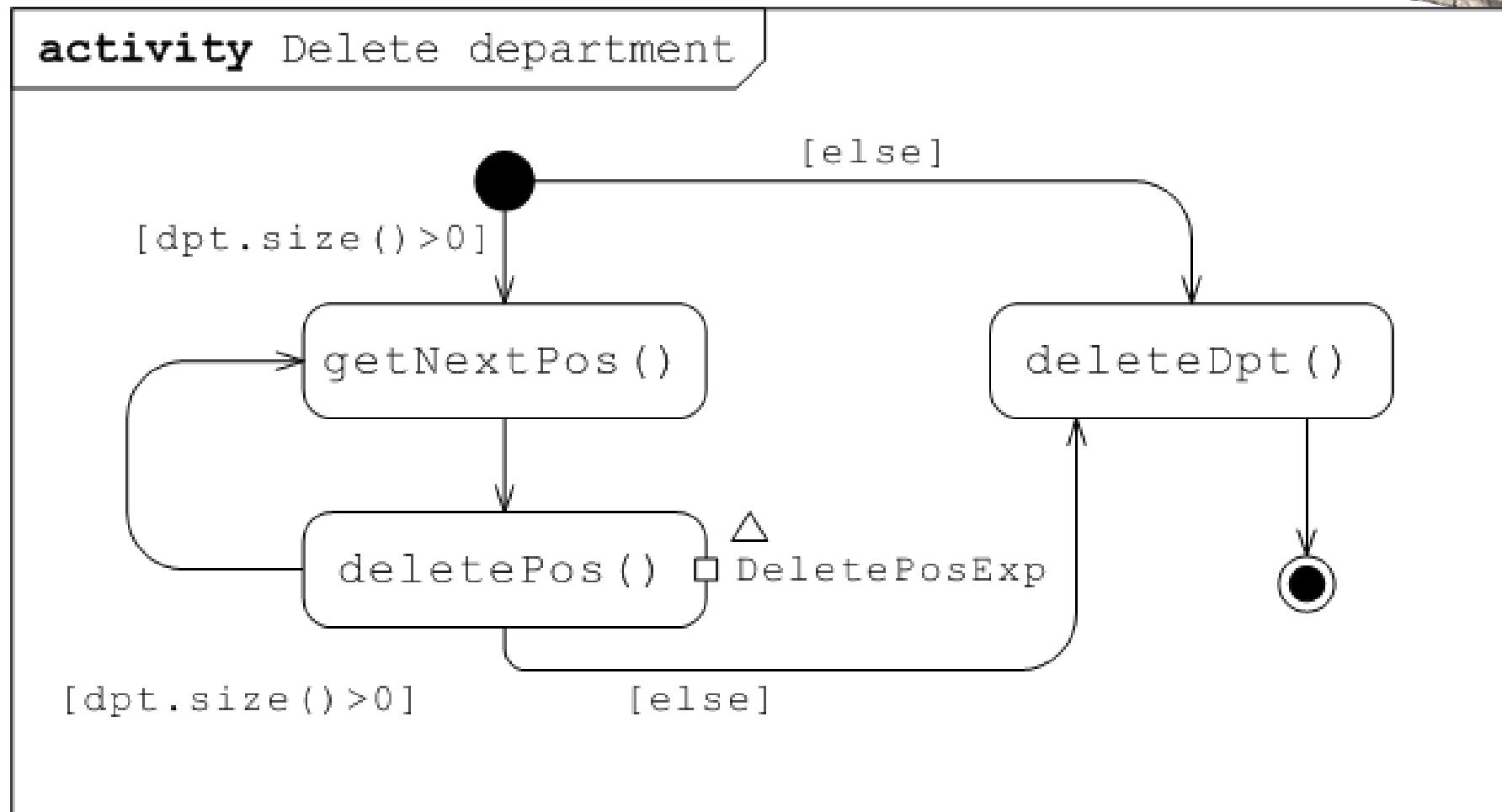
Соответствующий программный код

```
while dpt.size() > 0  
    pos := dpt.getNextPos()  
    dpt.deletePos(pos)  
        // возможна генерация  
        // исключений DeletePosExpr  
deleteDpt(dpt)
```



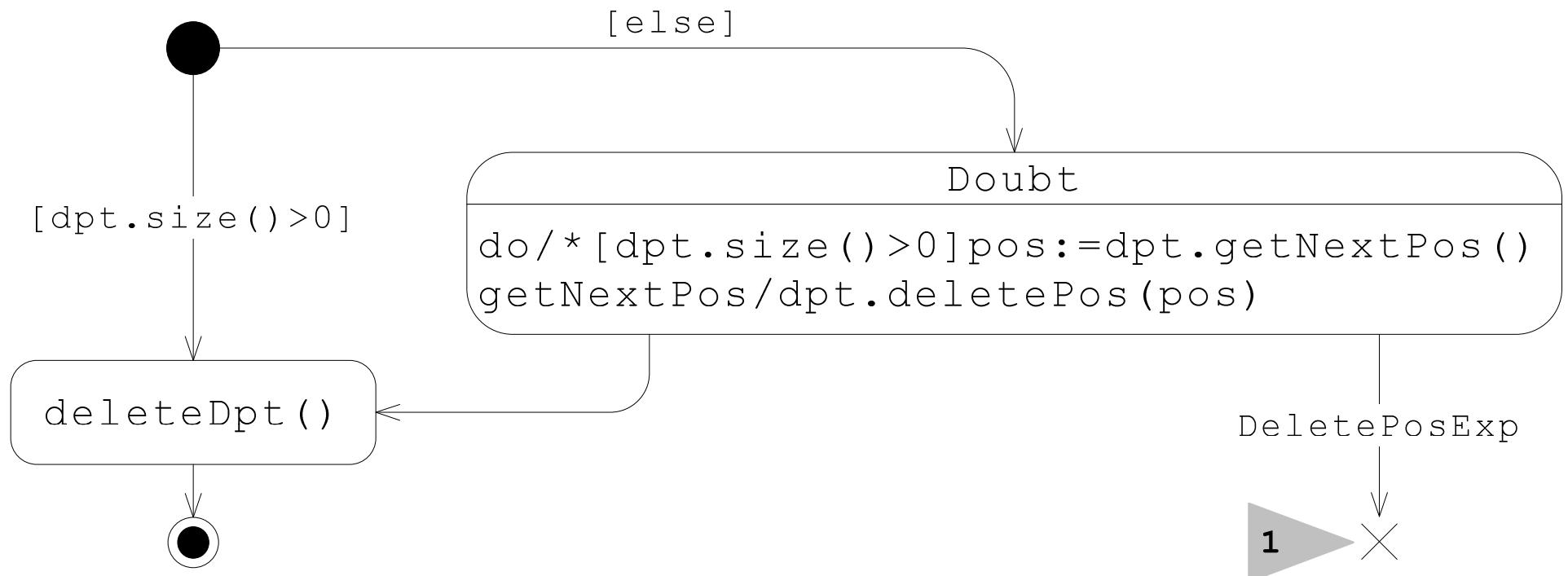
Элиминация состояний_выбора

ИС ОК



Использование простого состояния

ис ок



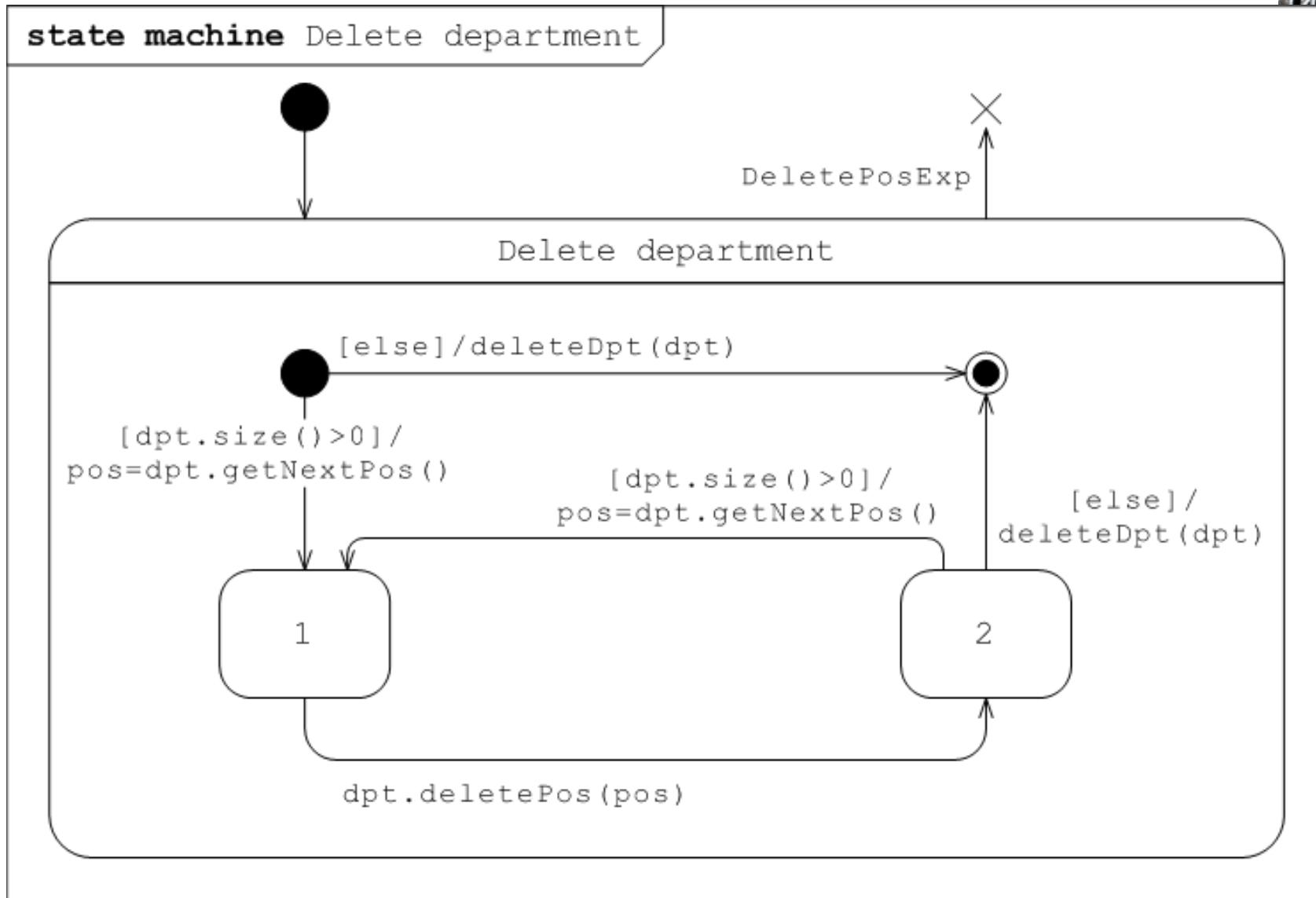
Повторитель

Повторитель (iteration) — это выражение, предписывающее выполнить действие несколько раз (может быть, ни разу)

* [повторитель] действие

- В UML 1 нечто вроде заголовка цикла целевого языка программирования
- В UML 2 устаревшая конструкция, но не запрещается

Эквивалентный конечный автомат



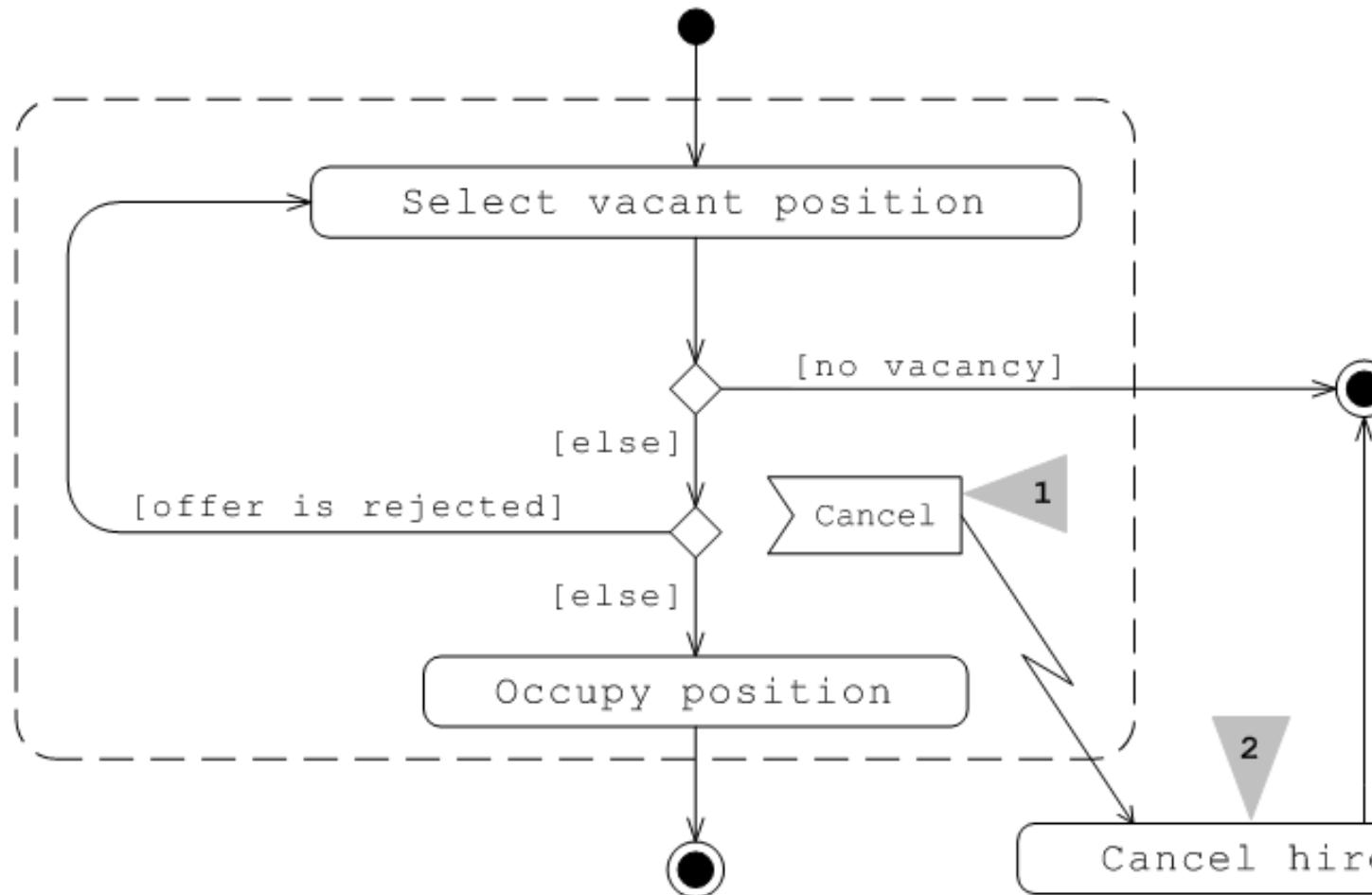
Прерывания

Область прерывания (interruptible activity region) — рамка, внутри которой возможно прерывание обычного порядка выполнения при возникновении определенного события

- Прерывающие события — прием сигнала
- Стрелка-«молния» (interruptible activity edge)
- Внешняя деятельность — **обрабочику прерывания** (interrupt handler)

Область прерывания

activity Прием сотрудника на работу (Hire Person)

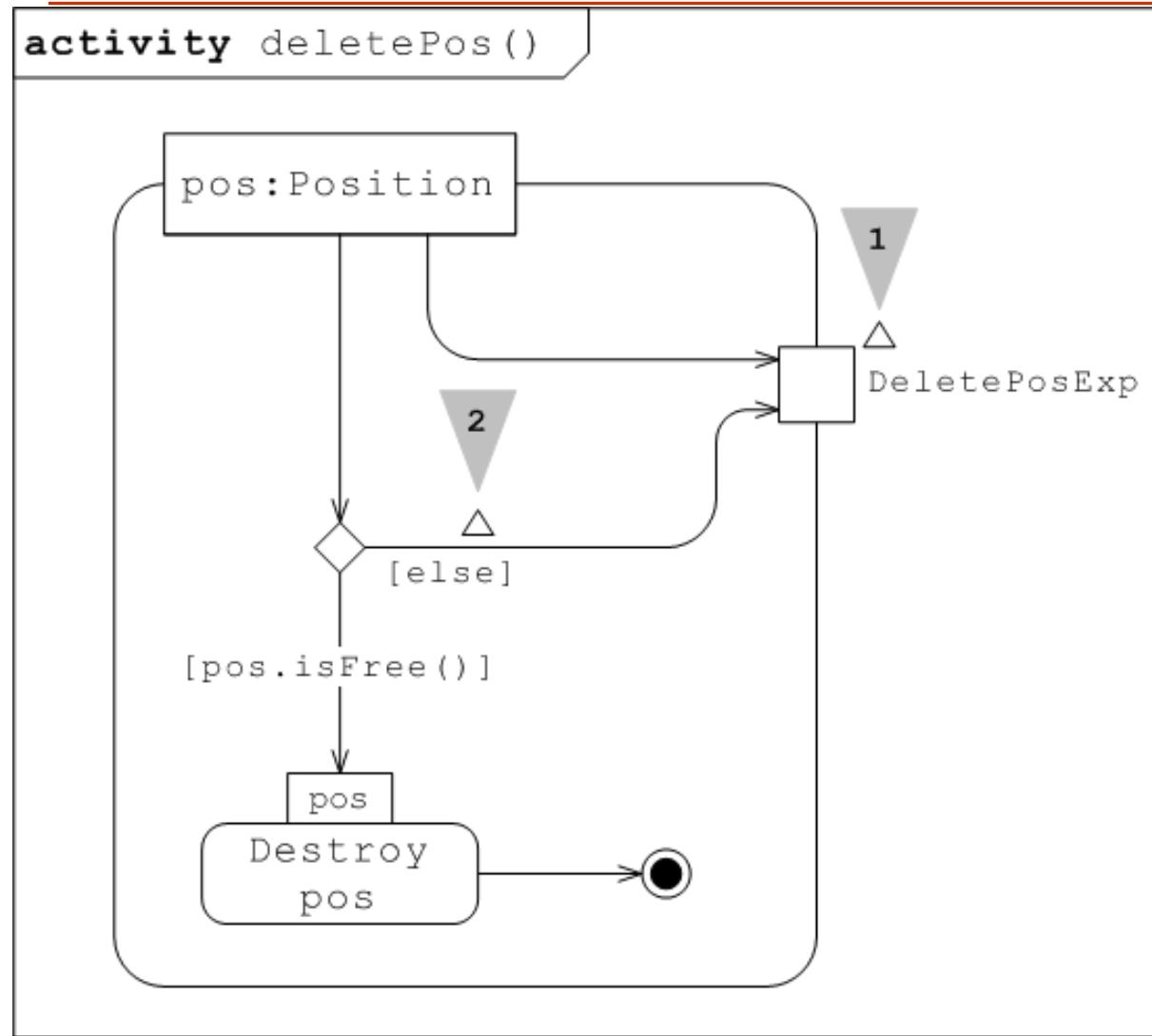


Исключения

Прекращение текущей деятельности при
возникновении события и передача управления

- Распространение исключения – передача исключения на верхние уровни до тех пор, пока не найден обработчик
- Иерархия исключений – являются классификаторами и могут образовывать иерархии обобщения
- Параметры исключения – атрибуты, заполняются требуемыми значениями
- Сохранение выходной сигнатуры – когда исключение обработано, программа выполняется «как будто ничего и не было»

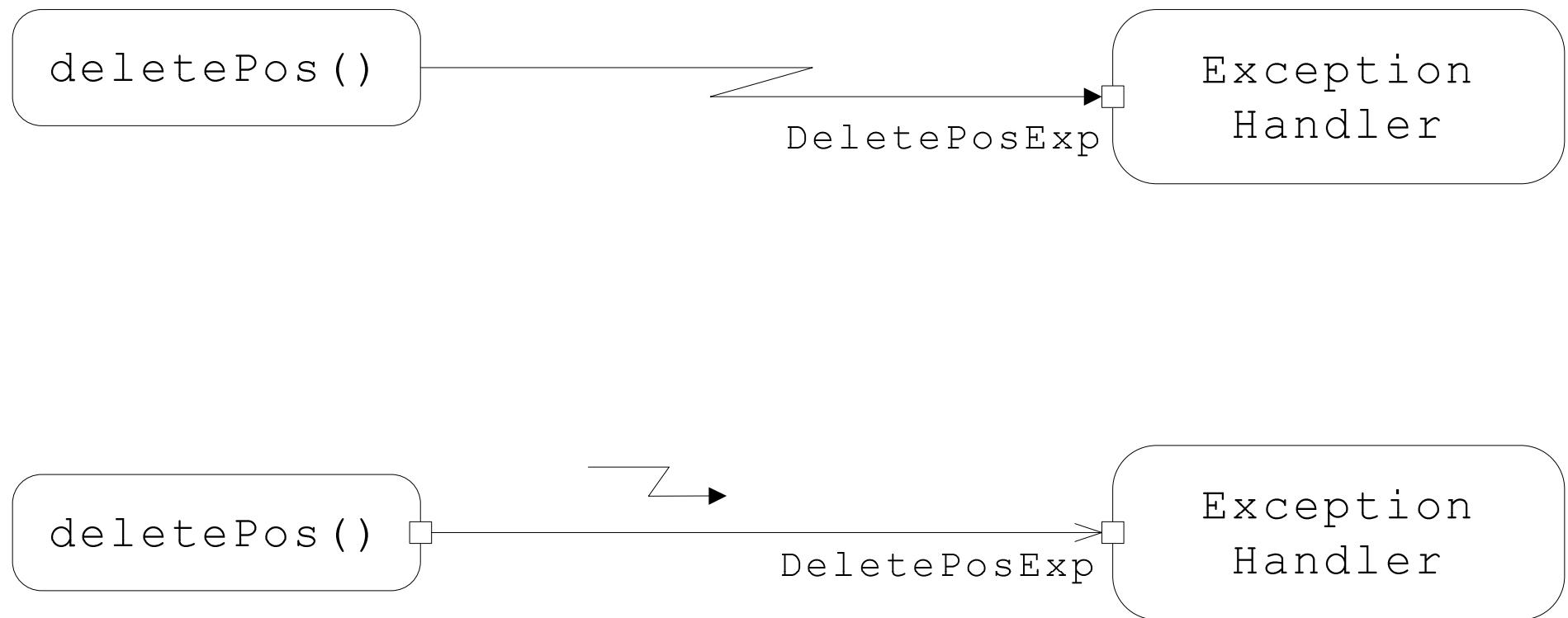
Диаграмма деятельности DeletePos



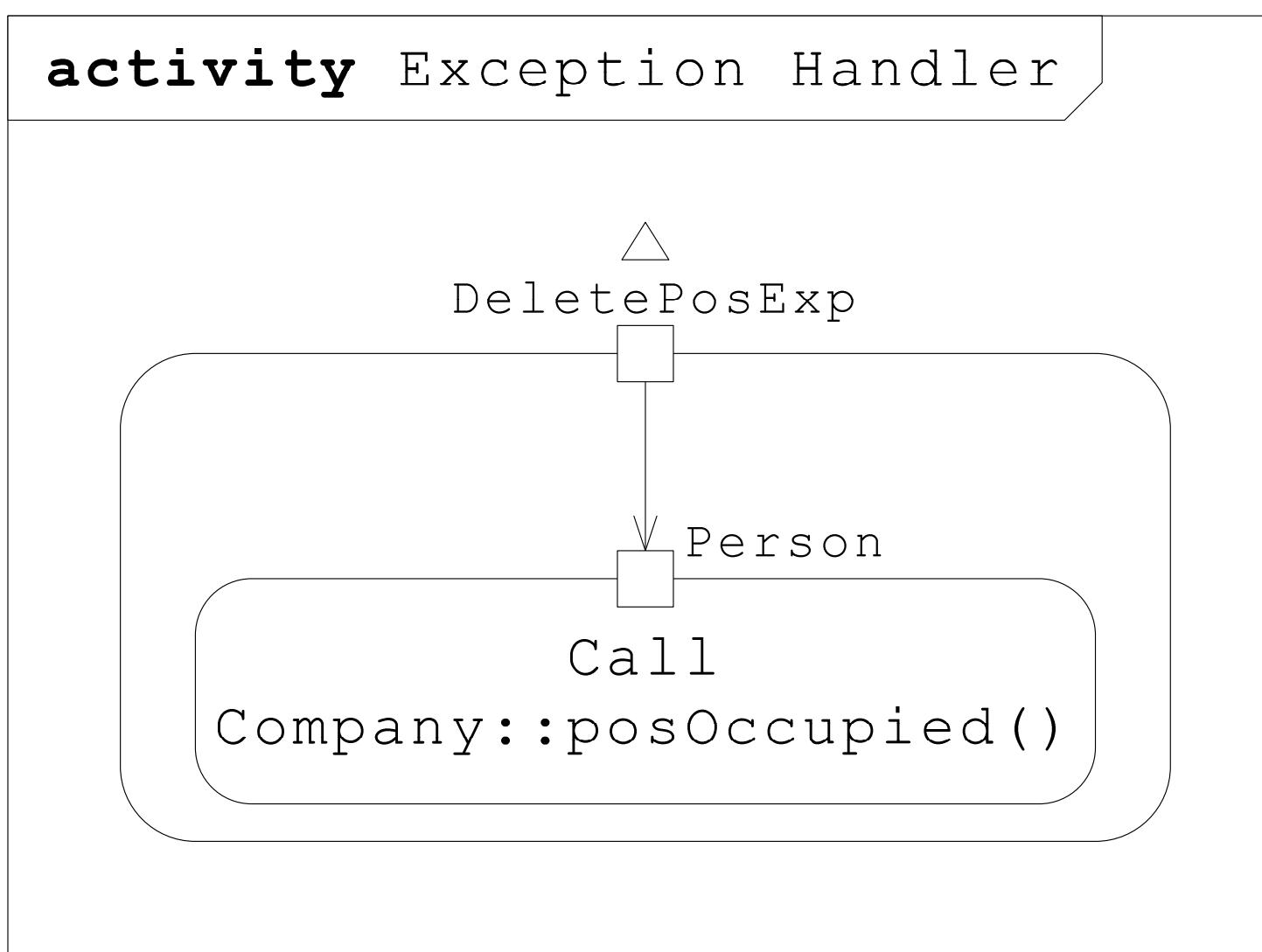
ИС ОК

- 1) значок исключения у контакта
- 2) значок исключения на дуге

Варианты нотации стрелки-“молнии”



Деятельность Exception Handler

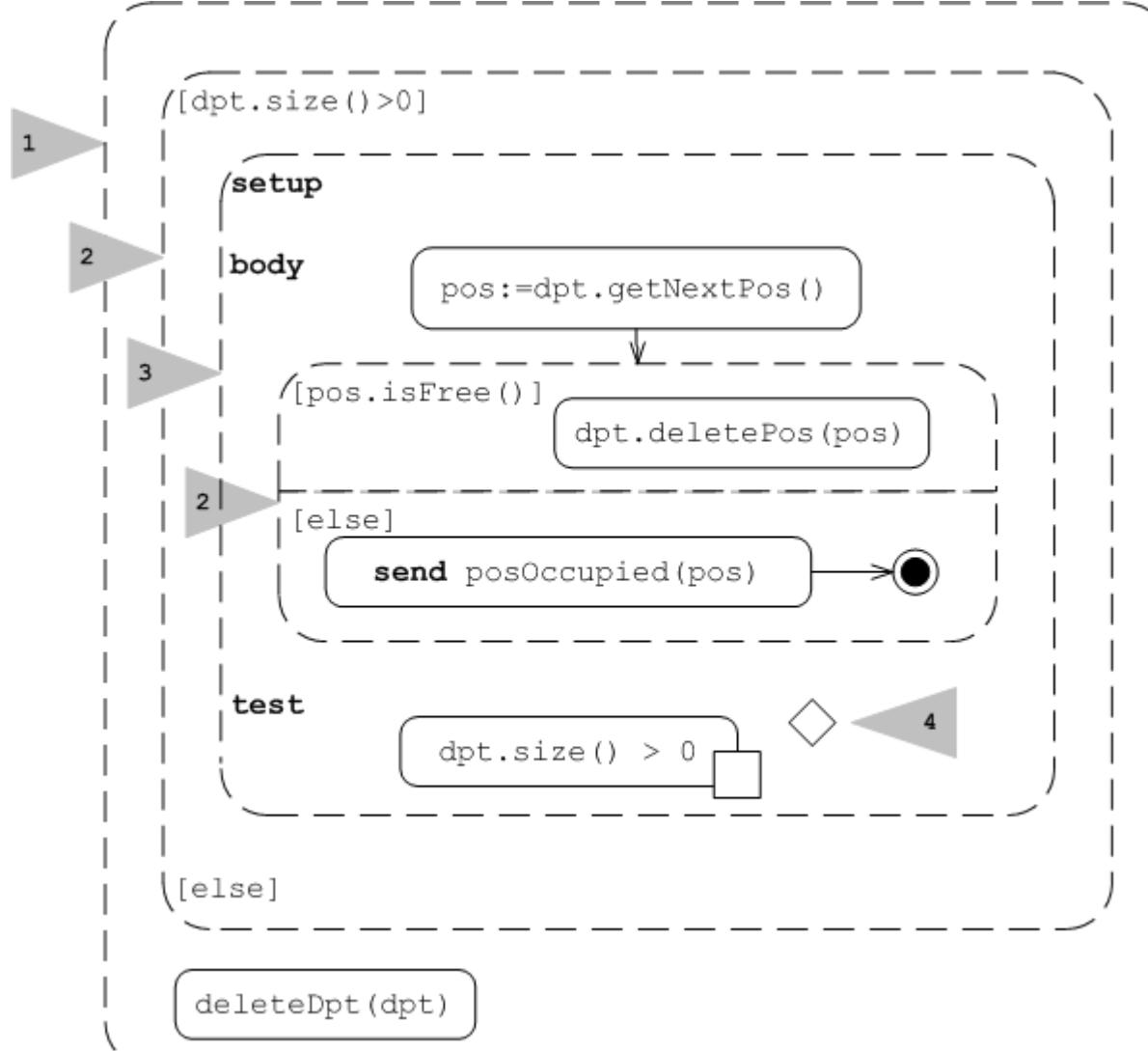


ис ок

Структурные узлы деятельности

- Структурный узел деятельности – область на диаграмме для задания структур управления
- Есть в метамодели, определена семантика, нет примеров
- Классические структуры:
 - Узел последовательности (;)
 - Узел условий (**if then else**)
 - Узел цикла (**while repeat**)

Деятельность Delete Department



ИС ОК

- 1) узел последовательности
- 2) узел условий
- 3) узел цикла

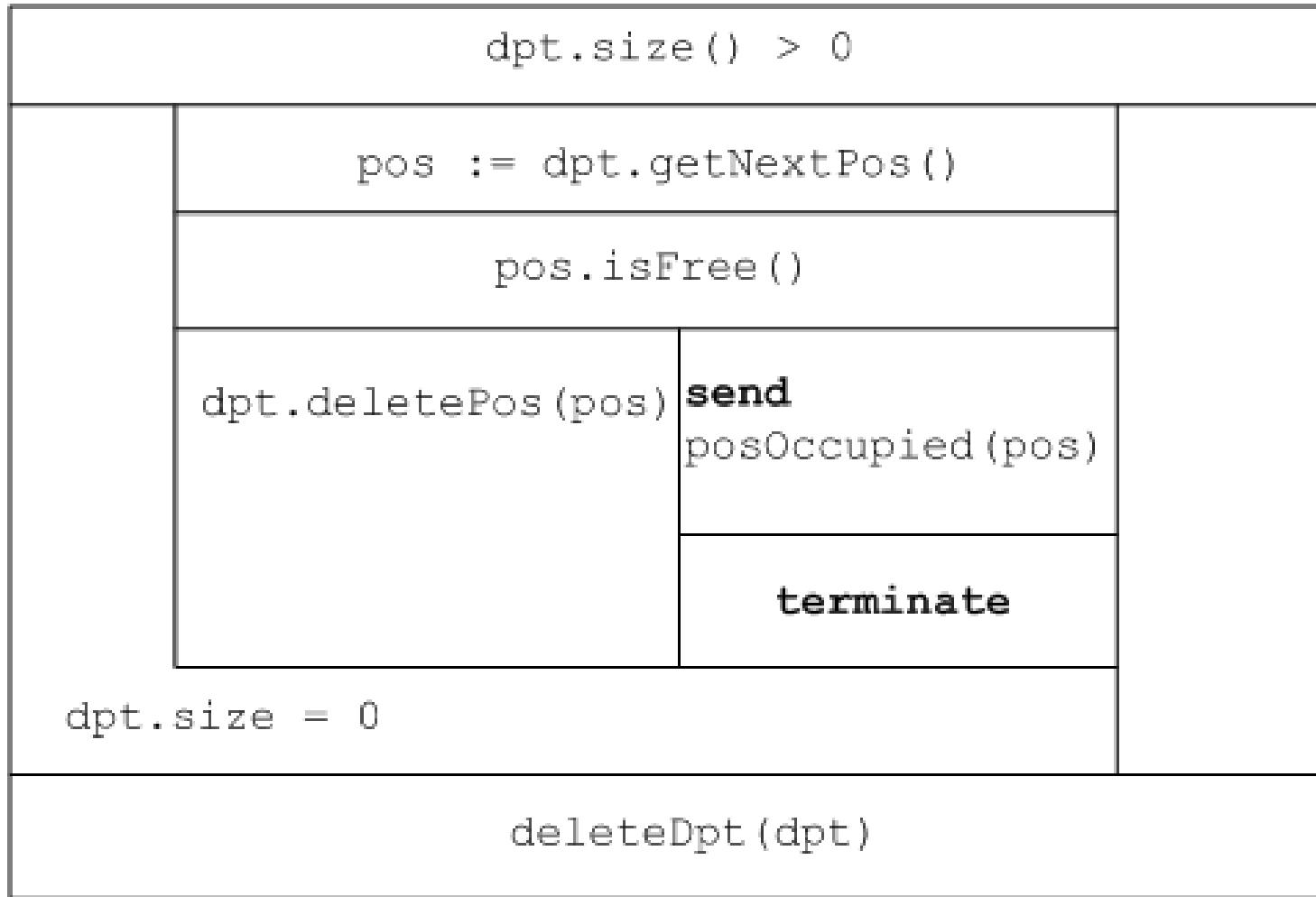
Сравнение различных нотаций (i)

Структура управления	Блок-схема	Нэсси Шнейдерман	Узлы UML
Последовательность S1 ; S2	<pre> graph TD S1[S1] --> S2[S2] </pre>	<pre> graph TD S1[S1] --- S2[S2] </pre>	<pre> classDiagram node S1 node S2 S1 .. S2 </pre>
Ветвление if B then S1 else S2 end if	<pre> graph TD start(()) --> B{B} B -- Да --> S1[S1] B -- Нет --> S2[S2] S1 --> merge(()) S2 --> merge </pre>	<pre> graph TD B[B] --- S1[S1] B --- S2[S2] </pre>	<pre> classDiagram node S1 node S2 [B] .. S1 [else] .. S2 </pre>

Сравнение различных нотаций (ii)

Структура управления	Блок-схема	Нэсси Шнейдерман	Псевдокод
Цикл с предусловием while B S end while	<pre> graph TD B{B} -- Да --> S[S] S --> B B -- Нет --> End[] </pre>	<pre> graph TD B[B] --- S[S] </pre>	<pre> setup test B body { S } </pre>
Цикл с постусловием repeat S until B	<pre> graph TD S[S] --> S B{B} -- Да --> S B -- Нет --> End[] </pre>	<pre> graph TD B[B] --- S[S] </pre>	<pre> setup body { S } test B </pre>

Деятельность Delete Department



ис ок

Применение диаграмм деятельности

- Применяются для описания поведения на самых *разных* уровнях абстракции
- Средство описания бизнес-процессов
- Средство описания алгоритмов
- Средство визуального структурного программирования

Совет: использовать для операций,
выполнение которых непосредственно
реализует вариант использования

4. Диаграммы взаимодействия

- Моделируют поведение, описывая **взаимодействие** объектов:
 - экземпляров классов, действующих лиц, вариантов использования, подсистем, компонентов, узлов
 - на уровне **ролей**, а не на уровне экземпляров
- Моделируют поведение «по индукции», от частного к общему
- Взаимодействие происходит путем обмена **сообщениями**
- «Ближе» к реальному выполнению программы

Виды диаграмм взаимодействия

- Диаграммы последовательности и диаграммы коммуникации — семантически эквиваленты
- Сущности: экземпляры классификаторов
- Отношения: ассоциации, по которым передаются сообщения
- Обзорные диаграммы взаимодействия — смесь диаграмм деятельности и последовательности
- Диаграммы синхронизации — смена состояний объекта во времени

Сообщения

предшественники / повторность номер :
атрибуты = ИМЯ (аргументы) : переменные

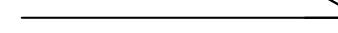
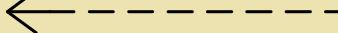
Сообщение (message) — передача управления и
данных от отправителя к получателю

- Отправка сообщения — действие
- Получение сообщения — событие
- Широковещательное сообщение (broadcast),
т.е. имеется несколько получателей
 - указывается с помощью повторителей
- Условное сообщение: со сторожевым условием

Действия отправки сообщения

- В UML 1:
 - вызов метода (call)
 - создание объекта (create)
 - уничтожение объекта (destroy)
 - возврат значения (return)
 - посылка сигнала (send)
- UML 1 переменные := ИМЯ (аргументы)
- UML 2
атрибуты = ИМЯ (аргументы) : переменные

Передача управления

Вид стрелки	Тип передачи сообщения
	Вложенный поток управления Сообщение отправляется после завершения всех инициированных действий
 Только UML 1	Простой поток управления Управление передается от отправителя получателю
 	Асинхронный поток управления Сообщение асинхронно передается, при этом у отправителя сохраняется свой поток управления
	Возврат управления Возврат управления после выполнения всех действий, инициированных передачей сообщения
Не определяется	Допускается использование при моделировании других типов передачи управления

Порядок сообщений

- На диаграмме последовательности:
 - Определяется **временем отправки**
 - Время считается текущим на диаграмме **сверху вниз**
- На диаграмме коммуникации:
 - Определяется **номером сообщения**
 - Обычная порядковая или иерархическая нумерация
- На диаграмме синхронизации:
 - Определяется **временем отправки**
 - Время считается текущим на диаграмме **слева направо**
- На обзорной диаграмме взаимодействия:
 - Определяется **потоком управления**

Диаграммы последовательности

- Моделируют поведение в форме описания **протокола**
 - сеанса обмена сообщениями
 - между взаимодействующими экземплярами классификаторов
 - во время выполнения сценария
- На диаграмме: только те экземпляры, связи и сообщения, которые участвуют во взаимодействии

Связь

Связь (link) — это экземпляр отношения
= набор ссылок на экземпляры
классификаторов, связанных этим
отношением

- Постоянные связи — экземпляры ассоциаций
- Временные связи могут ими не быть...

Линия жизни

- **Линия жизни** (lifeline): по оси времени
- Представляет объект во взаимодействии:
 - отправляет сообщение
 - получает сообщение
 - начинает существование
 - заканчивает существование

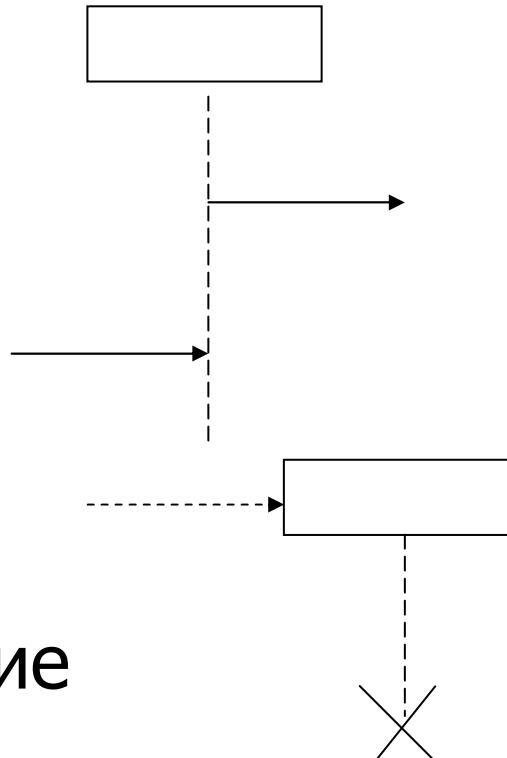
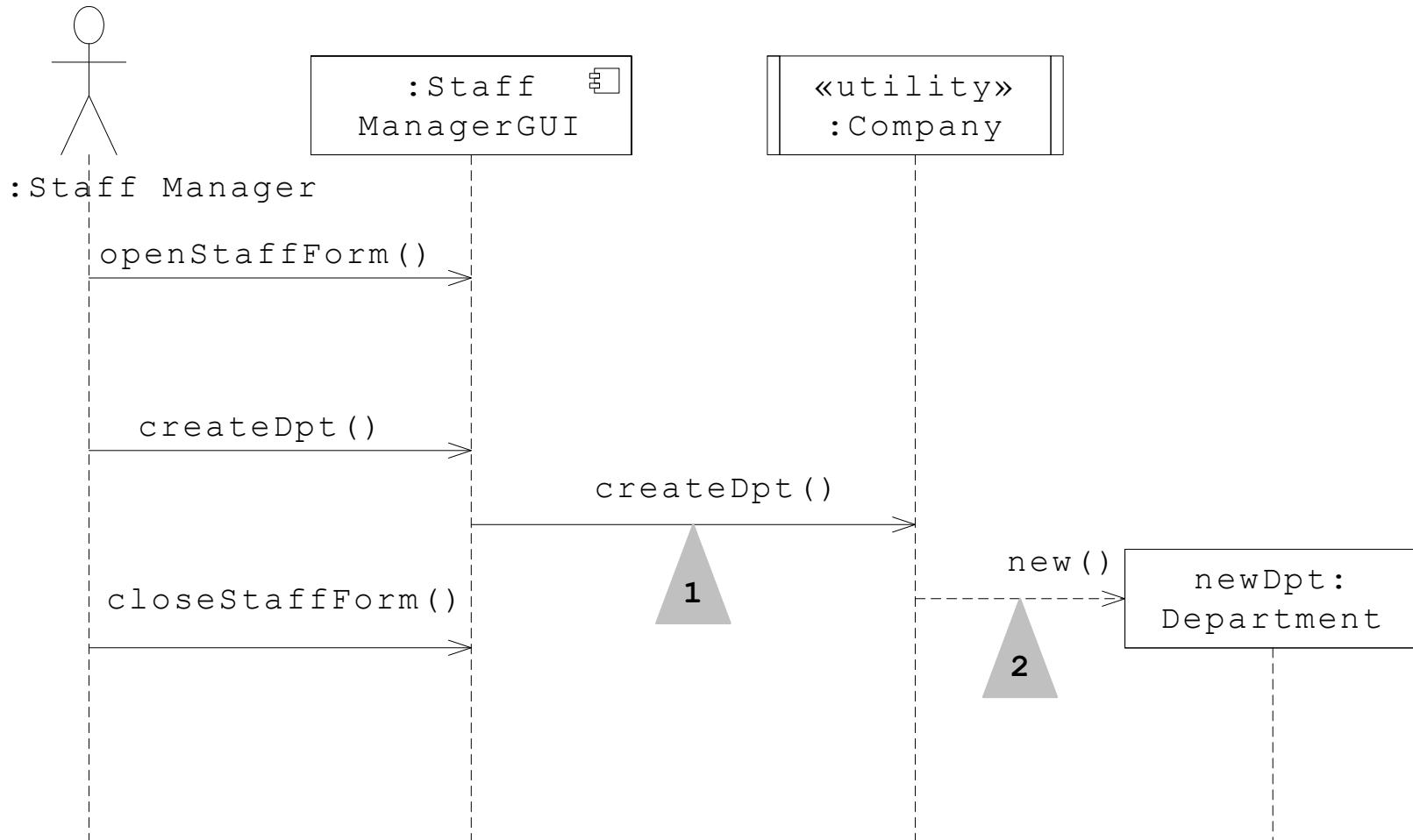


Диаграмма последовательности

исок

sd Создание подразделения



1) вызов метода

2) вызов конструктора

Основные классы компонента StaffManagerGui

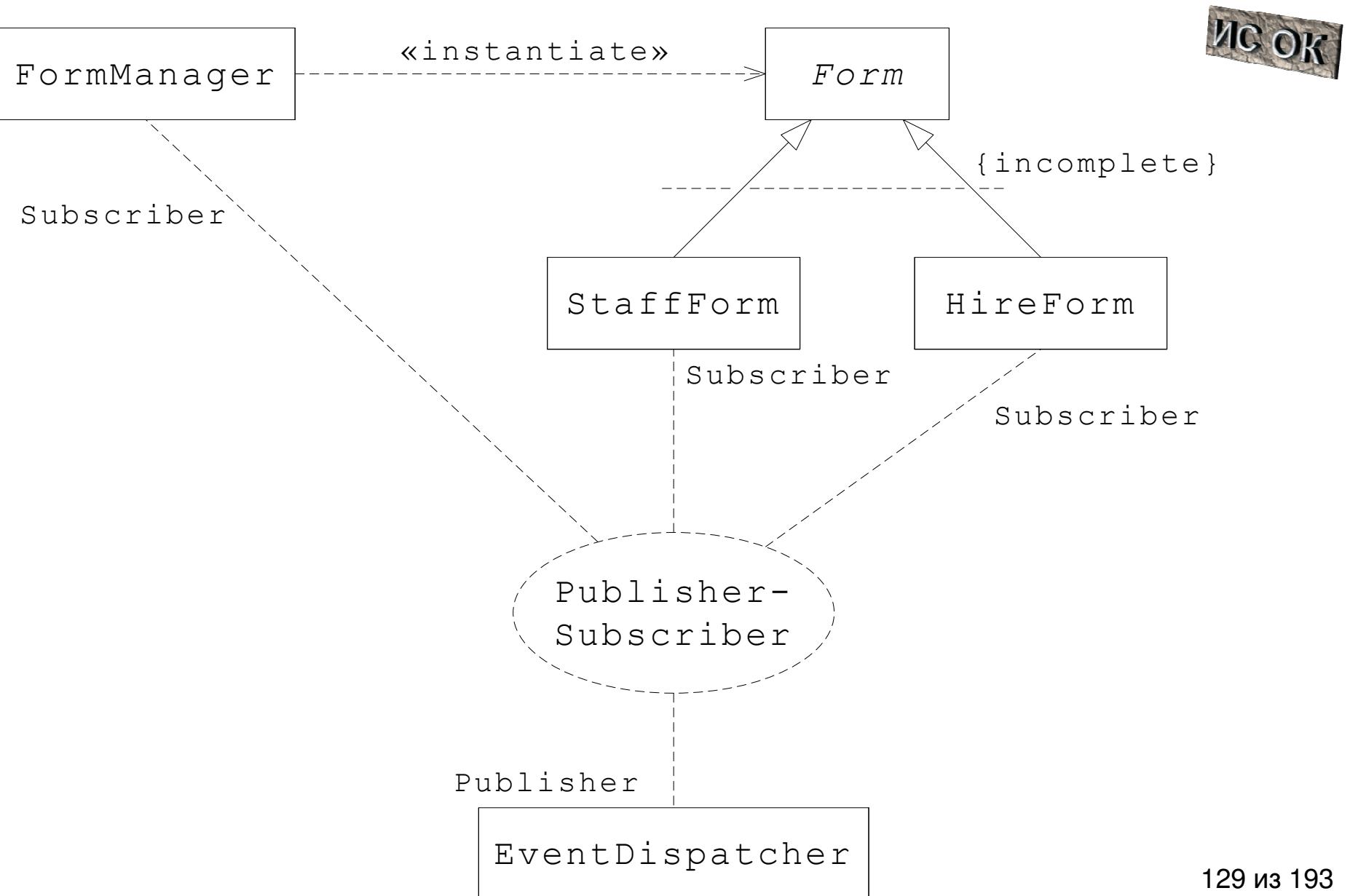
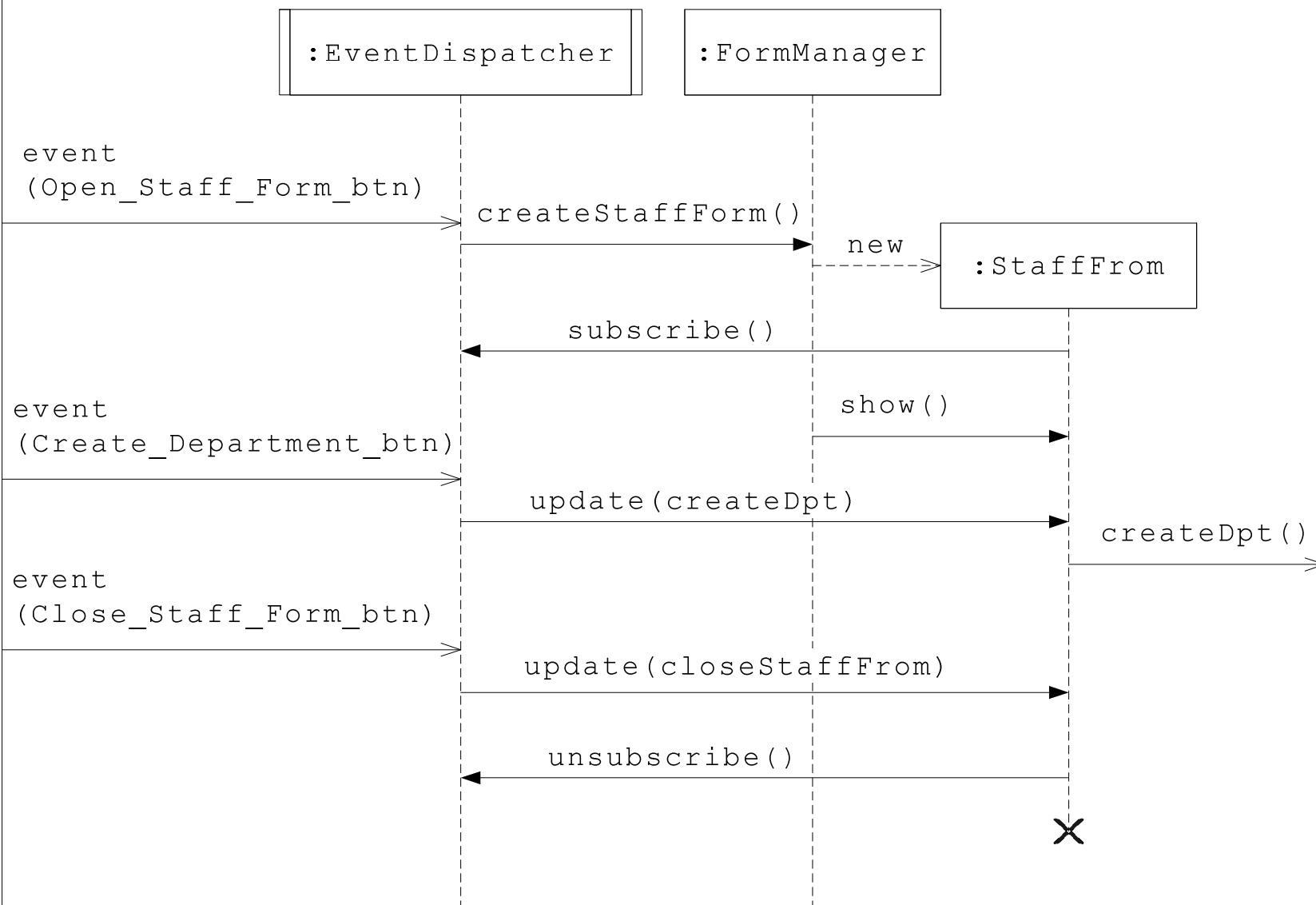


Диаграмма последовательности для StaffManagerGui

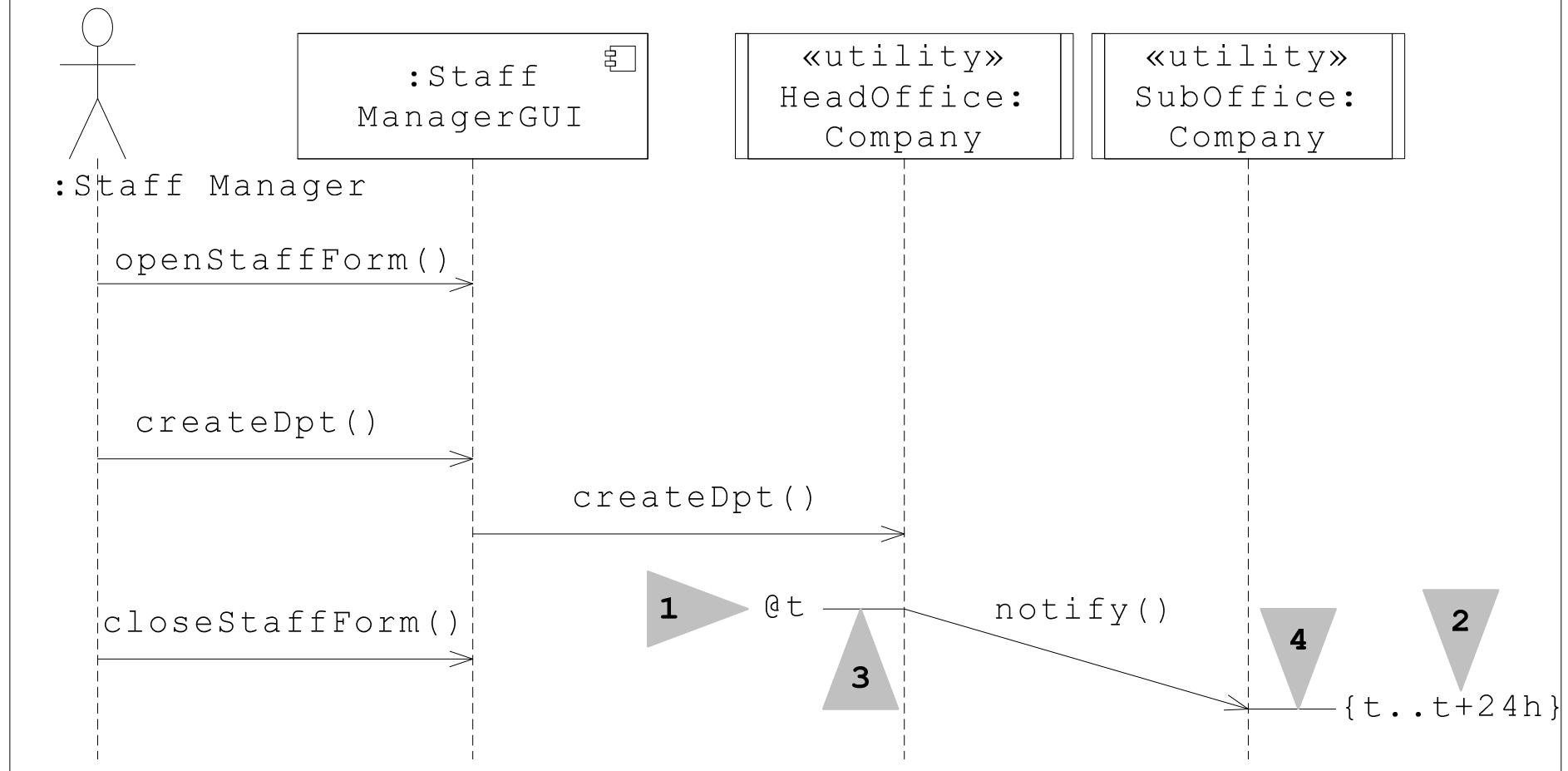
sd Создание подразделения (компонент StaffManagerGUI)



Задержанная доставка сообщения

исок

sd Создание подразделения (с нотификацией)

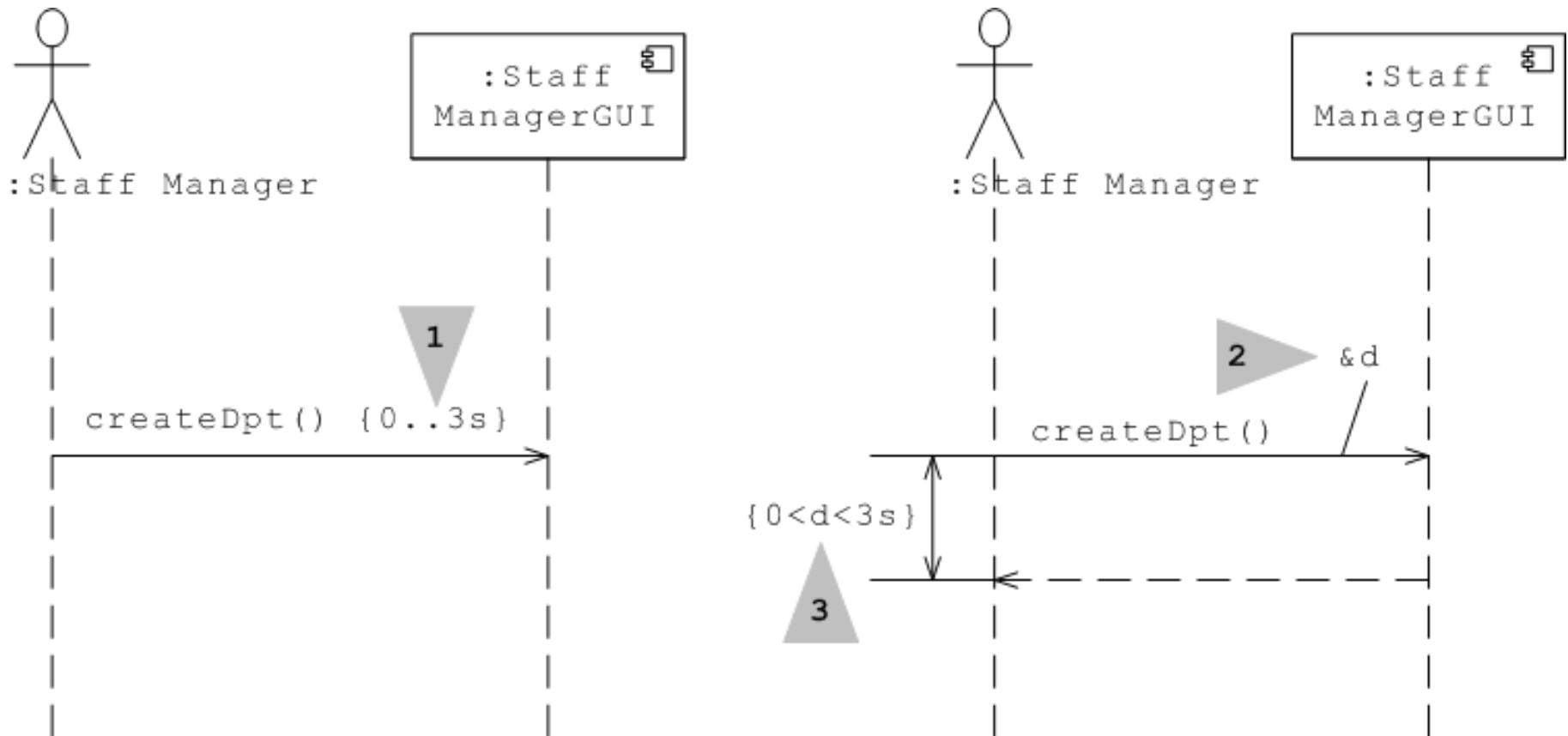


- 1) метка времени
2) временное ограничение

- 3, 4) привязка к линии жизни

Продолжительность выполнения метода

ис ок



1) ограничение

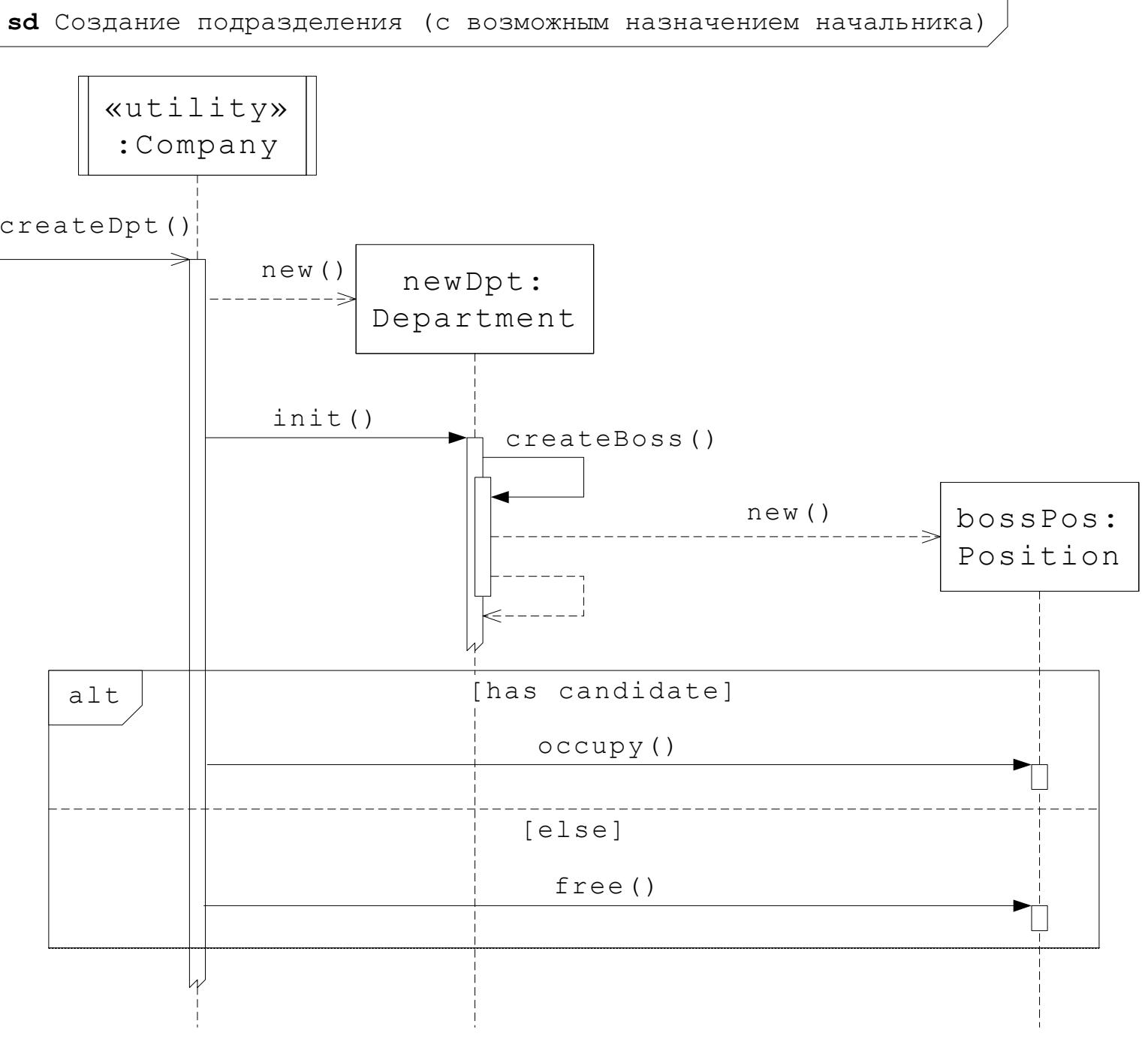
2) переменная длительности

3) расстояние между

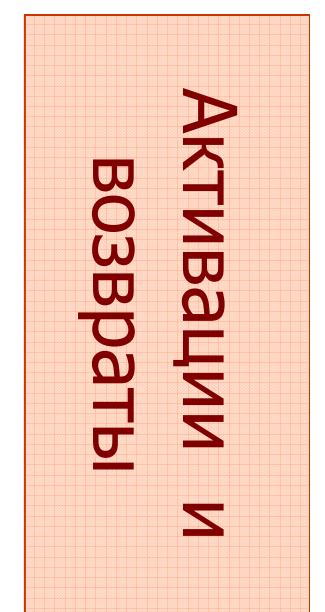
метками времени

Составной шаг Взаимодействия

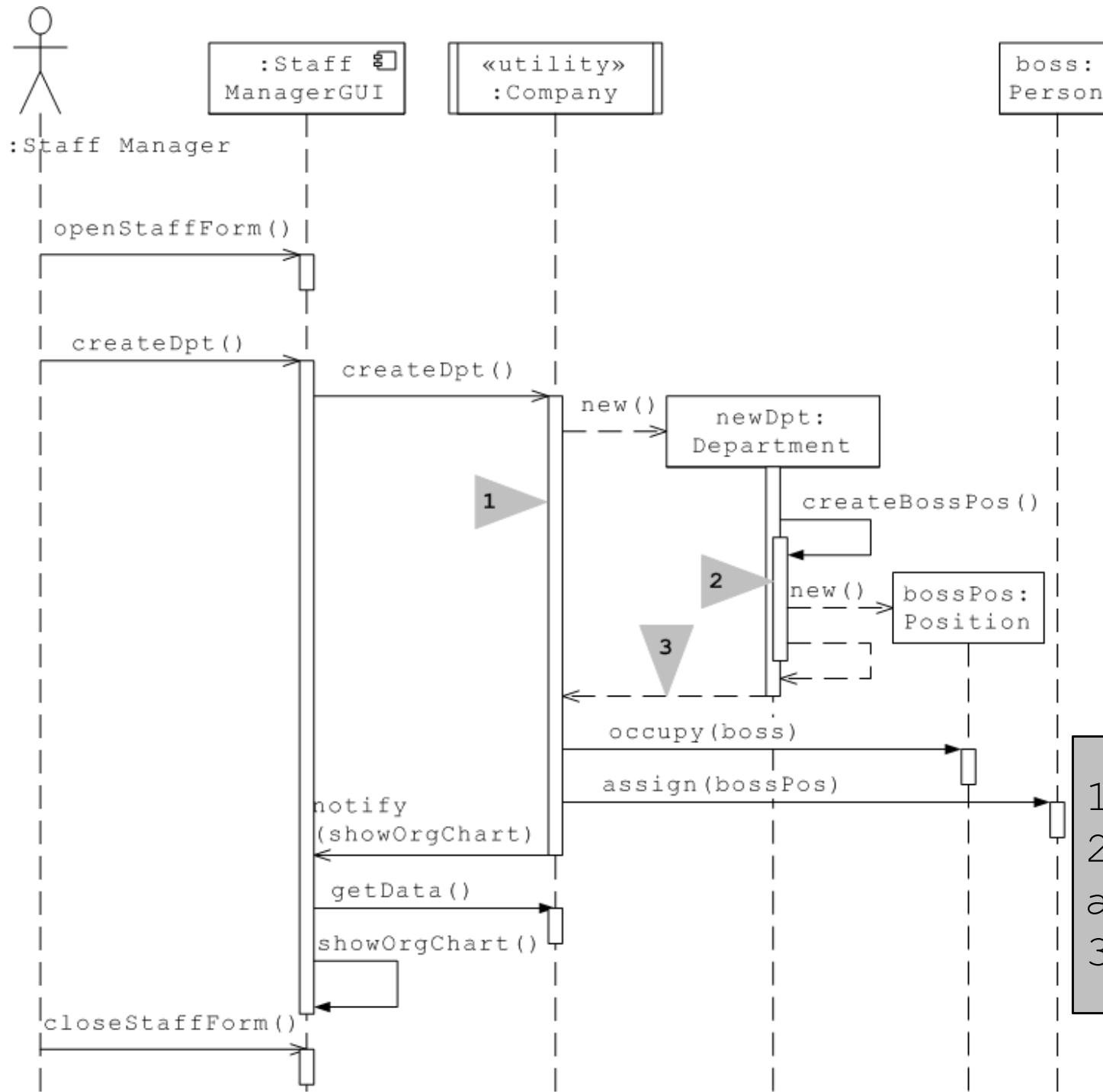
ИС ОК



Активации и возвраты



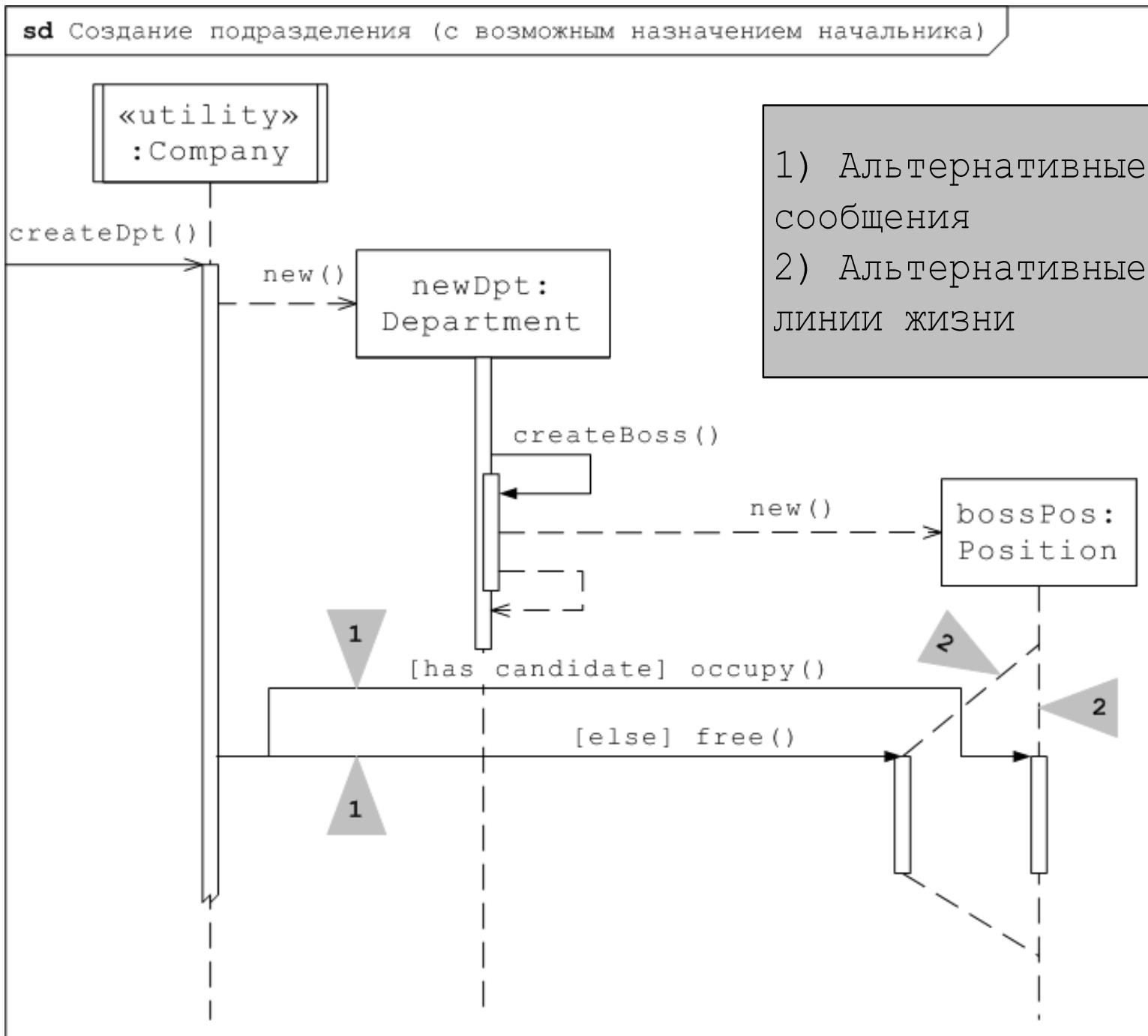
ИС ОК



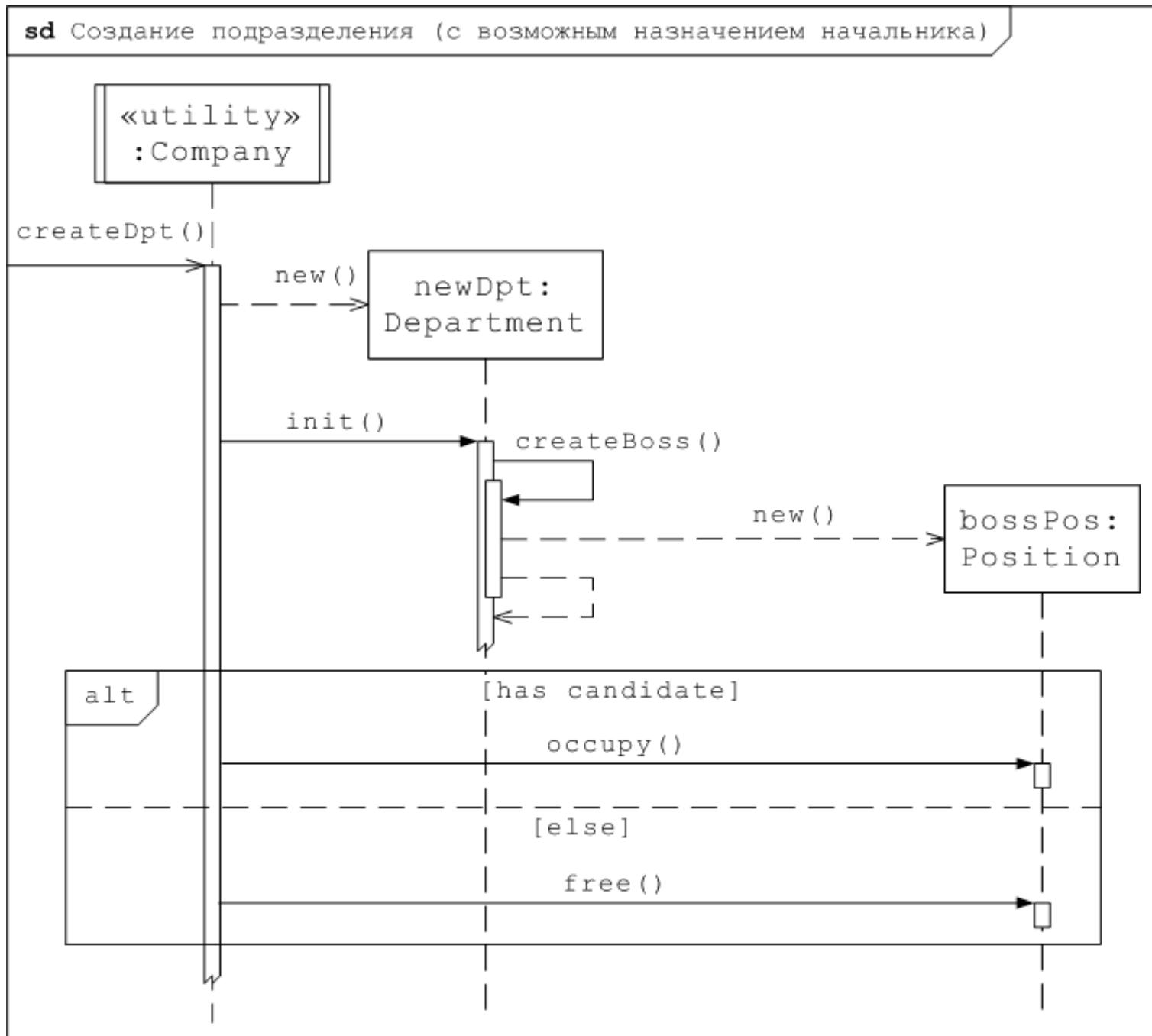
- 1) активация
- 2) вложенная активация
- 3) возврат

Ветвления и альтернативные линии жизни

ис OK



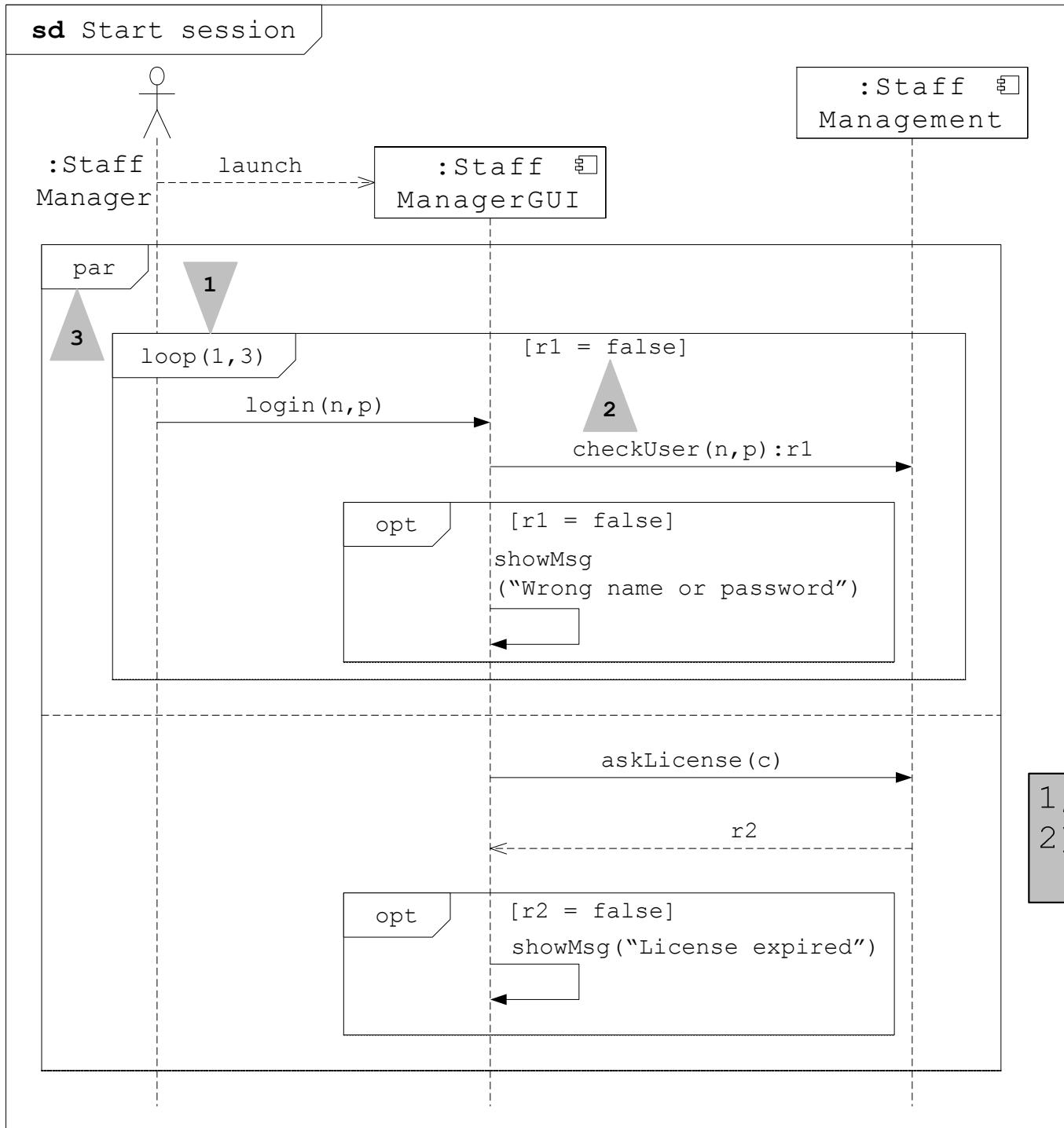
Составные шаги взаимодействия



ИС ОК

Составные вложенные шаги взаимодействия

ИС ОК

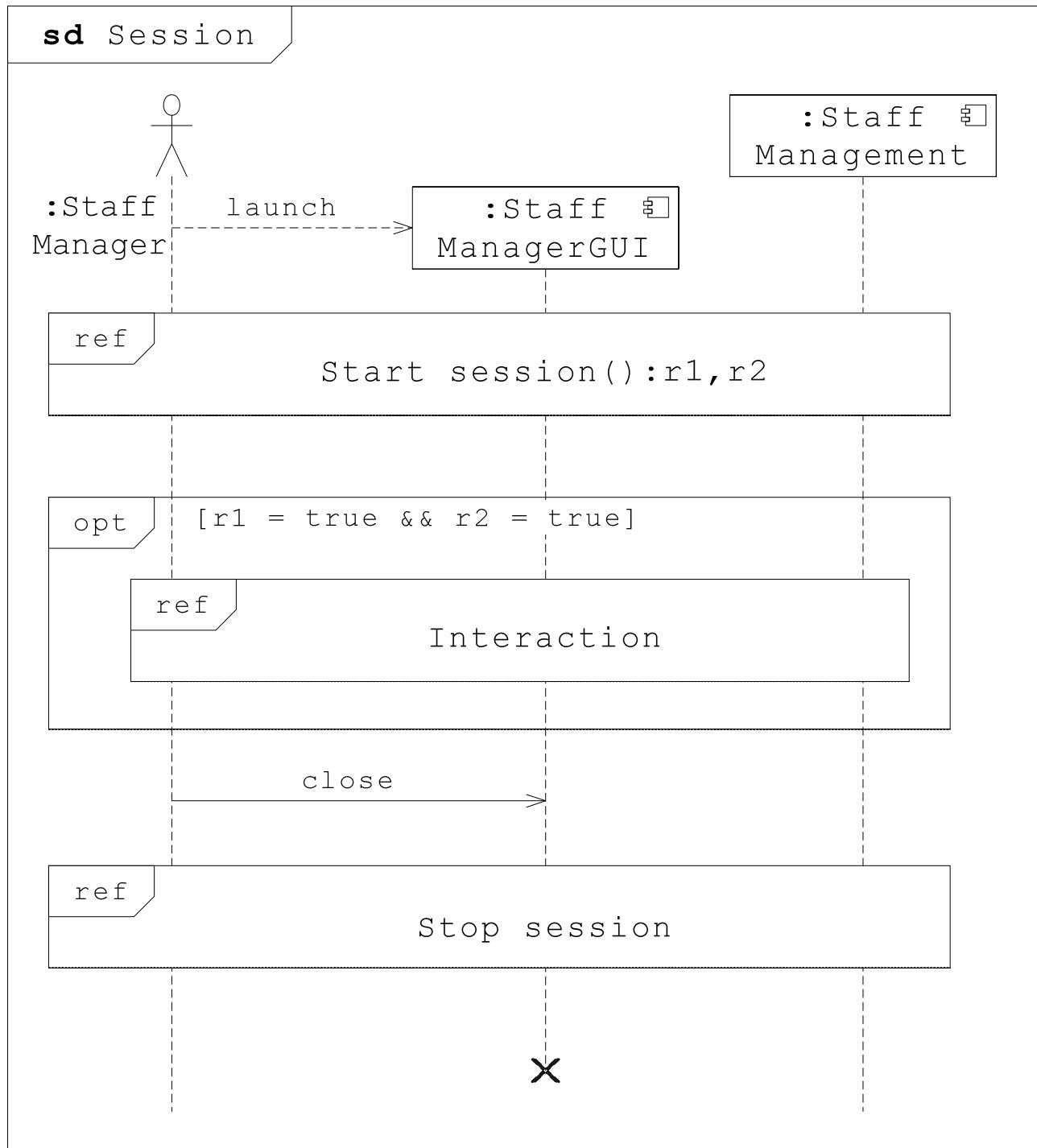


1, 3) составной шаг
2) сторожевое условие
составного шага

Использование взаимодействия

ИС ОК

Использование
взаимодействия
(interaction use)
ссылка на
взаимодействие,
определенное в
другом месте

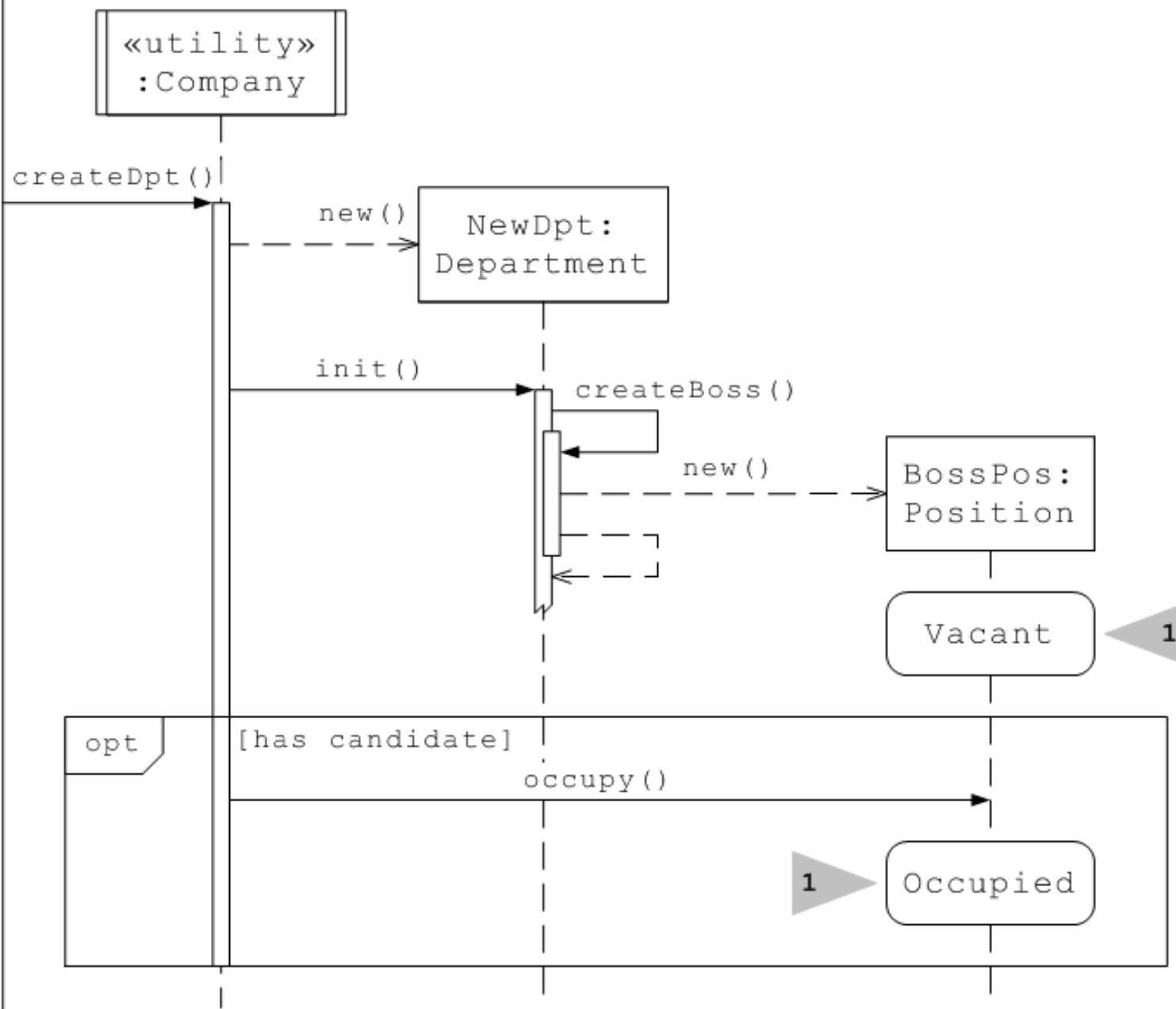


Отображение изменения состояний объекта на диаграмме последовательности

ис ок

139 из 193

sd Создание подразделения (с возможным назначением начальника)

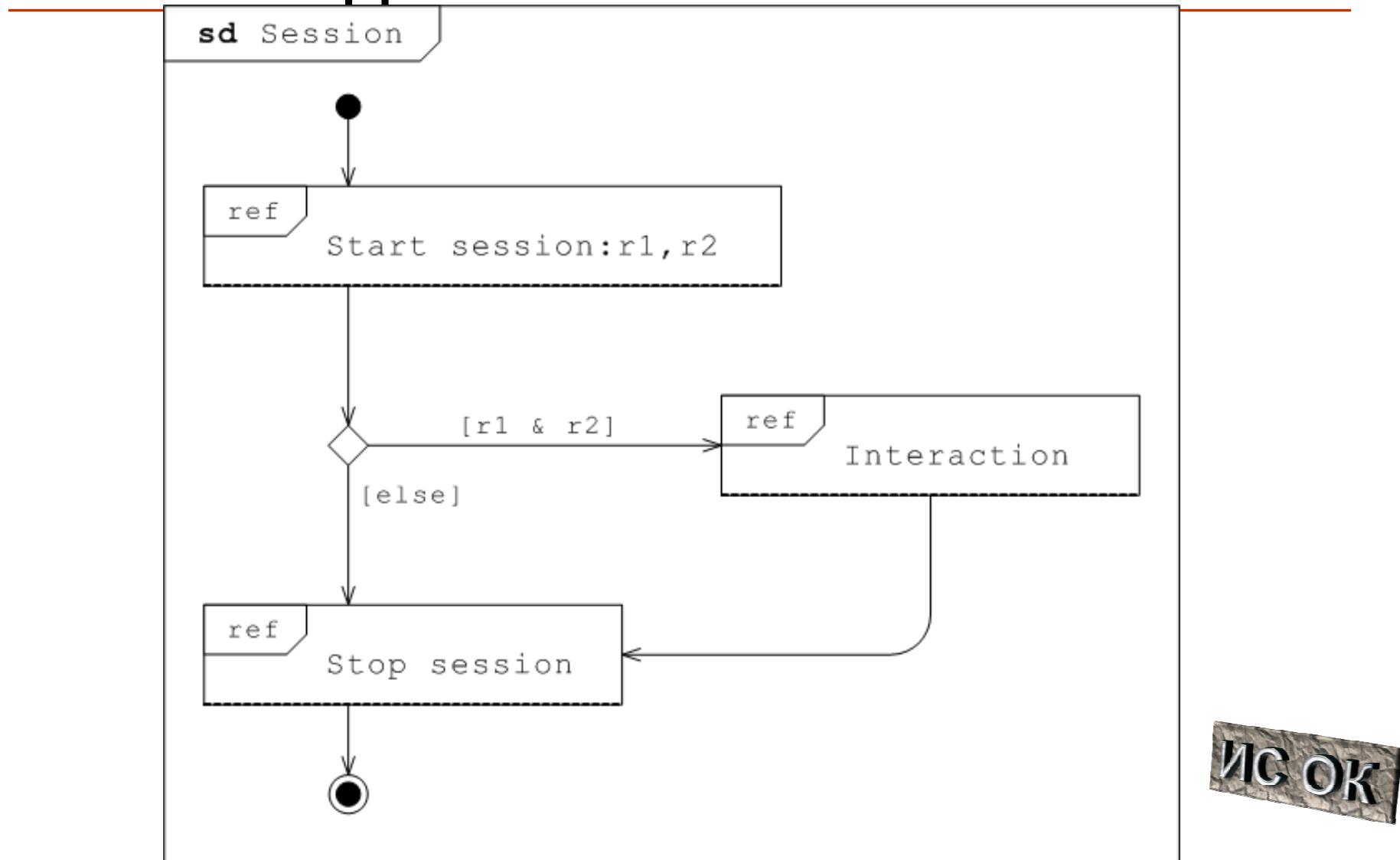


Обзорные диаграммы взаимодействия

= Interaction overview diagram

- Появились в UML 2
- Заимствованы из MSC
- Поток управления и узлы управления — как в диаграмме деятельности
- Вместо узлов действий — **ссылки** на диаграммы последовательности

Обзорная диаграмма взаимодействия для сессии ИС ОК



ИС ОК

Диаграммы коммуникации

- Сущность: объект
 - конкретный индивидуальный объект → конкретное взаимодействие
 - слот во фрейме взаимодействия → множество взаимодействий = **роль** (классификатора)
- Отношения: связи
 - связывает роли классификаторов → сама является слотом = **роль ассоциации**

Роли и кооперации

UML 1	UML 2	Обозначение	Определение
Роль классификатора	Роль (часть)	<code>name: Type</code>	Слот объекта, участвующего во взаимодействии
Роль ассоциации	Соединитель	_____	Слот связи, вдоль которой осуществляется взаимодействие
Контекст взаимодействия	Кооперация		Граф, показывающий роли и связи

Сообщения

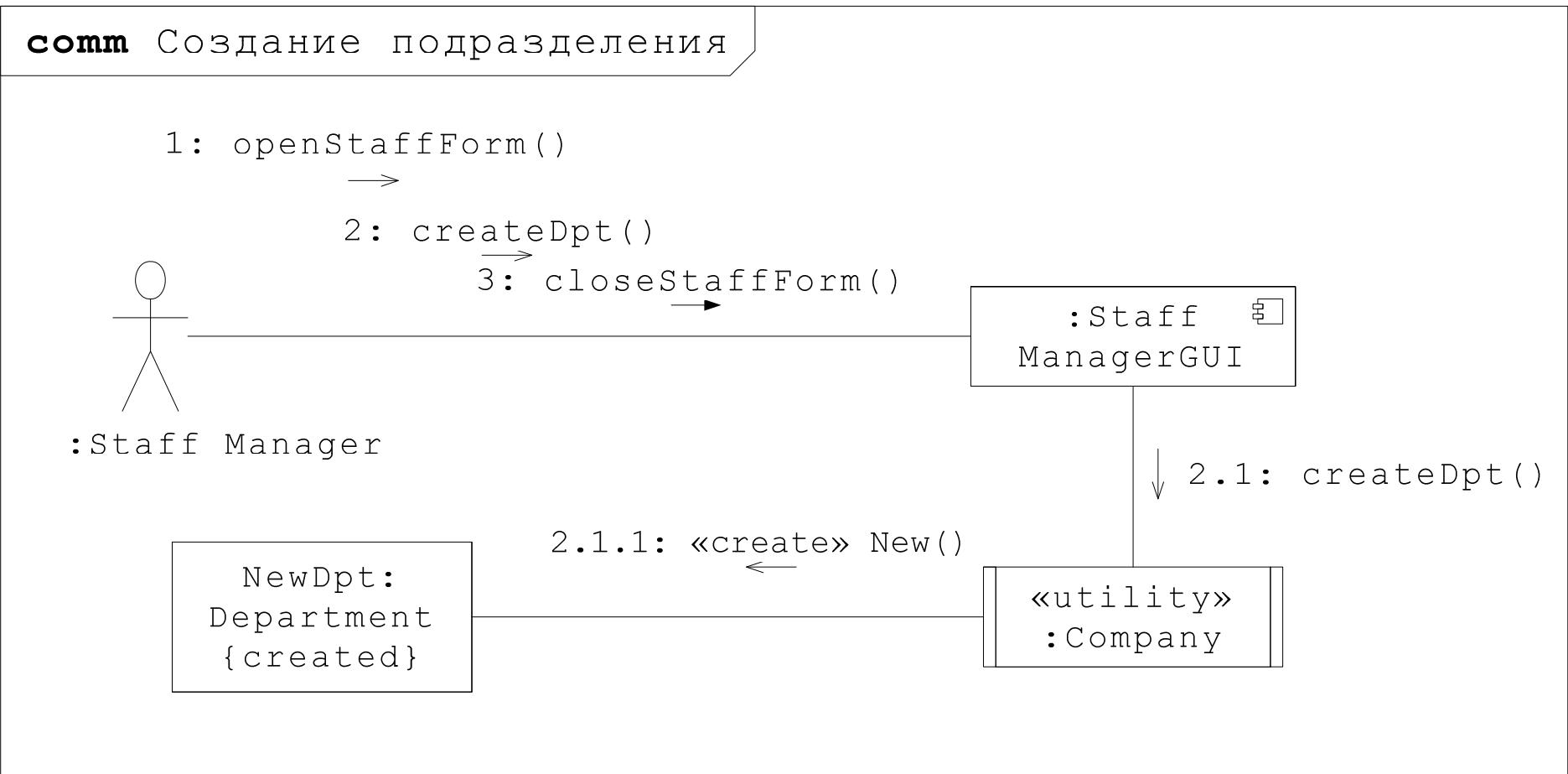
на диаграммах коммуникации

- Контекст взаимодействия: диаграмма коммуникации, в которой не указаны сообщения
 - Номер сообщения:
 - Если только простые или асинхронные сообщения — нумеруются подряд
- x
 $x.1, x.1.1, x.1.2, \dots$
 $x.2, \dots$
...

Время жизни объекта

Ключевое слово	Способ использования	Описание
Create	стереотип операции в сообщении	Операция создает объект
Destroy	стереотип операции в сообщении	Операция уничтожает объект
Destroyed	ограничение роли классификатора	Объект уничтожается в процессе описываемого взаимодействия
New	ограничение роли классификатора	Объект создается в процессе описываемого взаимодействия
Transient	ограничение роли классификатора	Объект создается и уничтожается в процессе описываемого взаимодействия Такой объект называется временным

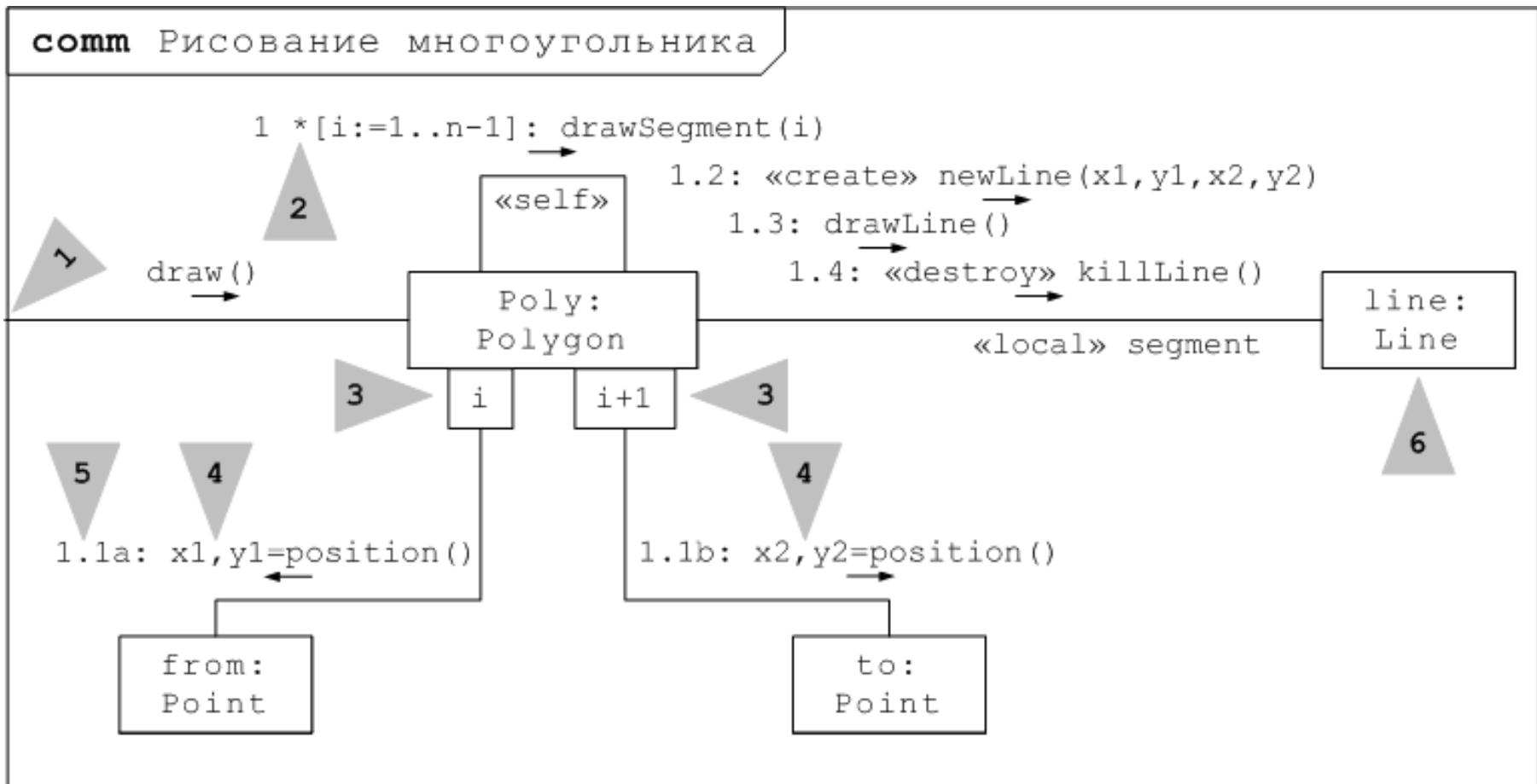
Диаграмма коммуникации



Стереотипы полюса роли ассоциации

Стереотип	Описание
«association»	Связаны фактической связью, реализующей ассоциацию
«global»	Глобальная область определения
«local»	Локальная область определения
«parameter»	Параметр операции объекта на противоположном полюсе
«self»	Роль ассоциации является фиктивной связью, введенной для моделирования вызова операции данного объекта

Рисование многоугольника



1. Шлюз и исходное сообщение
 2. Повторитель в сообщении
сообщения
 3. Значение атрибута квалификатора
 4. Возвращаемые значения
 5. Параллельные номера
 6. Локальный объект операции
- © 2008-2010
Ай Ти Ментор
- Д. Иванов, Ф. Новиков. Моделирование на UML. Вторая ступень
Моделирование поведения
- 148 из 193

5. Диаграммы синхронизации

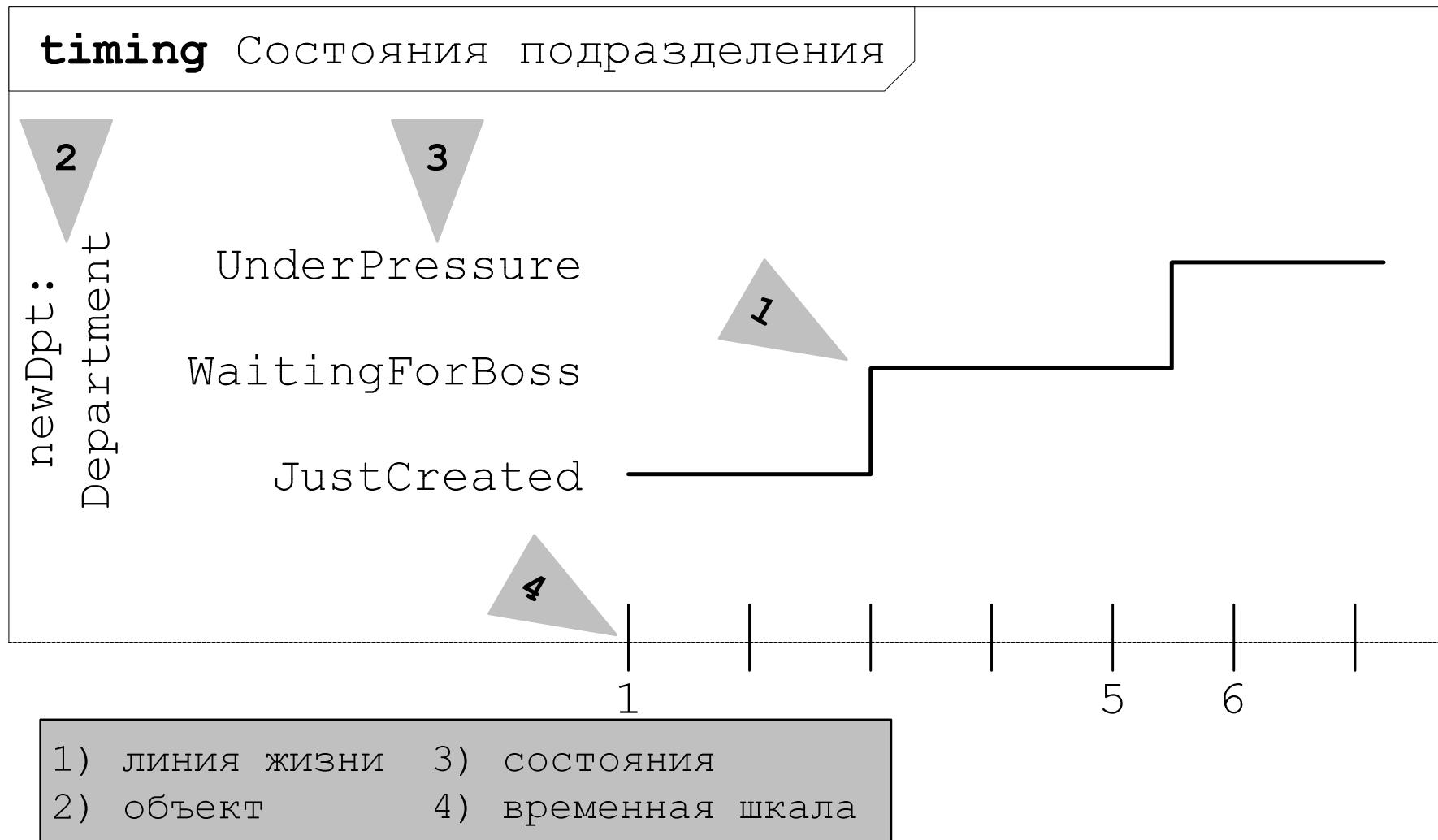
Диаграммы синхронизации (timing diagram)

— описание поведения системы объектов с указанием изменения их состояний во времени

- Состояния могут быть:
 - дискретны
 - непрерывны

Диаграмма синхронизации для одного объекта

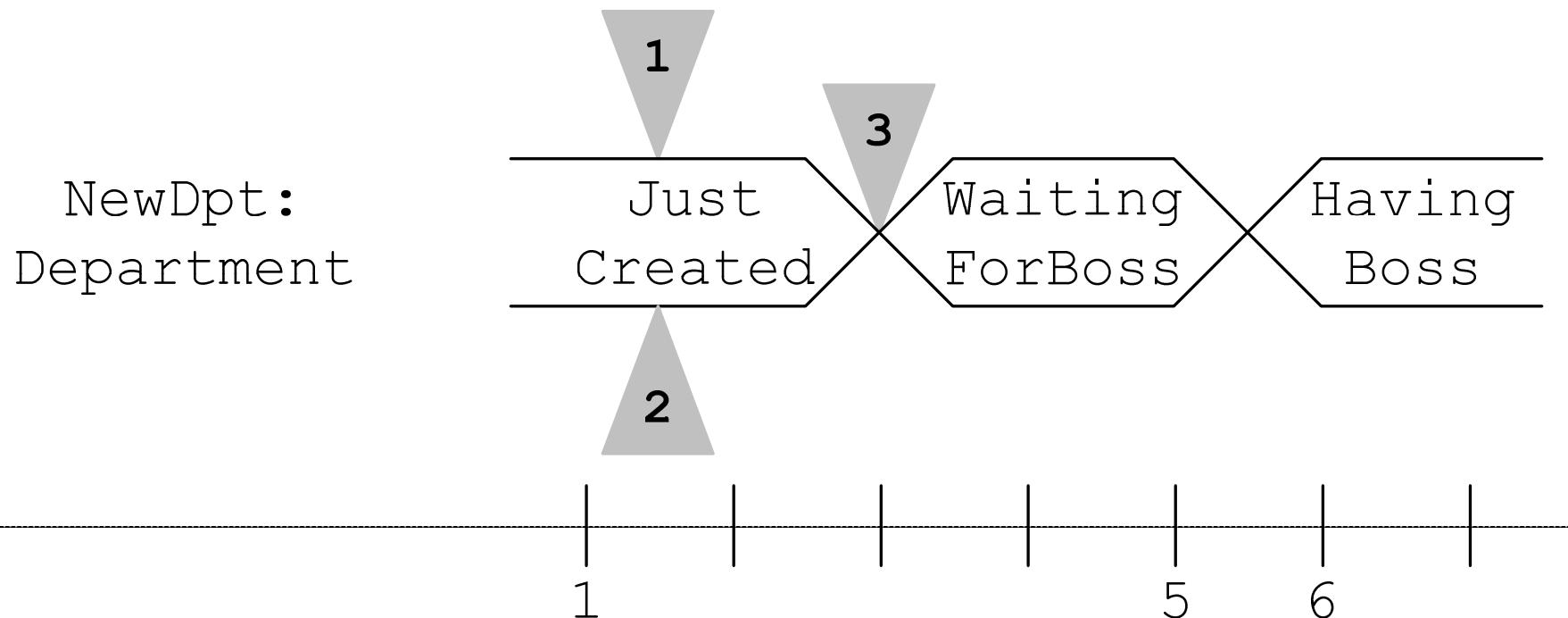
исок



Простое представление диаграммы синхронизации

ис ок

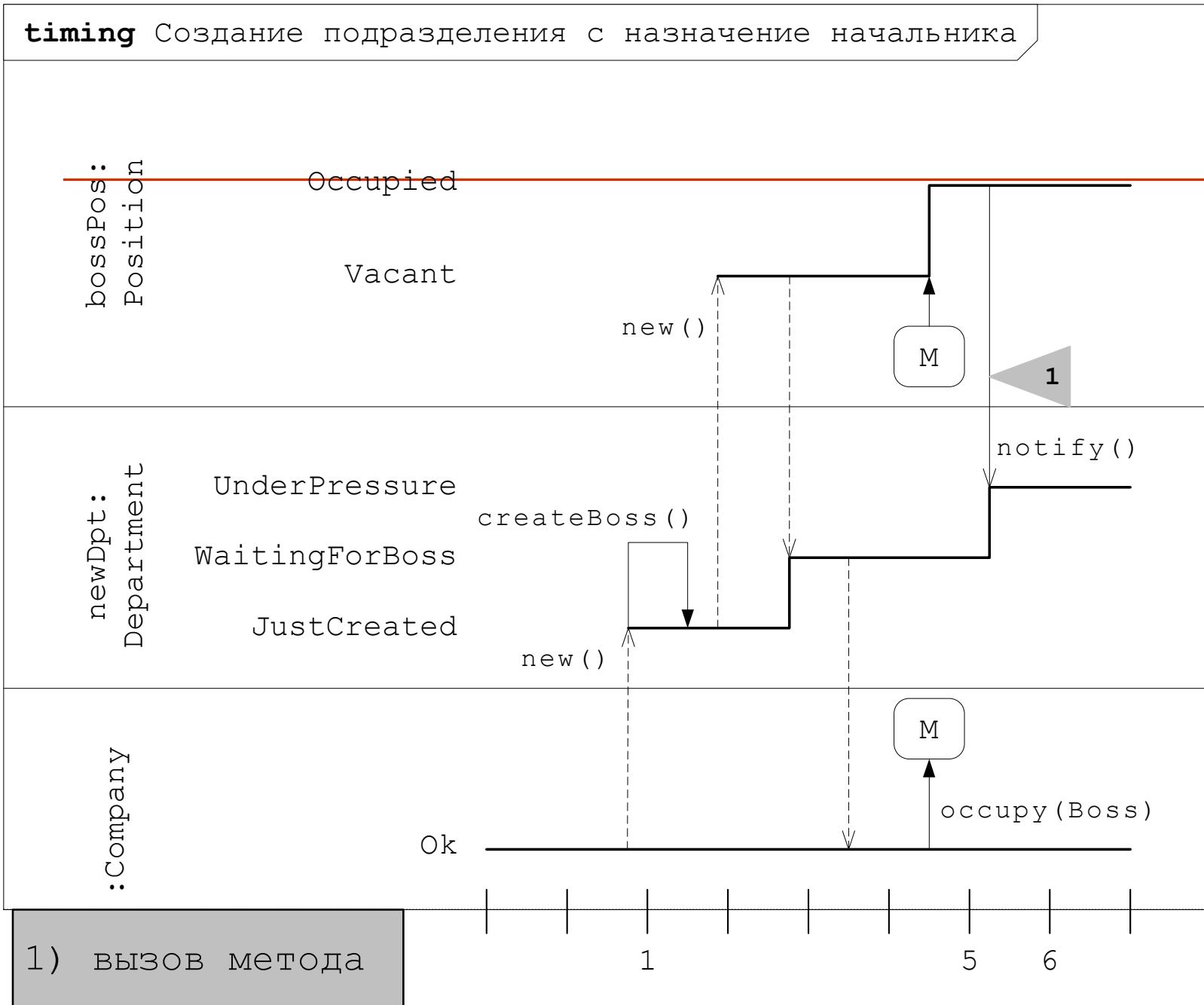
timing Состояния подразделения



1) состояние 2) точка смены состояний

Диаграмма синхронизации для нескольких объектов

исок



1) вызов метода

5. Моделирование параллелизма

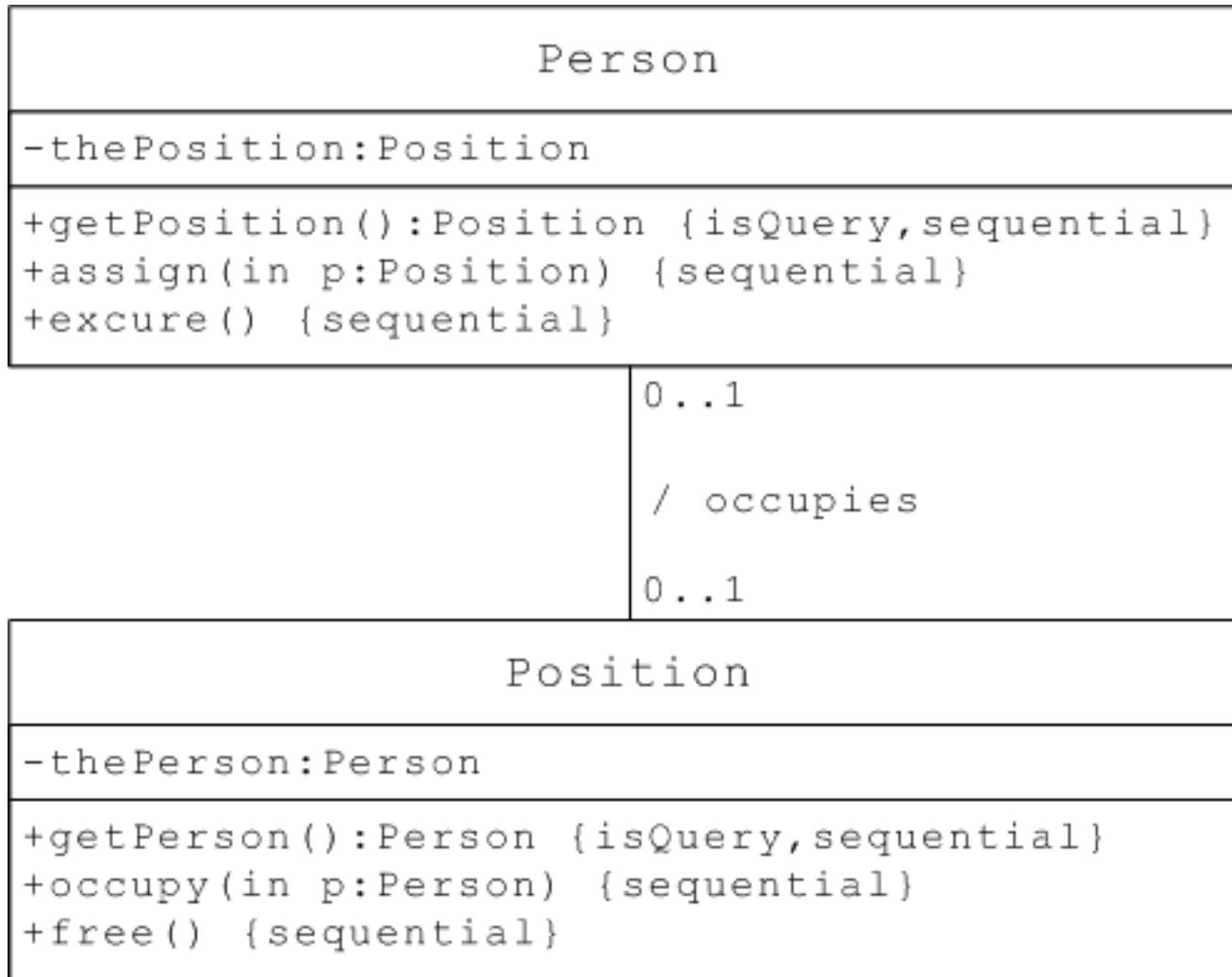
- Параллельность — “одновременное” выполнение нескольких действий
- Параллельные процессы не упорядочены во времени \Leftrightarrow последовательные: линейно упорядочены во времени
- UML: параллельные потоки управления и способы взаимодействия между ними

Взаимодействие последовательных процессов

- Взаимодействие в UML: **сообщения**
 - синхронные (вызовы операций)
 - асинхронные (посылка сигналов)
- Типичный вариант: взаимодействие машин состояний разных классов с помощью действий, выполняемых на переходах

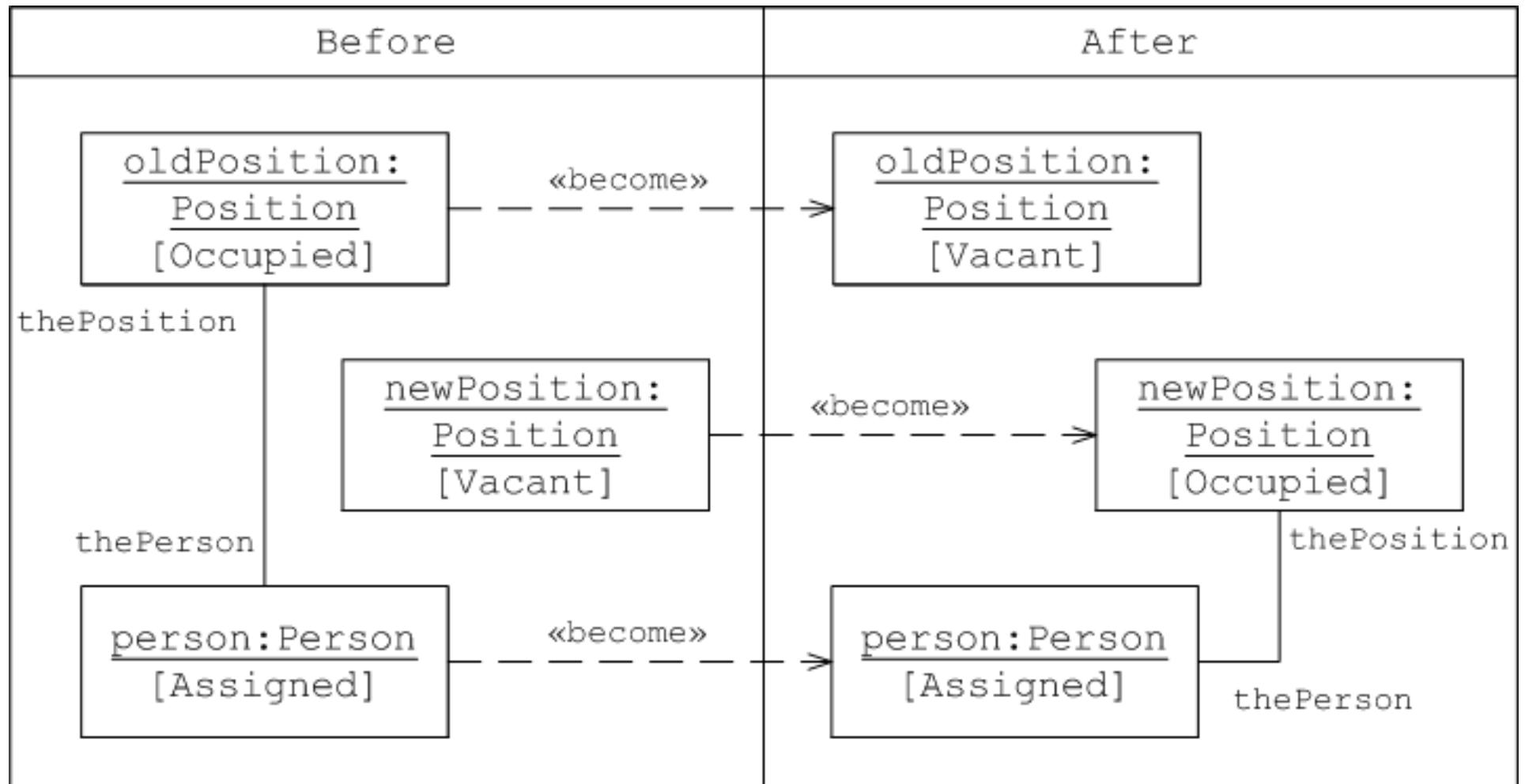
Пример: перевод сотрудника

исок



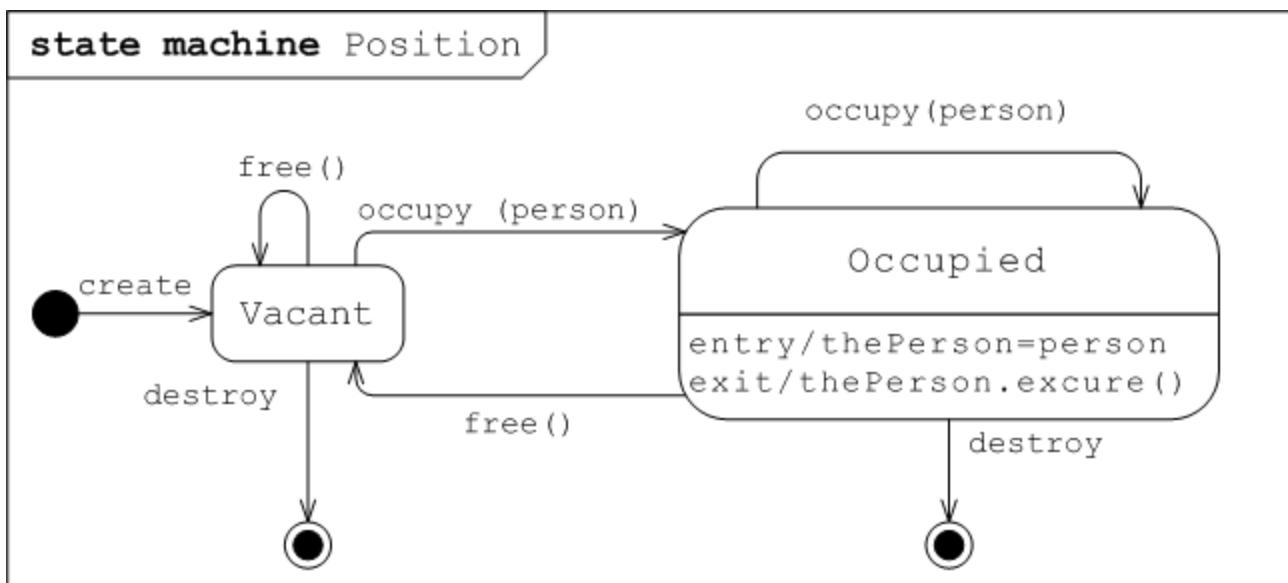
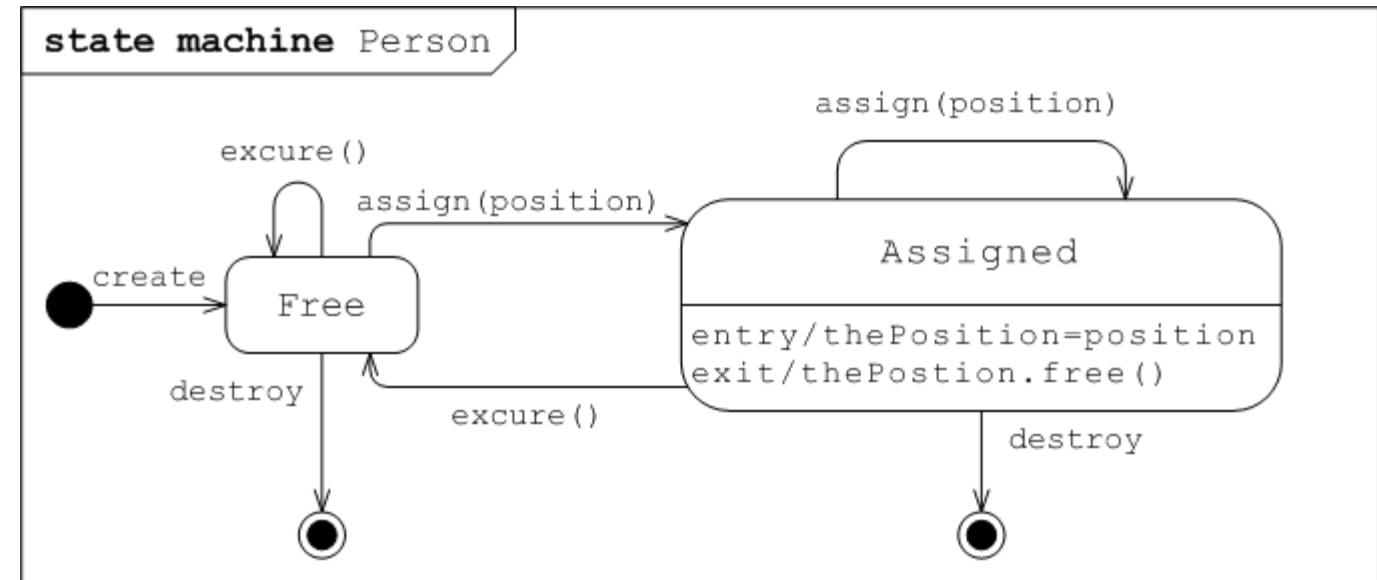
Требуется
первести с
одной
должности
на другую с
сохранением
ссылочной
целостности

Исходное и целевое состояния



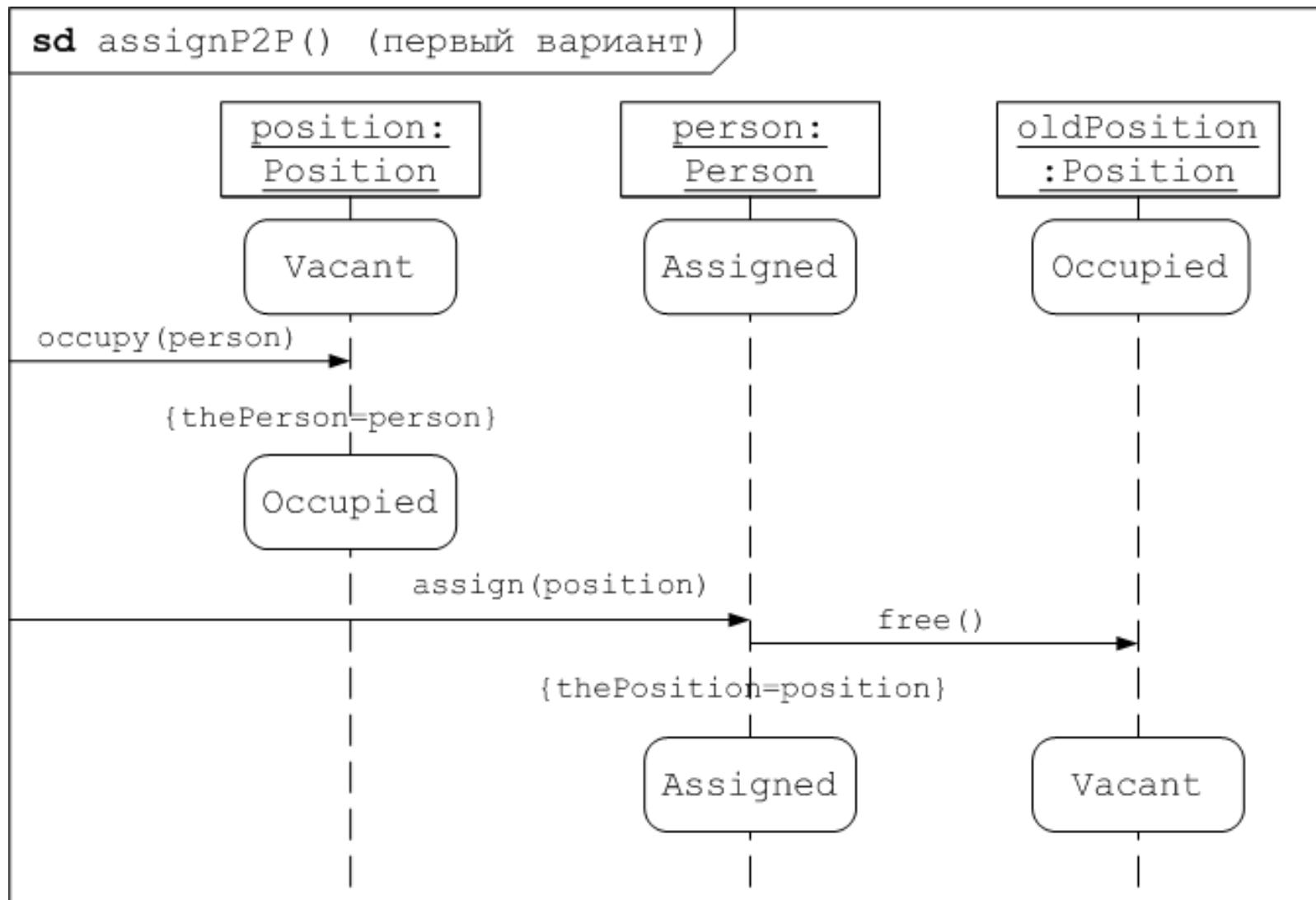
Изоморфные машины состояний

ис ок

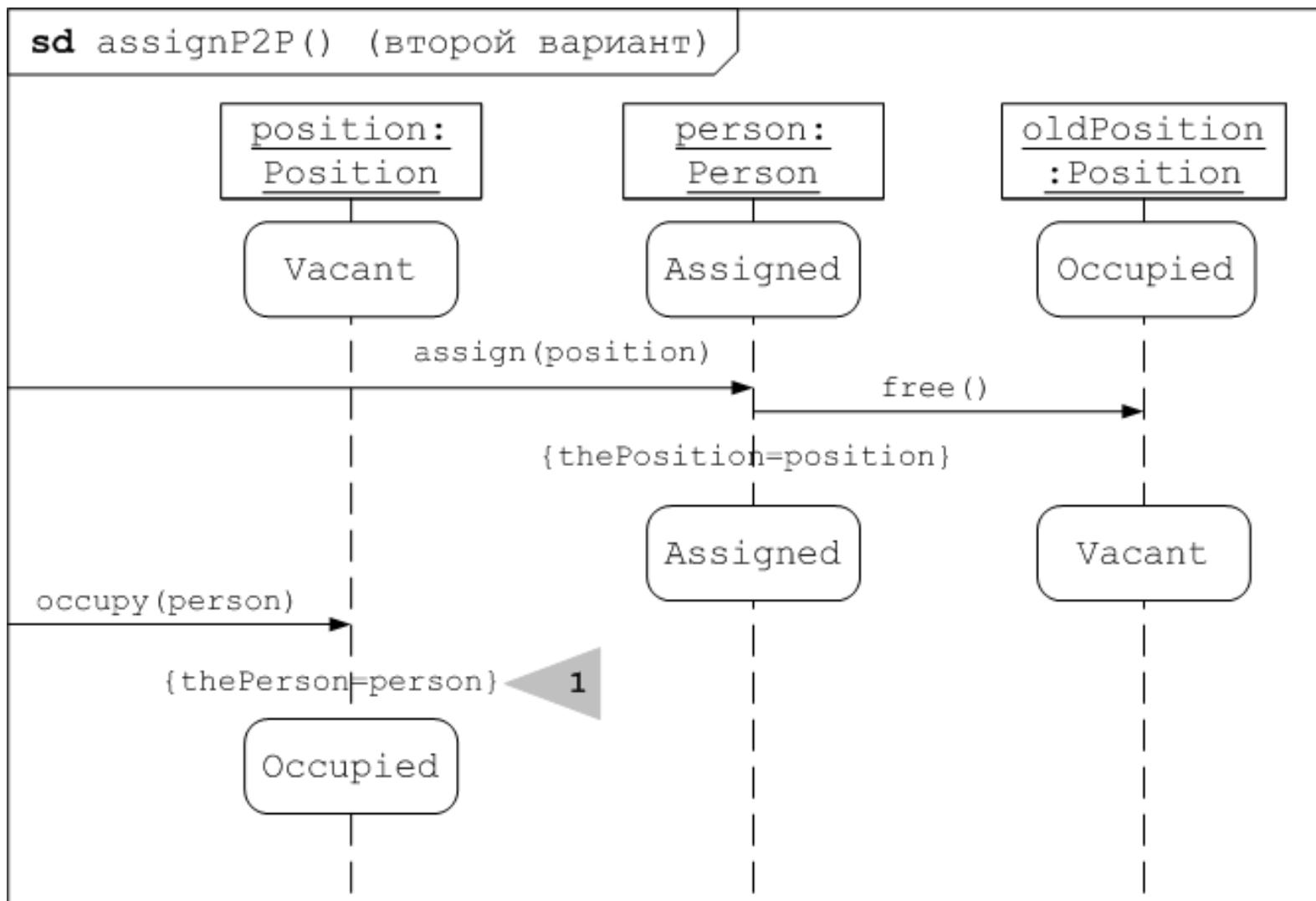


пень

Первый сценарий выполнения

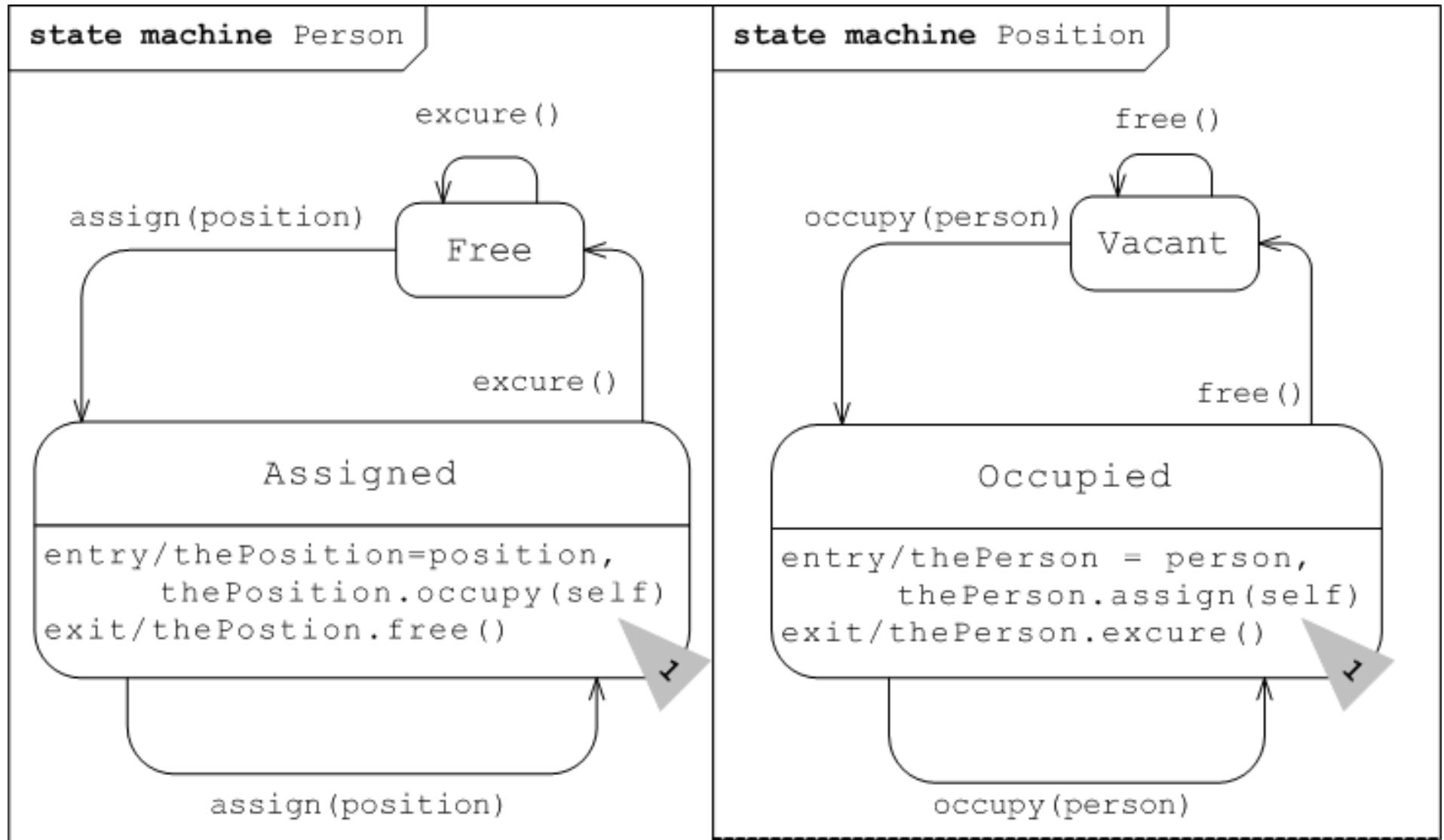


Второй сценарий выполнения

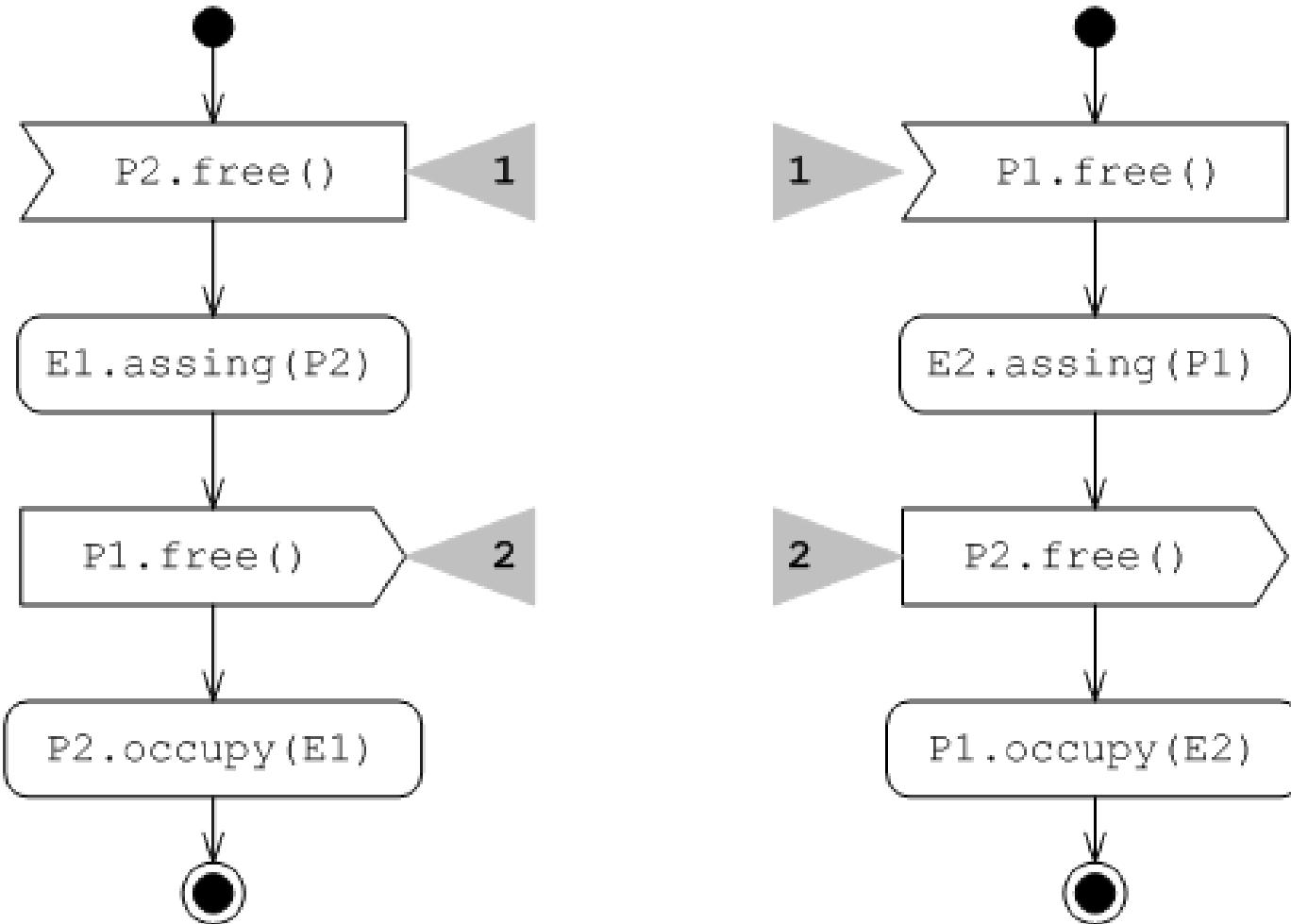


Бесконечная взаимная рекурсия

ИС ОК



Взаимная блокировка



Параллельная обработка данных

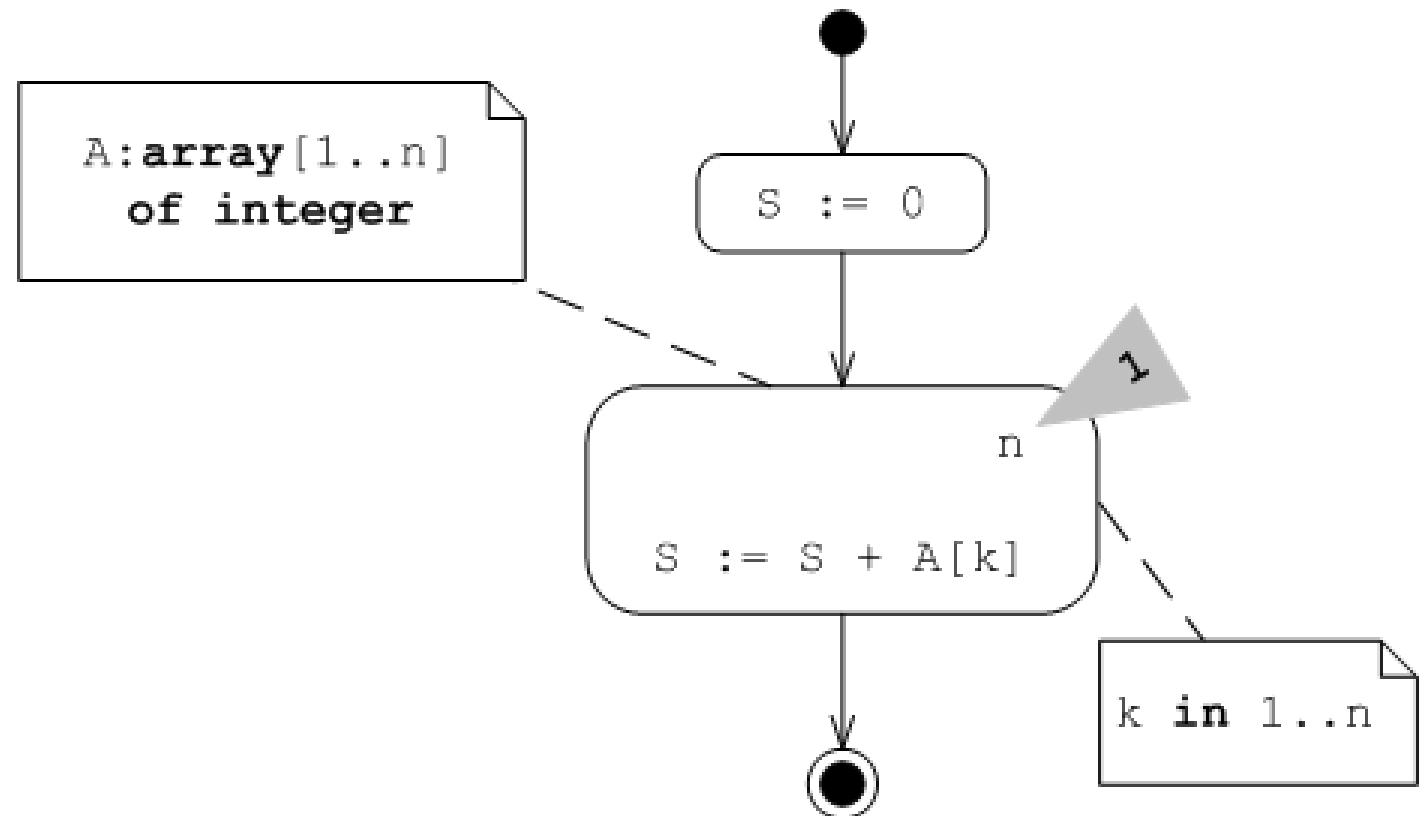
- Параллельная обработка данных:
 - Начисление зарплаты, рассылка писем, создание заказа из нескольких позиций и т.д.
- Решение задачи: найти сумму S заданных чисел $a_i, i = 1, \dots, n$
 - `var A : array [1..n] of real; S : real;`
`i : integer;`
 - `S := 0; for i:=1 to n do S := S + A[i];`

ИЛИ

- `var A : bag [n] of real; S : real;`
- `S := 0; for each a∈A do S := S + a;`

Динамическая параллельность

Динамическая параллельность
(dynamic concurrency) на диаграммах
деятельности в [UML 1](#)



Область разложения в UML 2

Область разложения (expansion region) — структурированный узел деятельности

- получает коллекцию на вход
- разлагает коллекцию на составляющие элементы
- производит указанные действия индивидуально с каждым элементом коллекции
- помещает элементы в выходную коллекцию, если нужно

Действия над элементами коллекции

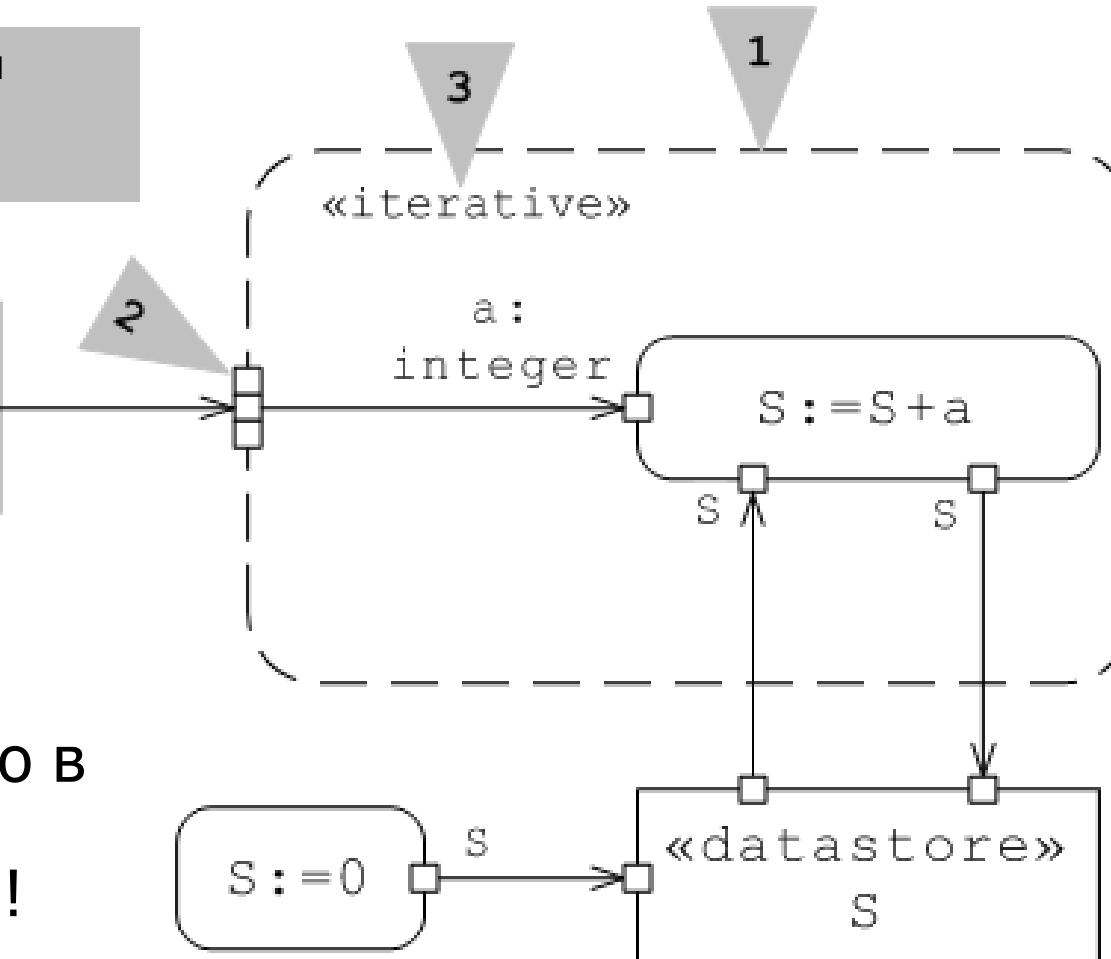
Ключевое слово	Описание	Примечания
«iterative»	в порядке, который задает коллекция	<ul style="list-style-type: none">• N маркеров данных и N экземпляров тела области• Выполнение экземпляров строго по очереди
«parallel»	параллельно, т.е. в произвольном порядке	<ul style="list-style-type: none">• N маркеров данных и N экземпляров тела области• Выполнение экземпляров в произвольном порядке
«stream»	конвейерным потоком	<ul style="list-style-type: none">• N маркеров данных и 1 экземпляр тела области• Маркеры данных поступают в тело по очереди, но следующий не дожидается, когда будет завершена обработка предыдущего

Последовательная область разложения

1. Рамка области разложения
2. Узел разложения
3. Ключевое слово

```
A:array[1..n]
  of integer
```

- Суммирование массива
- Суммировать можно в любом порядке, но переменная S одна!



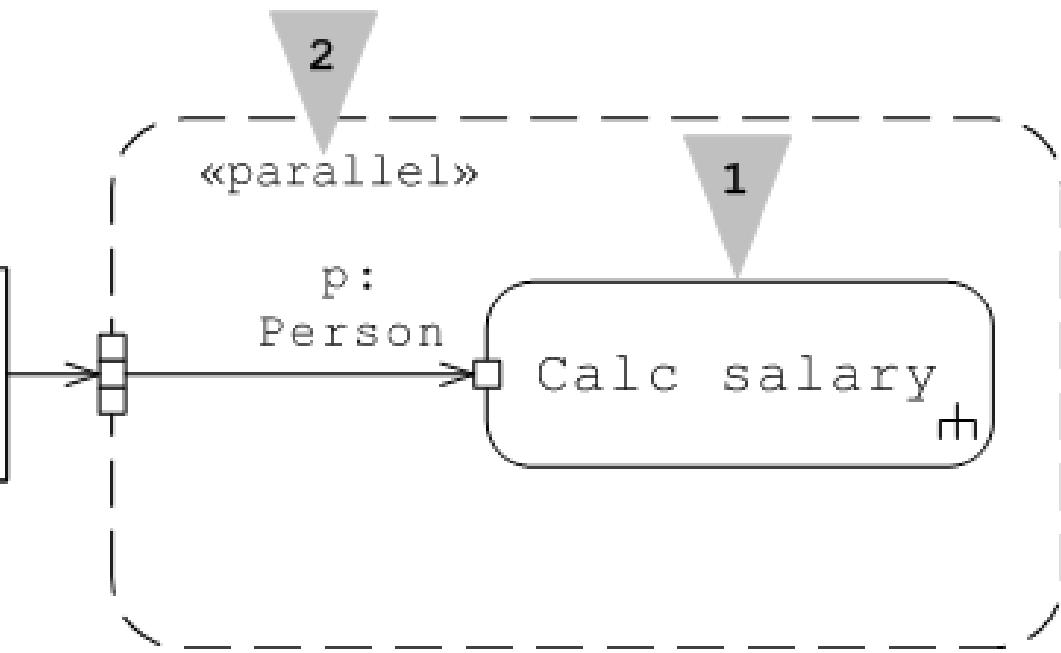
Параллельная область разложения

ИС ОК

- Начисление зарплаты
- Начислять зарплату можно в любом порядке и независимо для каждого работника!

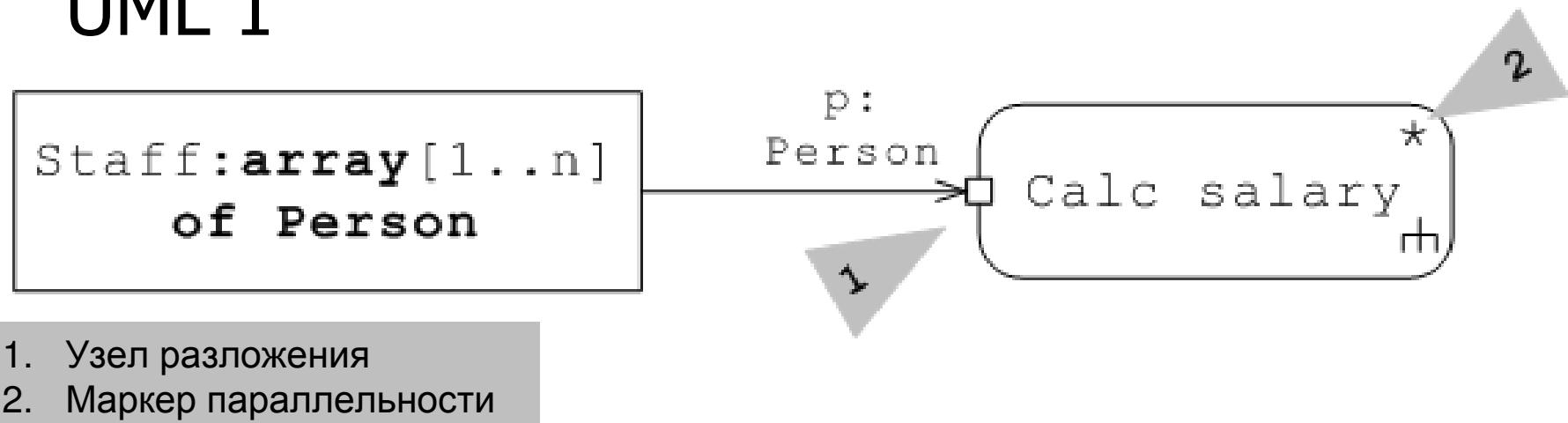
1. Действие по вызову деятельности
2. Ключевое слово

```
Staff:array[1..n]
      of Person
```

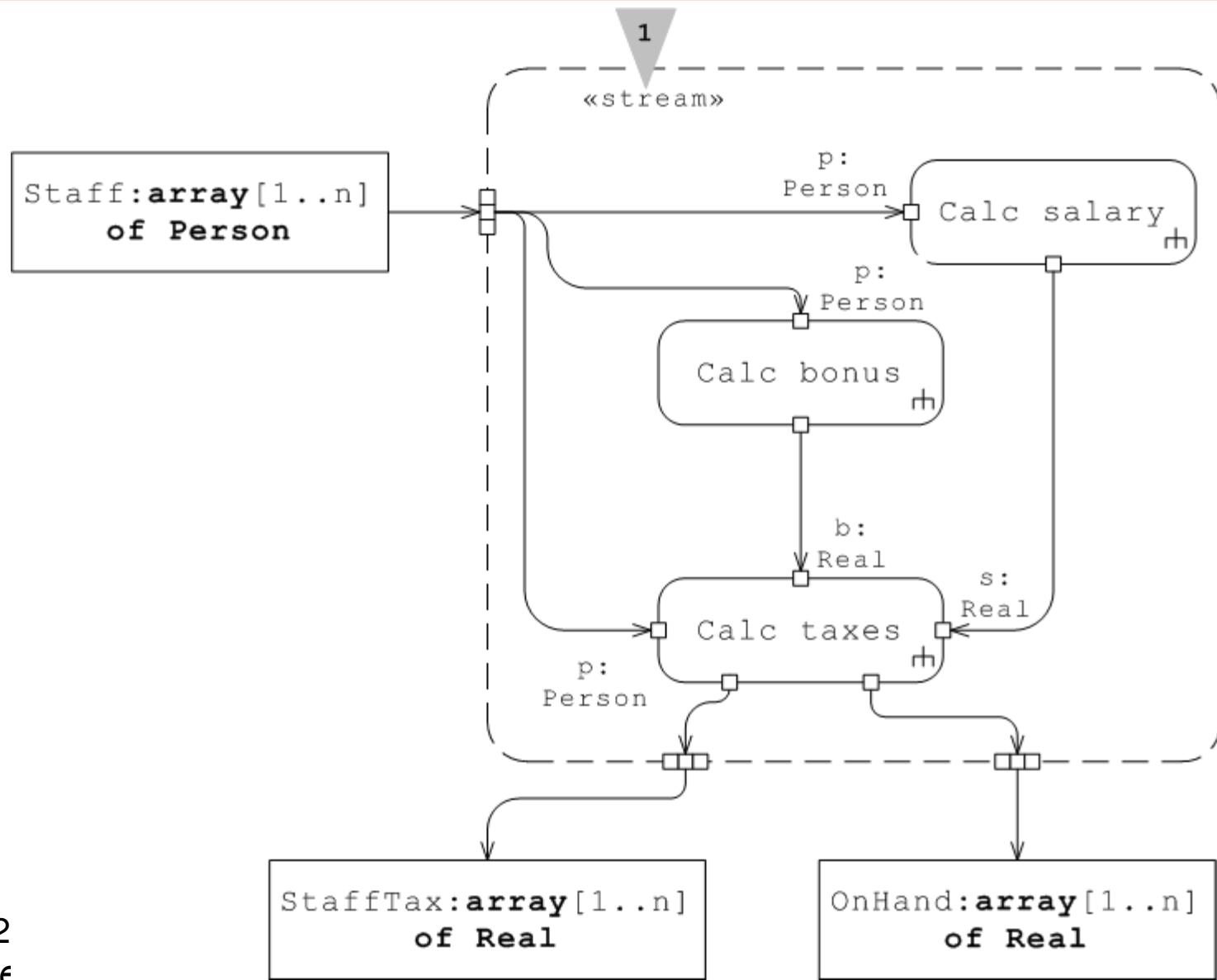


Сокращенная нотация

- Если внутри области разложения содержится всего одно действие
- Применяется только для параллельного разложения
- Напоминает динамическую параллельность UML 1



Конвейерная область разложения

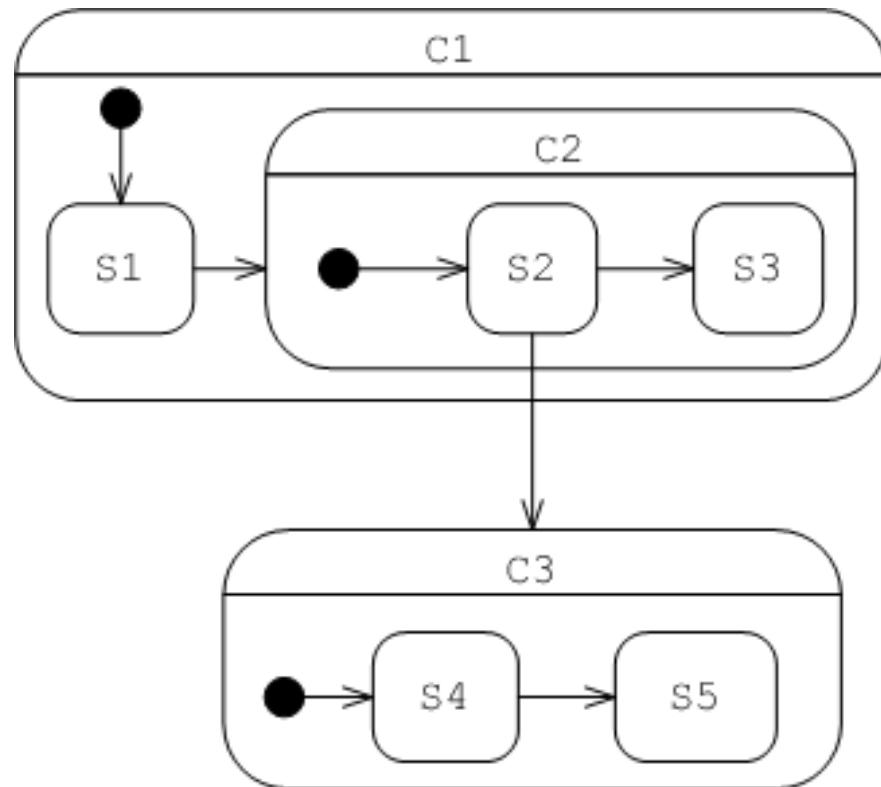


Параллельность на диаграммах состояний

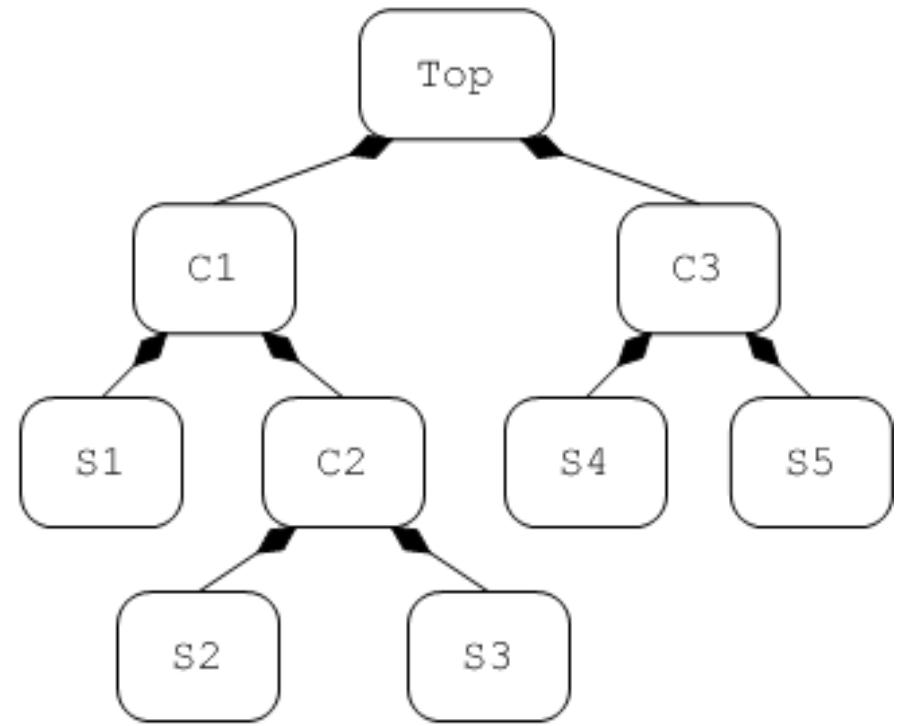
Ортогональные
(параллельные) составные состояния и
составные переходы

- Активное состояние (active state) — в котором находится машина состояний в данный момент
 - простое состояние
 - все составные состояния, объемлющие данное простое состояние — **конфигурация активных состояний**

Конфигурация активных состояний



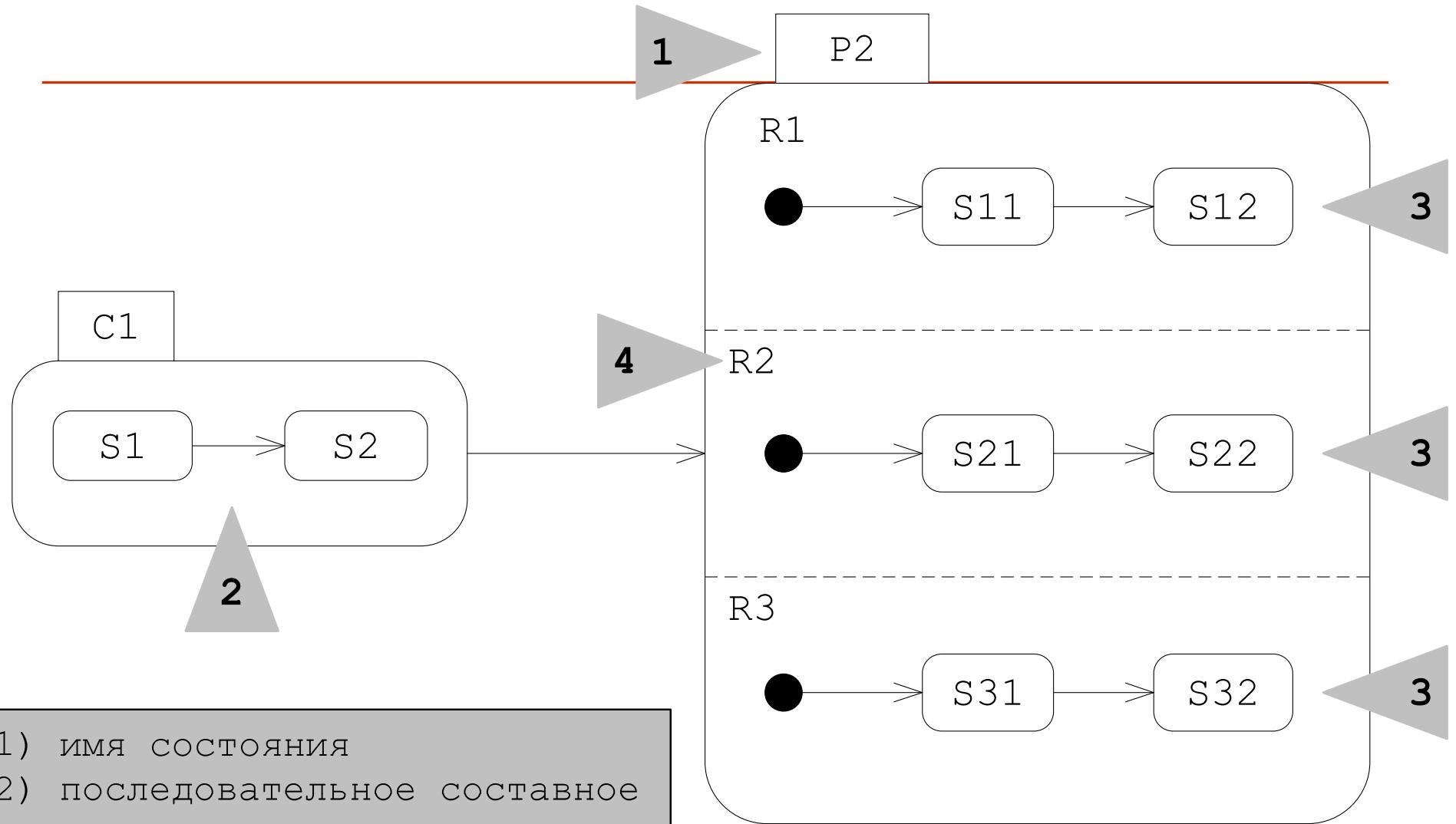
В нотации UML



В виде дерева

Конфигурация активных состояний — путь от корня к листу

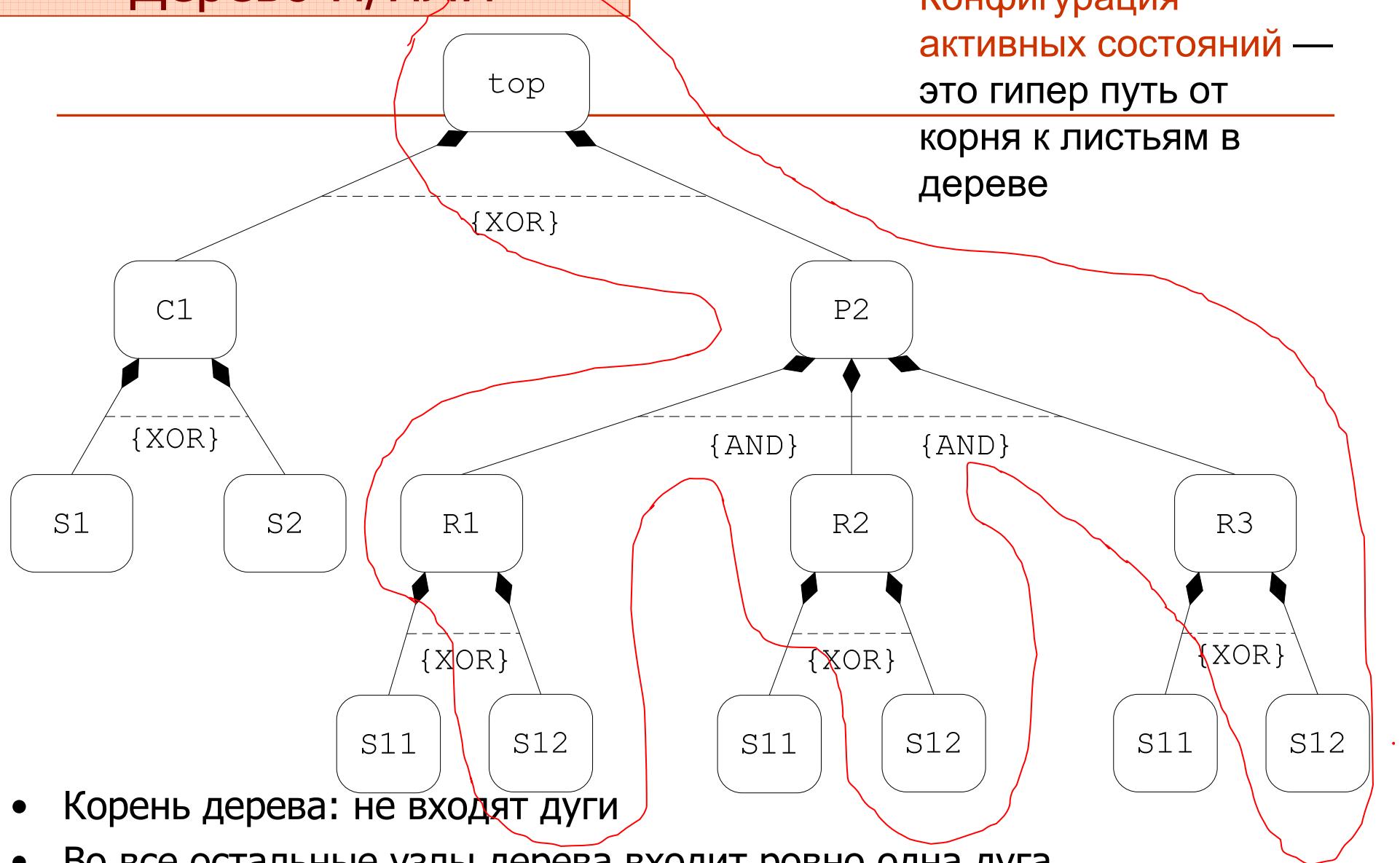
Ортогональное составное состояние



- 1) имя состояния
- 2) последовательное составное состояние
- 3) области ортогонального составного состояния
- 4) имя области

Дерево И/ИЛИ

Конфигурация
активных состояний —
это гипер путь от
корня к листьям в
дереве



- Корень дерева: не входят дуги
 - Во все остальные узлы дерева входит ровно одна дуга

Переходы, не пересекающие границу составного ортогонального состояния

Аналогично последовательным составным состояниям

- Переход в состояние — переход в начальные состояния всех областей данного состояния
- Переход по событию и/или со сторожевым условием из состояния — распространение данного перехода на все вложенные состояния всех областей
- Переход по завершении из состояния — синхронное завершение всех вложенных машин состояний

Составной переход

Составной переход (compound transition) — это переход, который начинается и/или заканчивается в нескольких состояниях

1 исходное * целевых

- Разветвление потока управления на несколько параллельных потоков
- Целевые состояния — вложены по одному на область

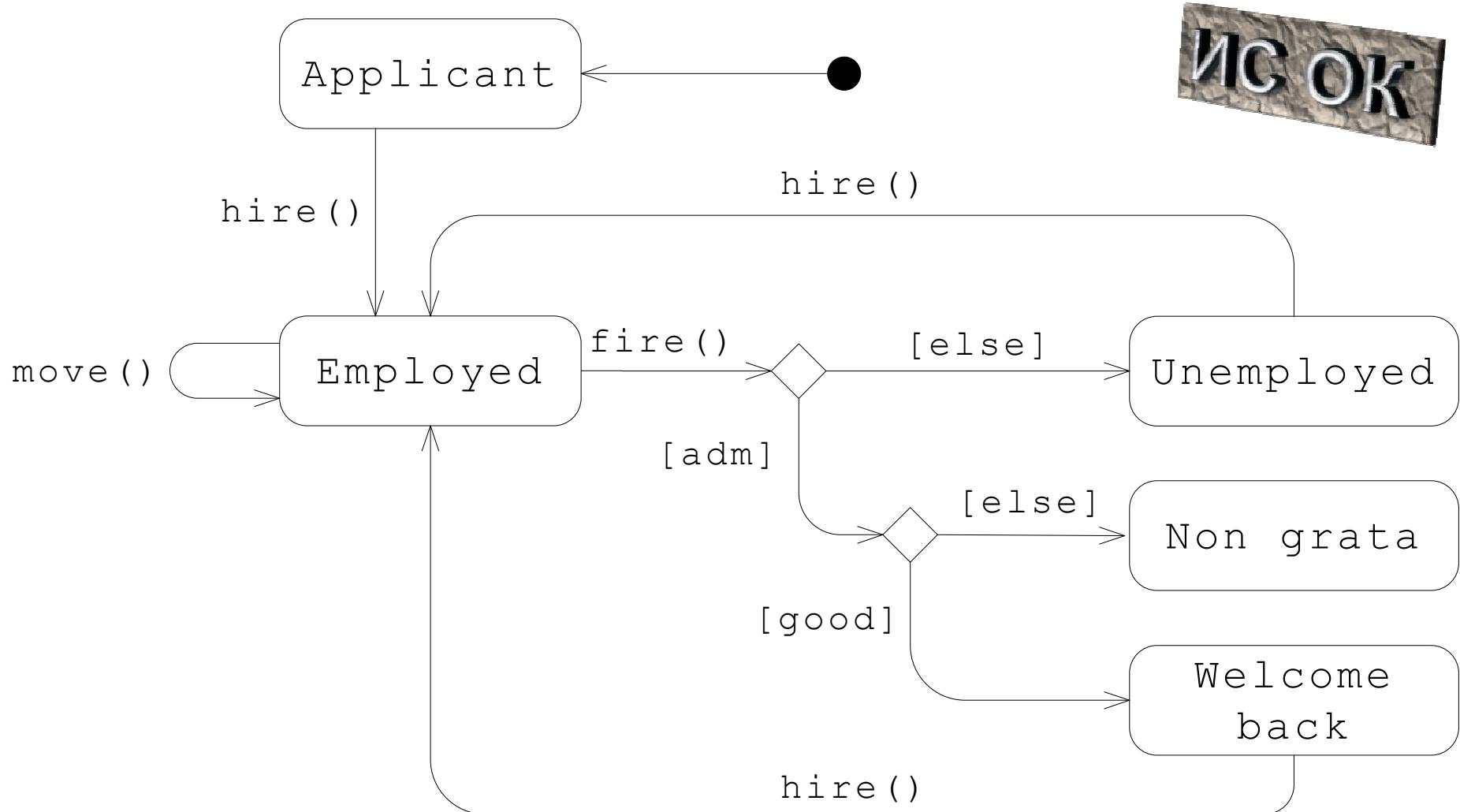
* исходных 1 целевое

- Слияние нескольких потоков управления в один
- Исходные состояния — вложены по одному на область

* исходных * целевых

- Синхронизация нескольких параллельных потоков управления
- Исходные и целевые состояния — вложены по одному на область

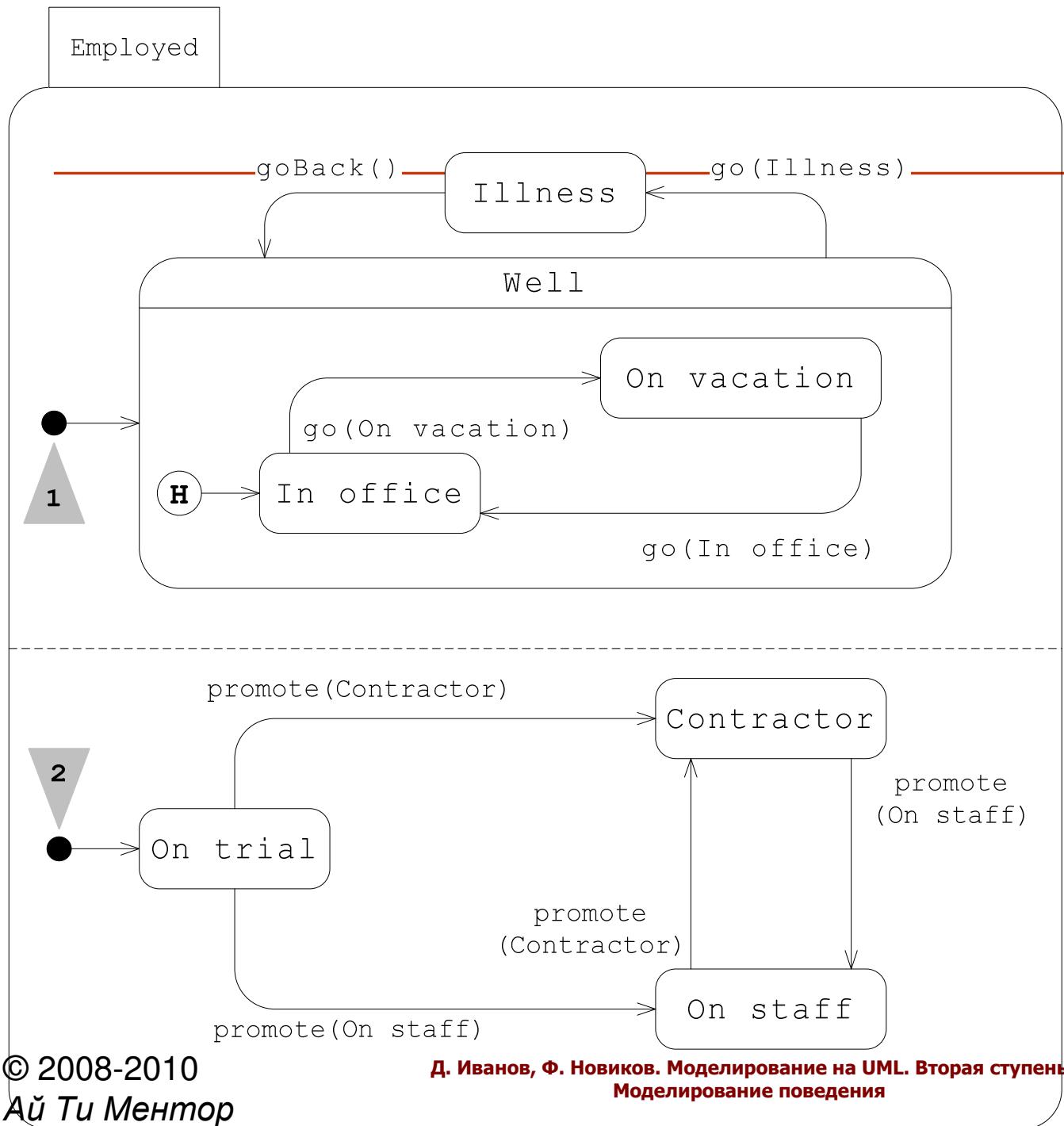
Диаграмма автомата сотрудника



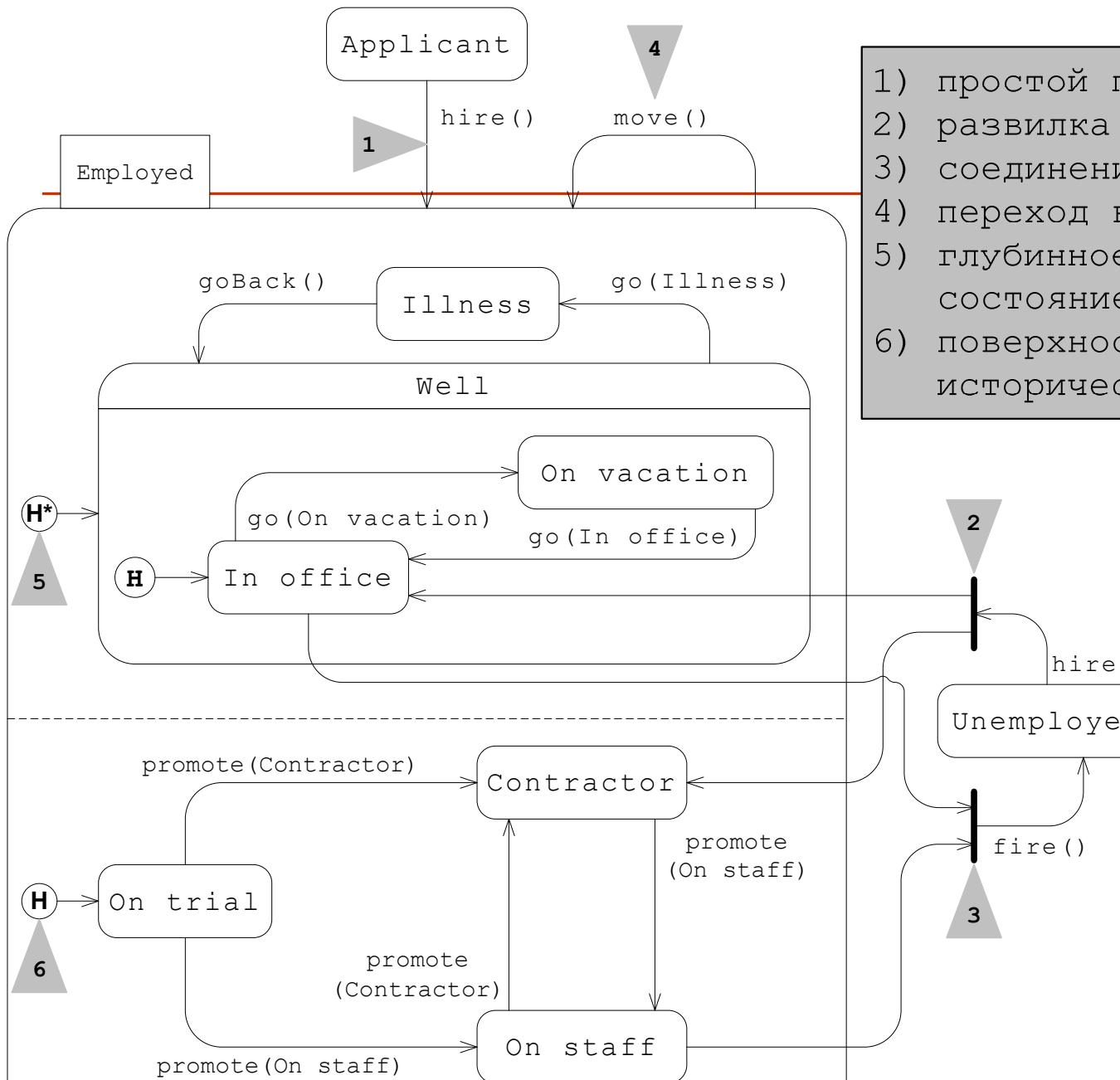
Ортогональные области

В состоянии Employed

ИС ОК



Составные переходы



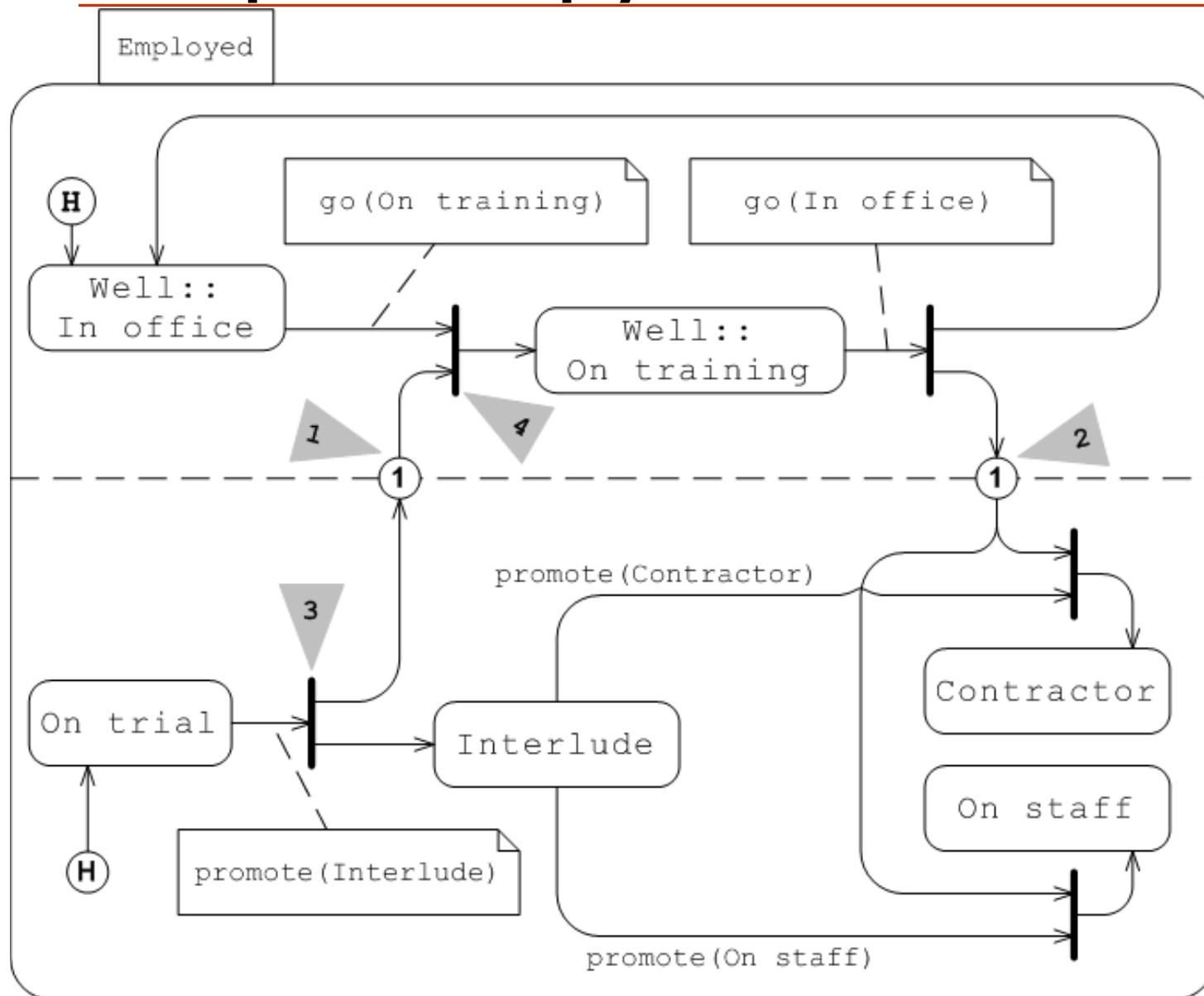
- 1) простой переход
- 2) разветвка
- 3) соединение
- 4) переход в себя
- 5) глубинное историческое состояние
- 6) поверхностное историческое состояние

ИС ОК

Синхронизирующие состояния (i)

- Используется в UML 1
- Синхронизирующие состояние (synch state): ставит срабатывание перехода в целевой машине в зависимость от срабатывания перехода в исходной машине
- Входные сегменты переходов: начинаются в развилках исходной машины состояний
- Выходные сегменты переходов: заканчиваются в соединениях целевой машины состояний

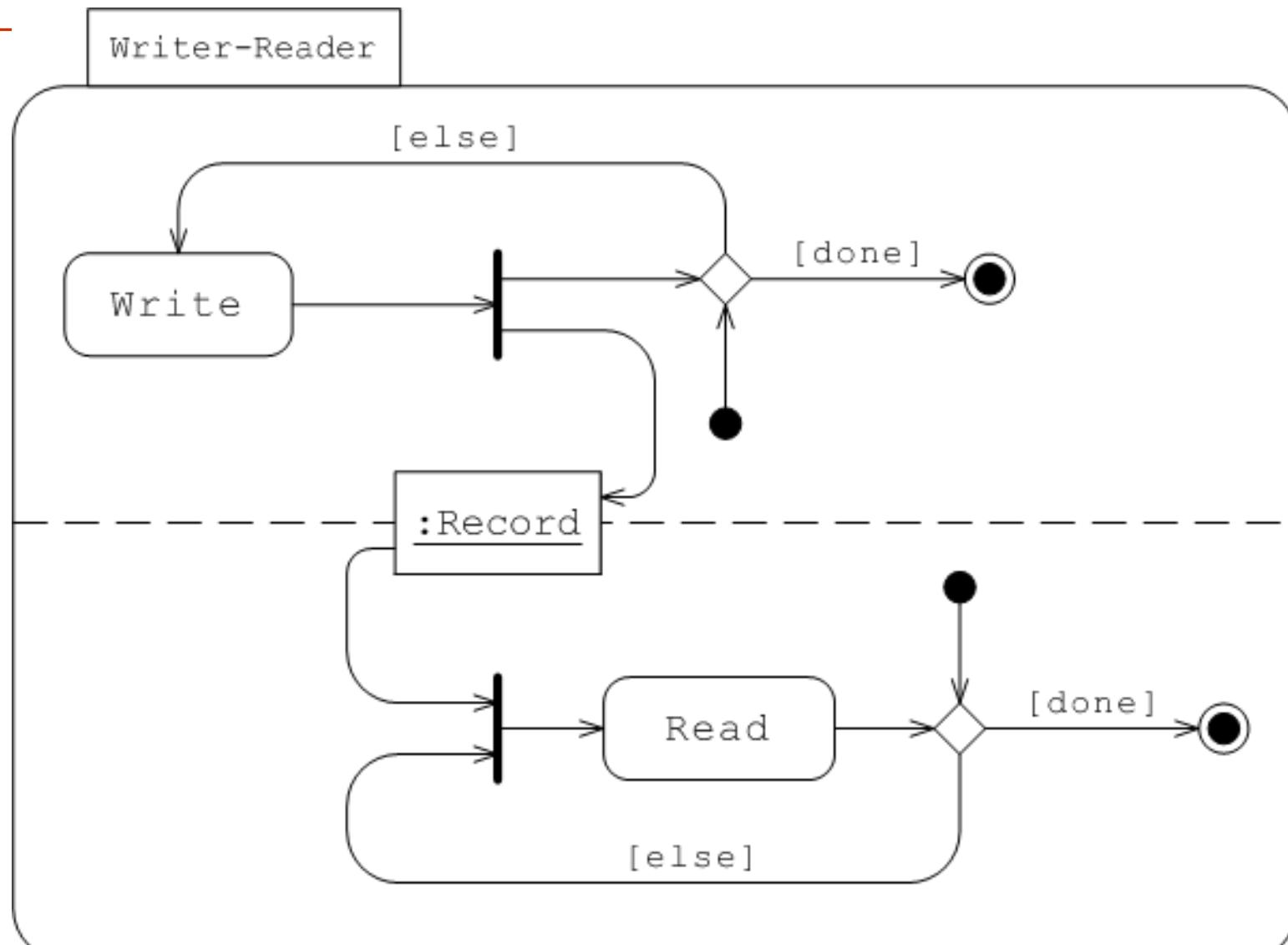
Синхронизирующие состояния (ii)



1, 2) синхро-
ниzierущее
состояние
3) разветвка
4) линейка
синхронизации

ИС ОК

Писатель и Читатель



Развилки и слияния

Составной переход по завершении

= развлка, слияние, линейка синхронизации

UML 1

- Сначала завершаются все деятельности, инцидентные входящим сегментам → параллельно запускаются все деятельности, инцидентные исходящим сегментам

UML 2

- Развилка создает копии пришедшего маркера на каждую исходящую дугу
- Слияние получает маркер по всем входящим дугам → отправляет маркер по исходящей дуге

Баланс разветвок и слияний

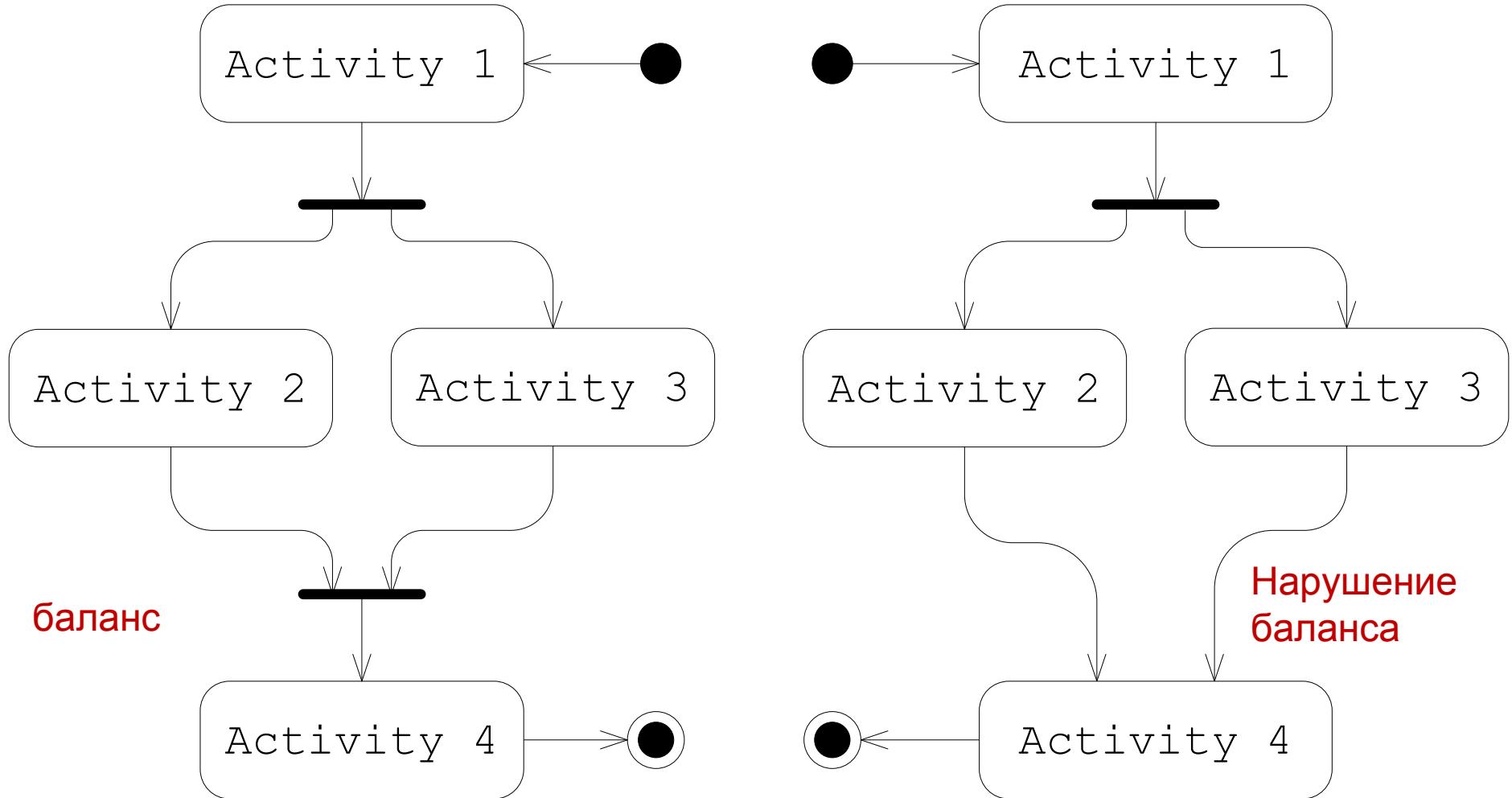
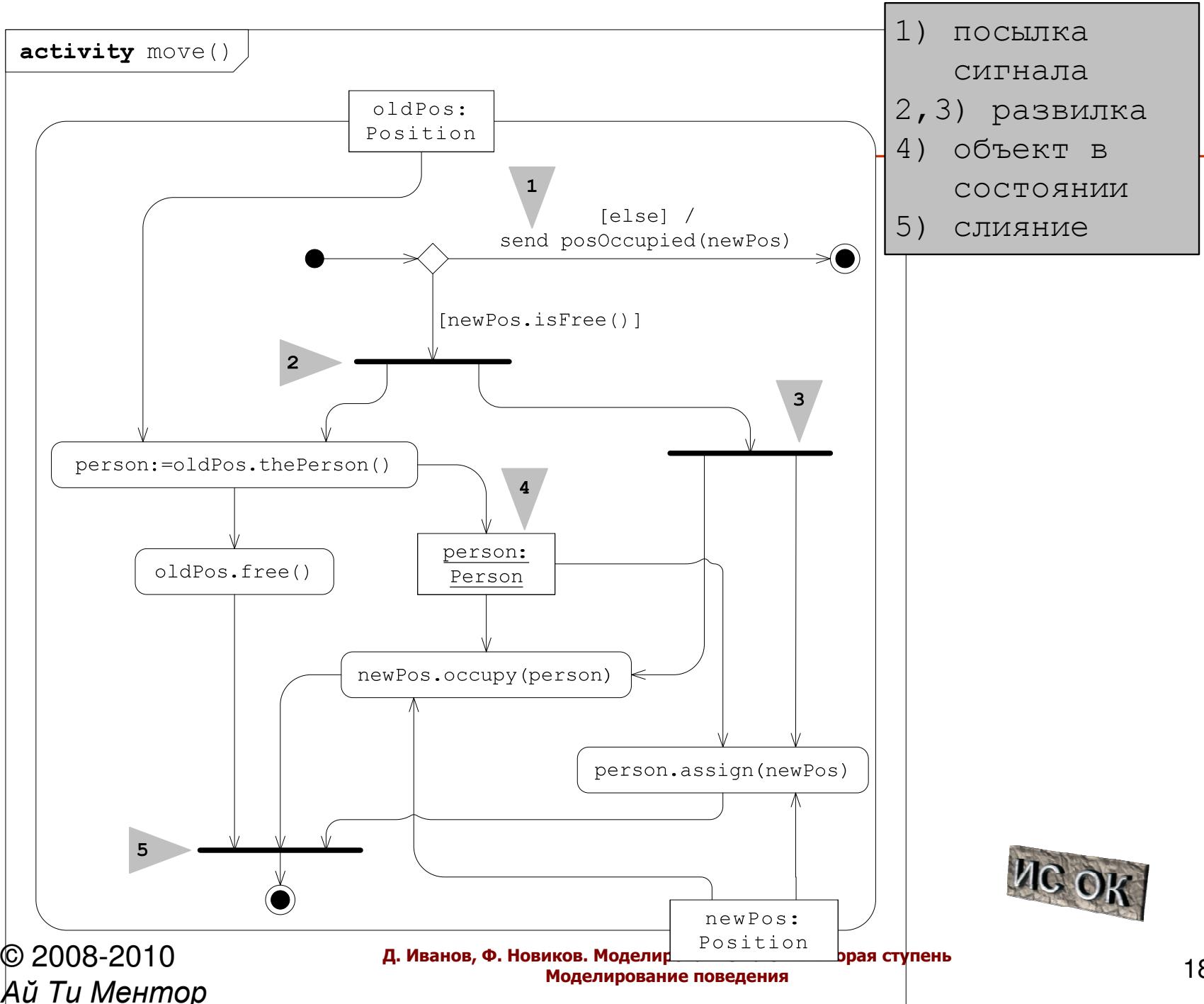


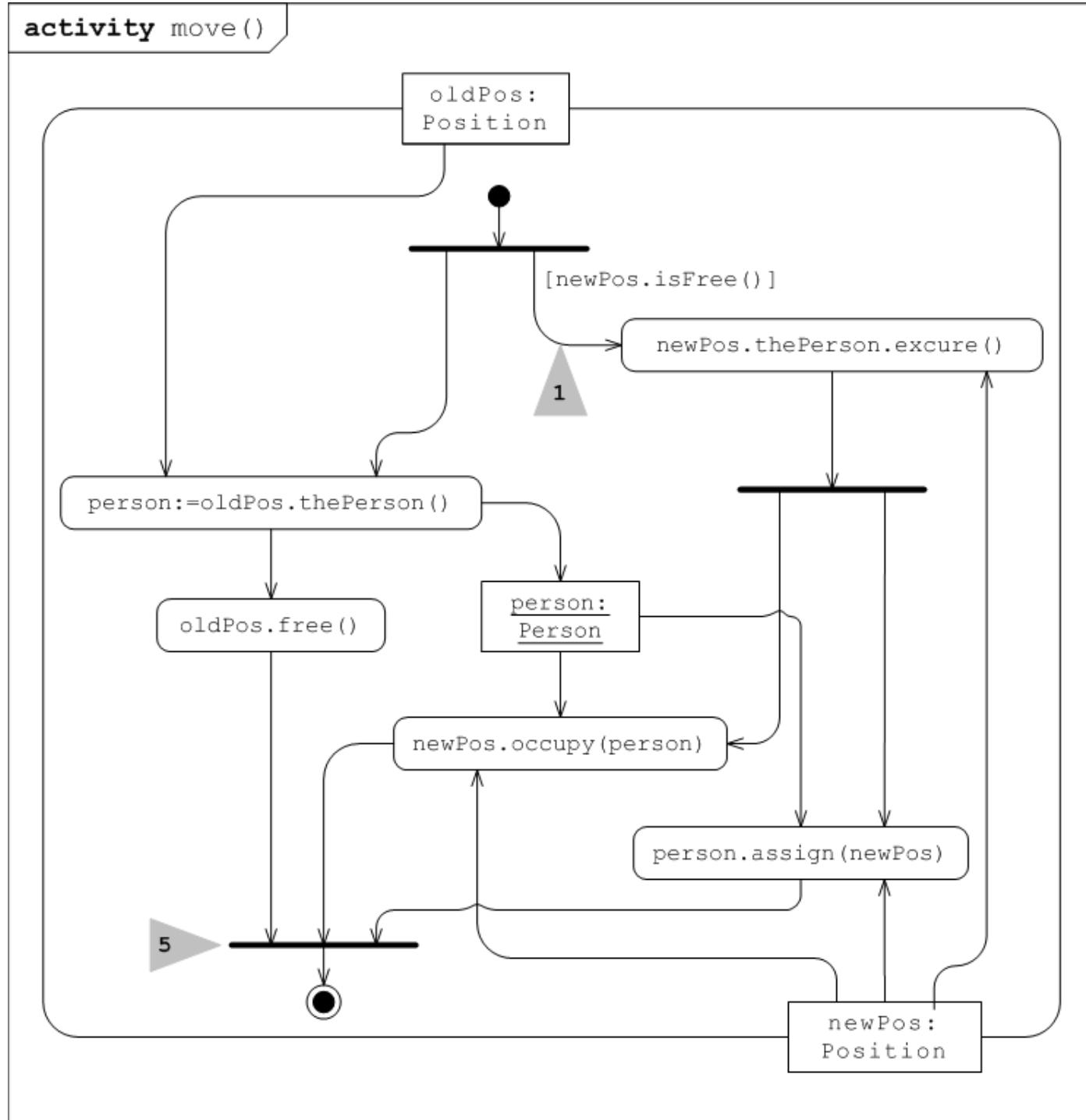
Диаграмма перевода сотрудника



ИС ОК

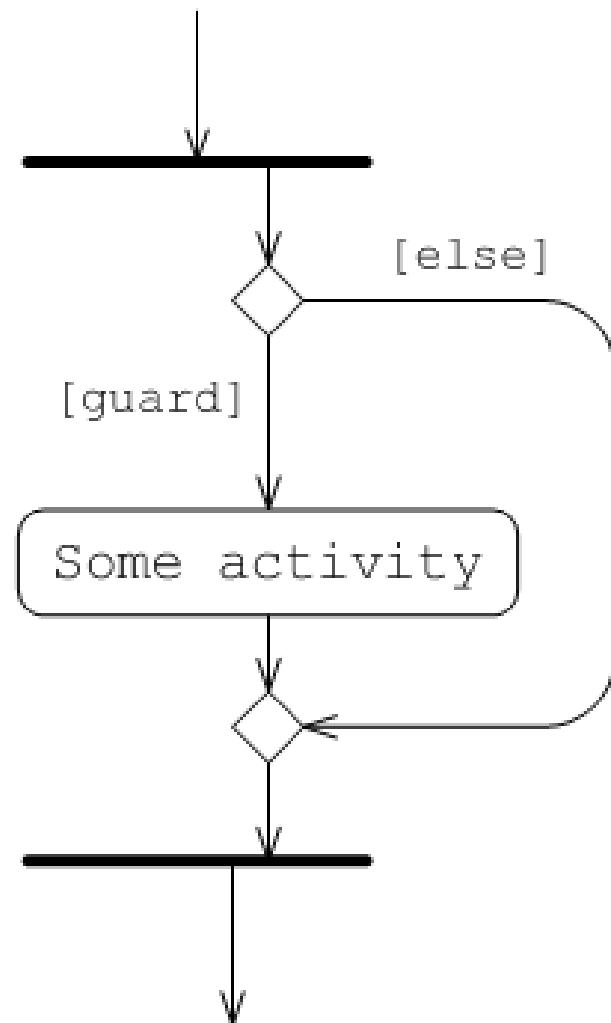
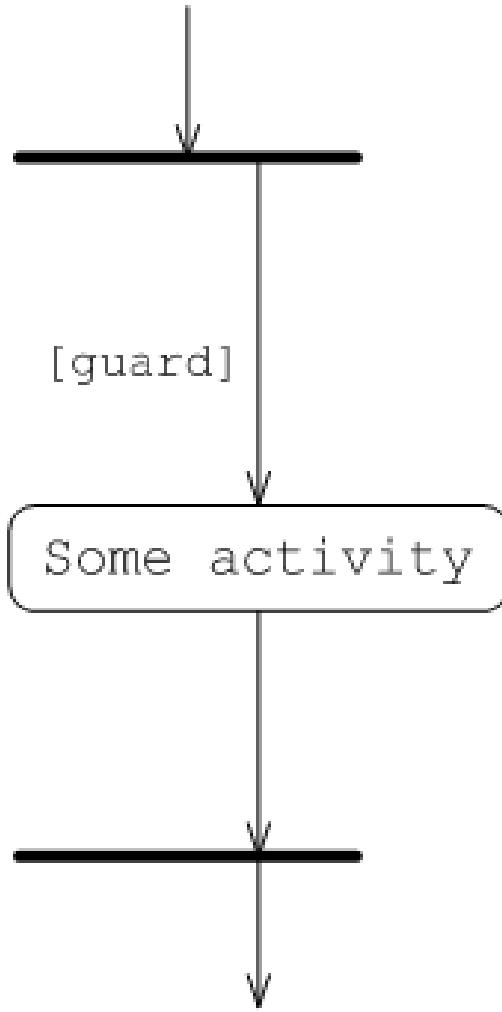
Обусловленный поток управления

исок



- 1) обусловленный поток управления
 - 5) слияние

Выражение обусловленного потока управления через ветвление



Параллелизм на диаграммах взаимодействия

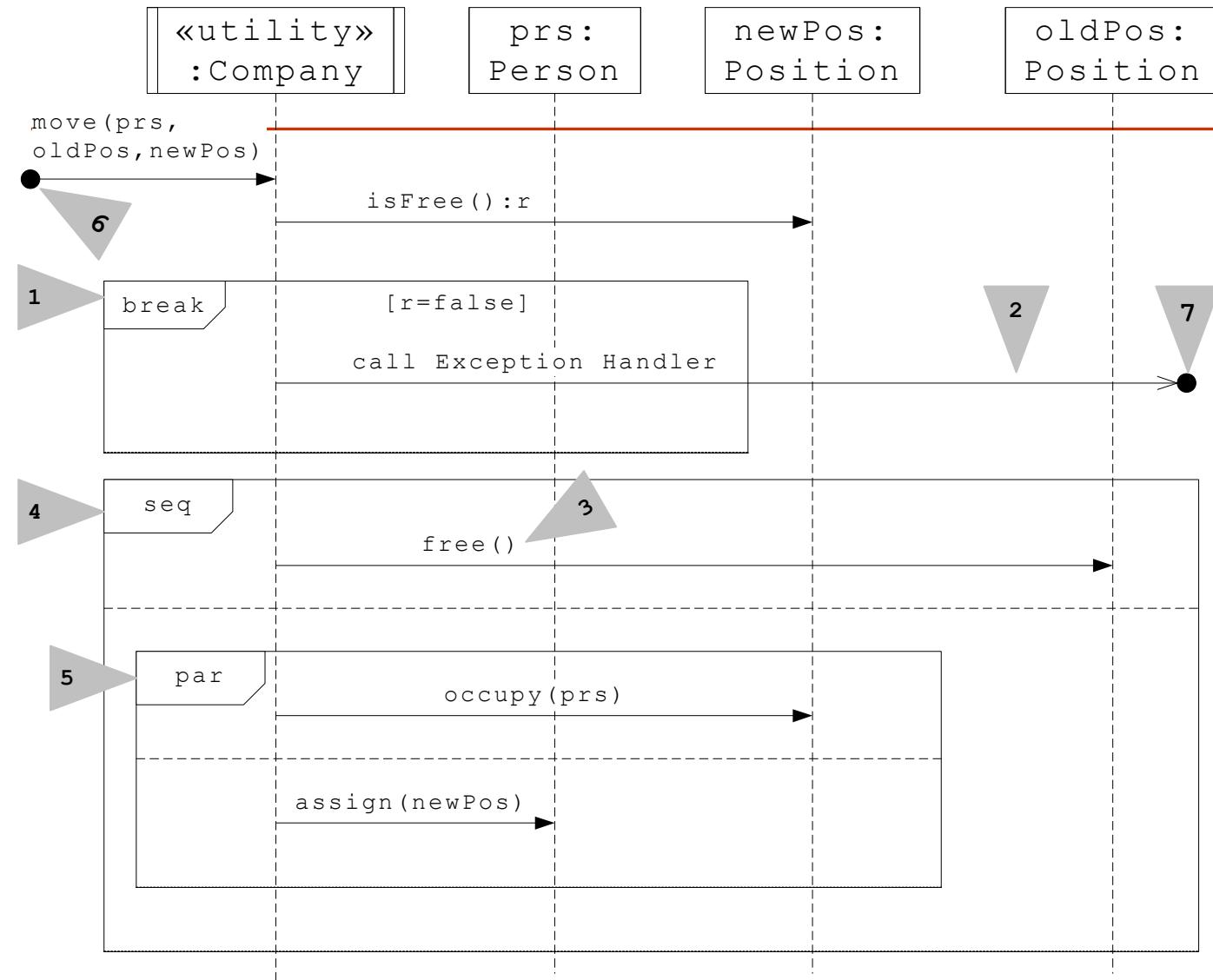
На диаграммах коммуникации

- Номера сообщений: цифры и буквы
- Символ || в повторителе

На диаграммах последовательности

- Исходят из одной точки (как и альтернативные!) – в UML 1
- Есть четыре типа составных шагов параллельного взаимодействия – в UML 2

sd Перевод служащего с одной должности на другую



Взаимодействие Параллельное

1, 4, 5) составные шаги взаимодействия

2) генерация исключения

3) вызов метода

6) найденное сообщение

7) потерянное сообщение

ис.ок

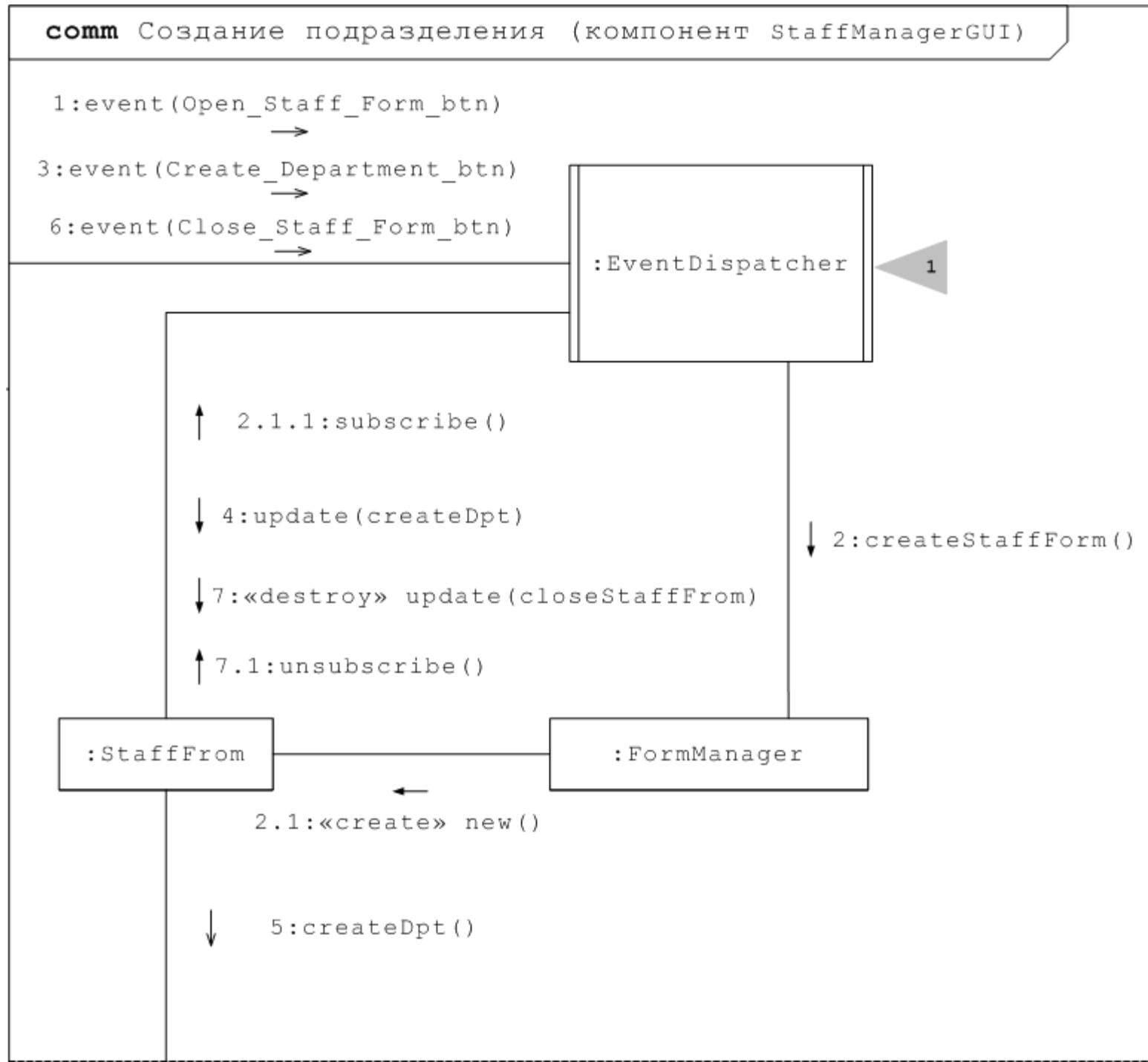
Активные классы

Активный класс —

- это класс, экземплярами которого являются активные объекты.
- это объект, имеющий собственный поток управления = это и **есть** поток управления

Пассивный объект — это объект класса, который **не** является активным

Диаграмма коммуникации с активным классом



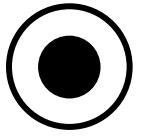
ИС ОК

6. Выводы (i)

- Конечные автоматы — базовая алгоритмическая техника моделирования поведения
- Диаграмма автомата (= граф переходов) — средство моделирования поведения явным выделением состояний
- Диаграмма деятельности (= блок-схема) — средство моделирования потоков управления и потоков данных
- Семантика диаграмм деятельности определена в UML 1 через конечные автоматы, а в UML 2 — через сети Петри

Выводы (ii)

- Диаграммы взаимодействия – последовательность сообщений при взаимодействии объектов
- Диаграммы взаимодействия – наиболее детальный способ описания поведения (протокол)
- Диаграммы коммуникации и последовательности семантически эквивалентны
- Диаграммы синхронизации показывают изменение состояния во времени
- Обзорные диаграммы взаимодействия являются комбинацией диаграмм деятельности и диаграмм взаимодействия



Выводы (iii)

- Параллелизм на диаграммах автомата задается ортогональными составными состояниями
- Параллелизм на диаграммах деятельности задается разветвлениями и слияниями потоков управления и данных
- Параллелизм на диаграммах последовательности задается специальными составными шагами взаимодействия
- Объект активного класса задает поток управления