**2. Реализация программой на псевдокоде -** этот способ

хорош тем, что понятен любому разработчику.

**Однако в настоящее время вряд ли можно рекомендовать такой способ реализации, как основной, по следующим причинам:**

1. Реализация на псевдокоде плохо согласуется с современной

парадигмой ООП.

2. При использовании псевдокода теряются все преимущества

использования UML.

3. Решения на псевдокоде практически невозможно

использовать повторно.

**Тем не менее, рассмотрим пример реализации вариантов**

**использования 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 (add\_payment)

if (from Self Fire)

начислить за неиспользованный отпуск

else if (from Adm Fire)

начислить выходное пособие

Увольнение по собственному желанию запускается по инициативе сотрудника. Увольнение по инициативе администрации начинается с приказа об увольнении. В остальном последовательность действий в обоих случаях совпадает.

В этих текстах использовано ключевое слово ***Include***, отражающее наличие зависимостей с таким стереотипом в модели. А именно, это означает, что в этом месте в текст псевдокода для данного варианта использования *нужно включить текст псевдокода для варианта использования Delete Account, который мы здесь не приводим.*

Вариант использования Pay Compensation запускается, если есть условия для выплаты компенсаций. При этом основные варианты использования не должны знать, каковы эти условия и как рассчитывается компенсация — за это отвечает вариант использования Pay Compensation. Зависимость со стереотипом «extend» означает, что псевдокод варианта использования Pay Compensation должен быть включен в текст основных вариантов использования. При этом вариант использования Pay compensation должен знать, в какое место ему нужно включиться. Для этого в основных вариантах использования определена **точка расширения (extension point)** — по сути, просто метка в программе.

**Для зависимости со стереотипом «include» никакой метки не**

**нужно: место включения определяется тем, где в псевдокоде стоит**

**ключевое слово Include.**

Этот пример в достаточной мере объясняет сходство и различие между зависимостями со стереотипом «include» и «extend».