

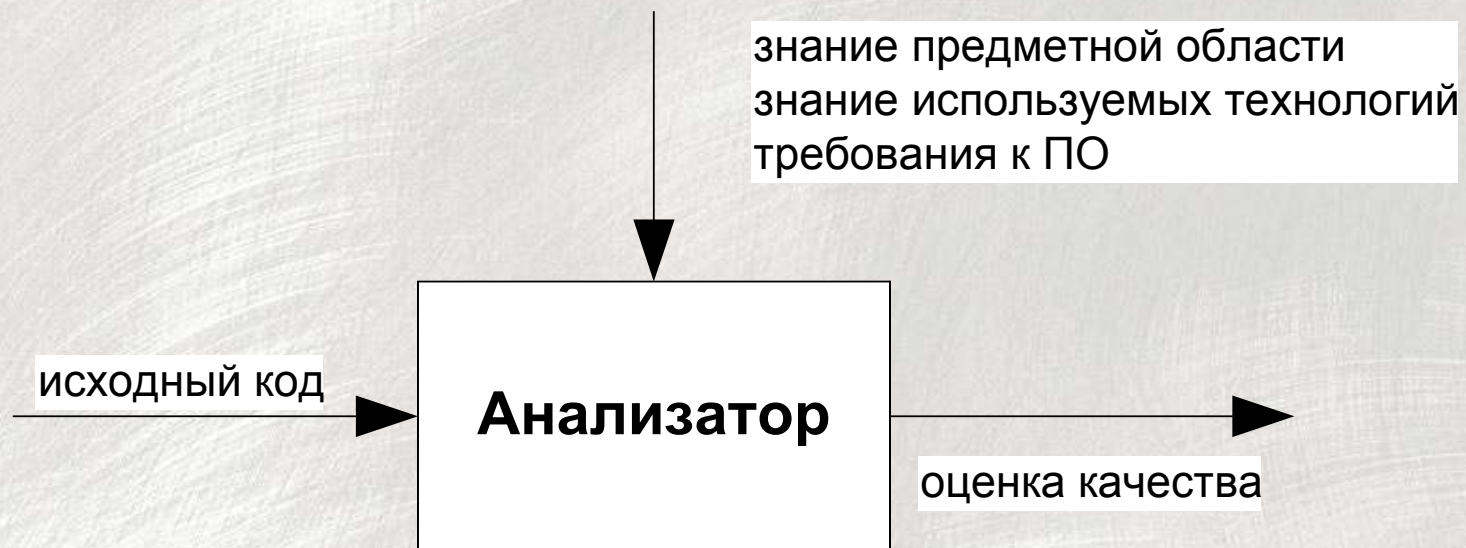
isv **ORACLE** **Forum V**

- Способ оценки критерия качества исходного кода
 - Пояснение к названию доклада
 - Качество ПО
 - Оценка качества ПО
 - Теоритические рассуждения
 - Практическое применения
 - Выводы

- Качество ПО – это нефункциональное (желательное) требование.

- Качество ПО влияет на
 - отладку (debugging)
 - тестирование (testing)
 - поддержку (maintenance)
 - внесение изменений (modification)
 - исправление ошибок (fixing)
 - и т.д.

- Оценка качества ПО
 - относительная

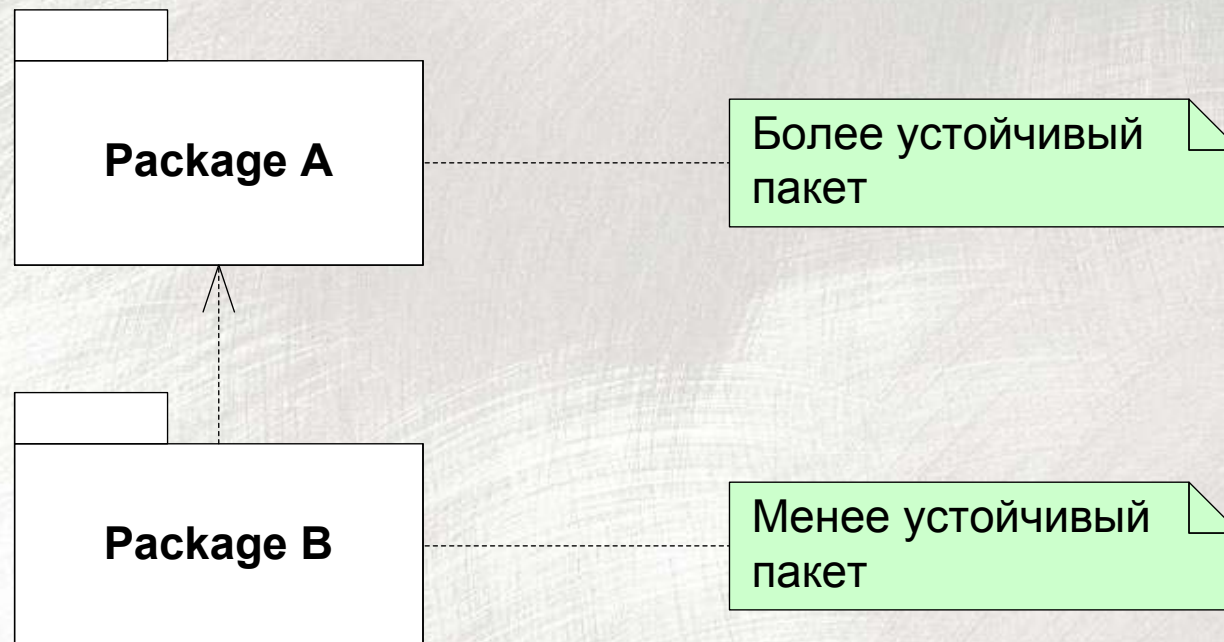


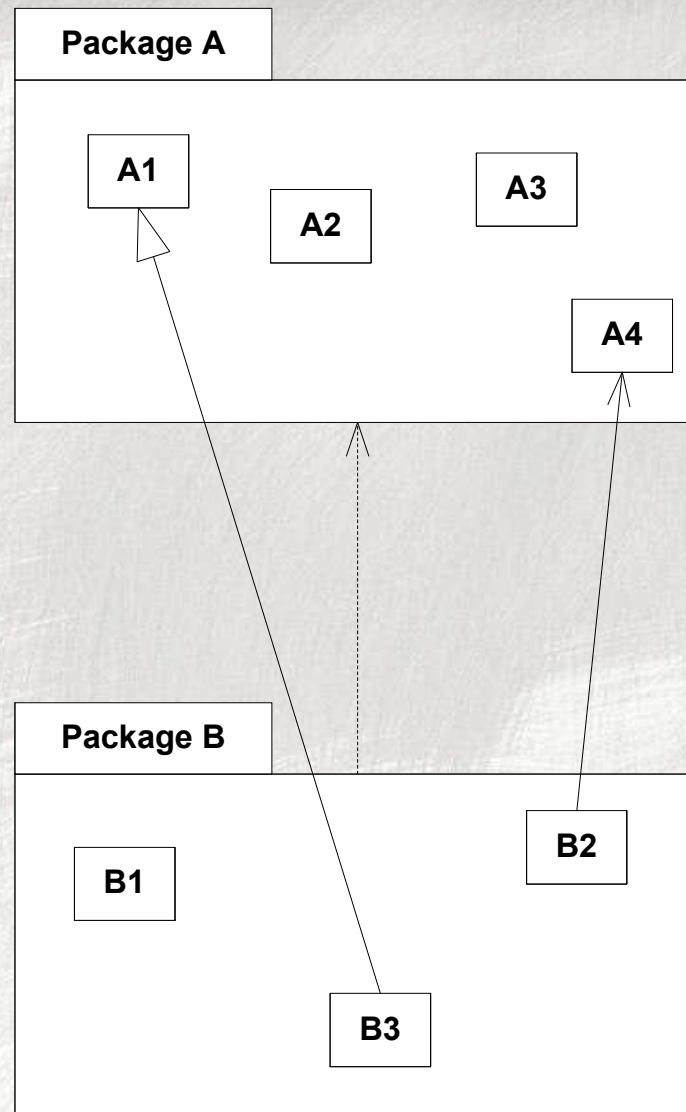
- Признаки «плохого» программного обеспечения
 - неустойчивость (fragile)
 - монолитность (not reusable)
 - вязкость (high viscosity)
 - и т.д.

- Анализ структуры, но не поведения

- Структурные уровни
 - уровень компонент
 - уровень пакетов (каталогов)
 - пакеты в UML
 - пакеты в Java \approx пространство имен
 - пакеты – каталоги
 - пакеты (by R. Martin) \approx компоненты
 - уровень классов

- The Stable Dependencies Principle (Принцип устойчивости/стабильности зависимостей)
 - *Depend in the direction of stability*

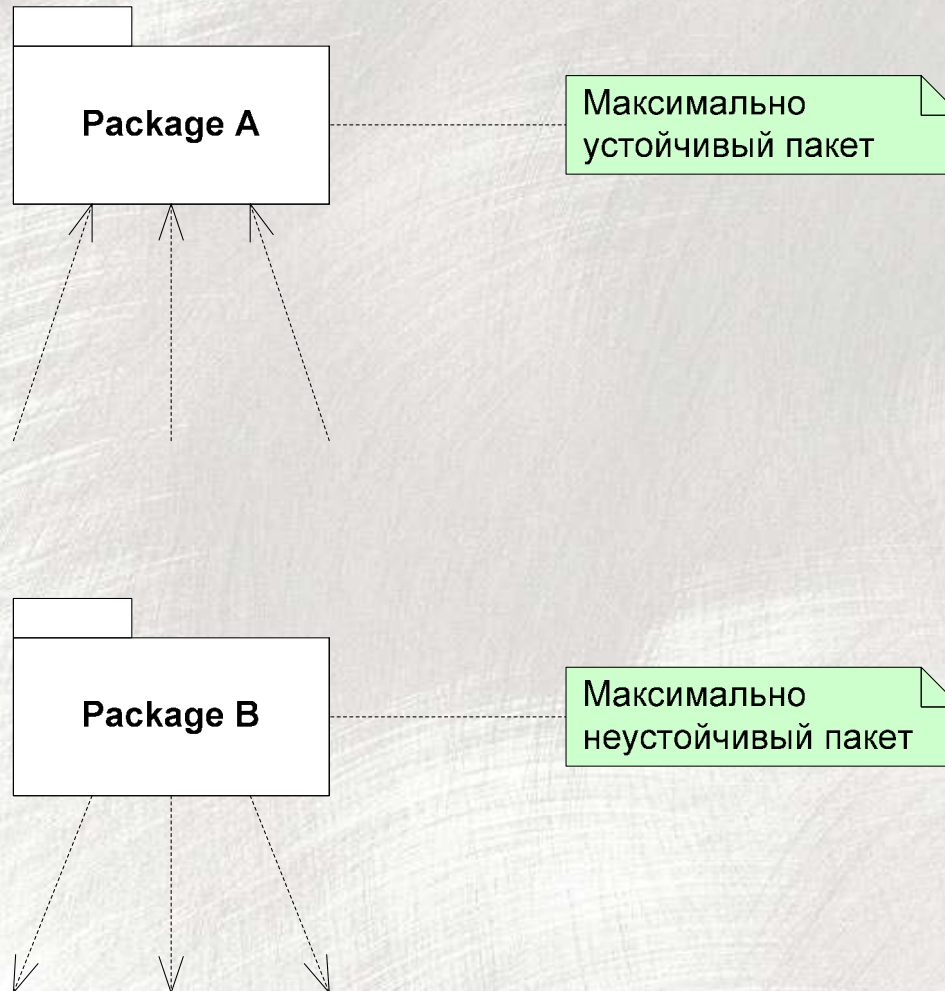




- C_a – количество классов **вне** пакета, которые зависят от классов, находящихся **внутри** пакета
- C_e – количество классов **внутри** пакета, которые зависят от классов, находящихся **вне** пакета
- I – неустойчивость пакета

$$I = \frac{C_a}{C_a + C_e}$$

- $I = 1$ – максимально **неустойчивый** пакет
- $I = 0$ – максимально **устойчивый** пакет



- Относится к пакету, но не ко всей системе
- Не учитывает косвенные зависимости
- Хорошо или плохо?

- Анализ структуры, но не поведения

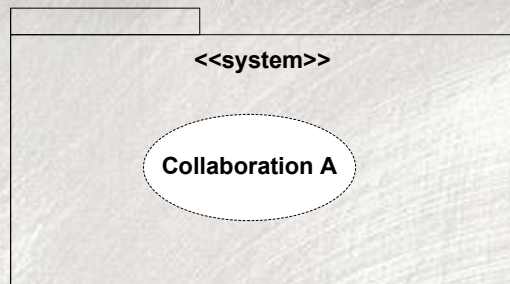
- Структурные уровни
 - уровень компонент
 - уровень пакетов (каталогов)
 - пакеты в UML
 - пакеты в Java \approx пространство имен
 - пакеты - каталоги
 - пакеты (by R. Martin) \approx компоненты
 - уровень коопераций классов
 - уровень классов

- все отношения между элементами кооперации
 - обобщение (generalization)
 - ассоциация (association)
 - агрегация (aggregation)
 - композиция (composition)
 - реализация (realization)
- = зависимость (dependency)

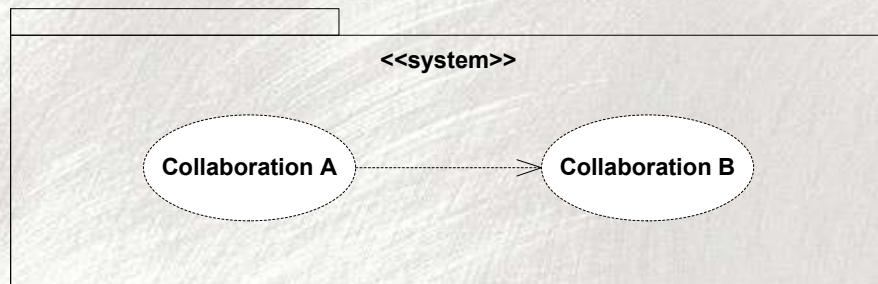
- Классы объединяются в кооперации исходя из структуры исходного кода
 - группировка классов по файлам
 - группировка внутренних классов

Неустойчивость кооперации к внесению изменений в программную систему – *приблизительная* вероятность того, что изменение в некоторой другой кооперации повлечет за собой изменение в исходной

- Относится к кооперации и может быть перенесено на всю систему
- Учитываются косвенные зависимости
- Хорошо или плохо?

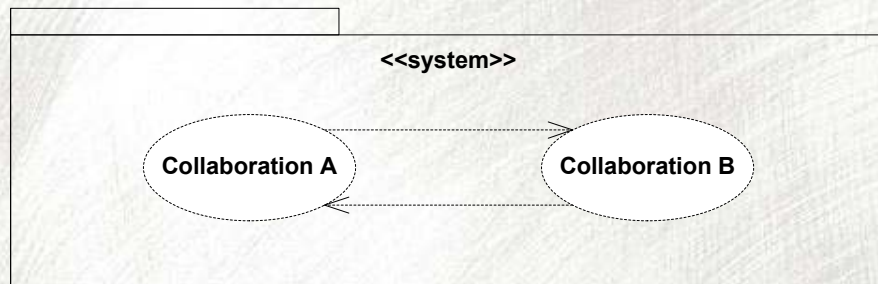


Неустойчивость кооперации A, $I_A = 1$



Неустойчивость кооперации A, $I_A = 1$

Неустойчивость кооперации B, $I_B = 1/2$

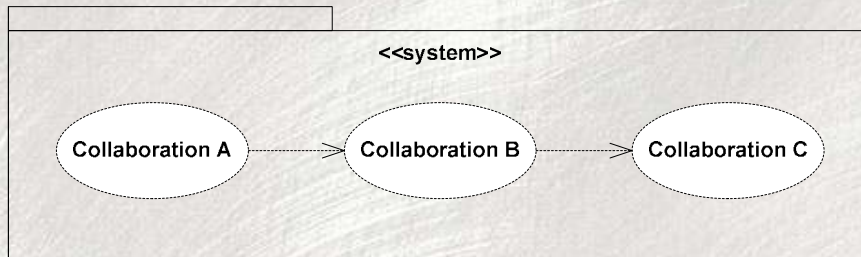


Неустойчивость кооперации A, $I_A = 1$

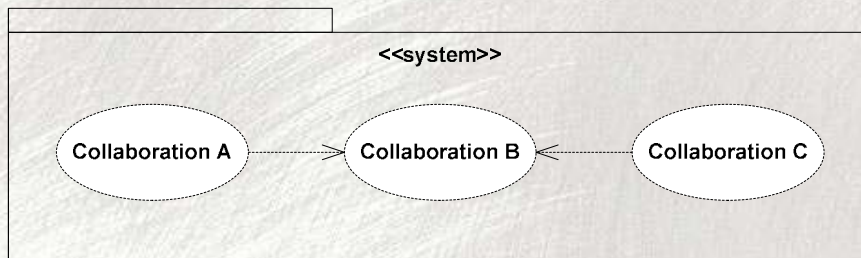
Неустойчивость кооперации B, $I_B = 1$

В качестве оценки неустойчивости системы I_{sys} будем использовать среднюю неустойчивость среди всех ее коопераций

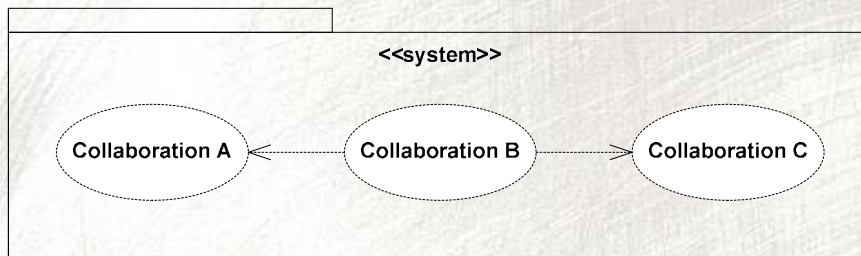
$$I_{sys} \approx \frac{1}{N} \sum_{i=1}^N w_i I_i, \quad w_i - \text{вес кооперации}$$



$$I_{sys} = (3/3 + 2/3 + 1/3) / 3 = 6/9$$

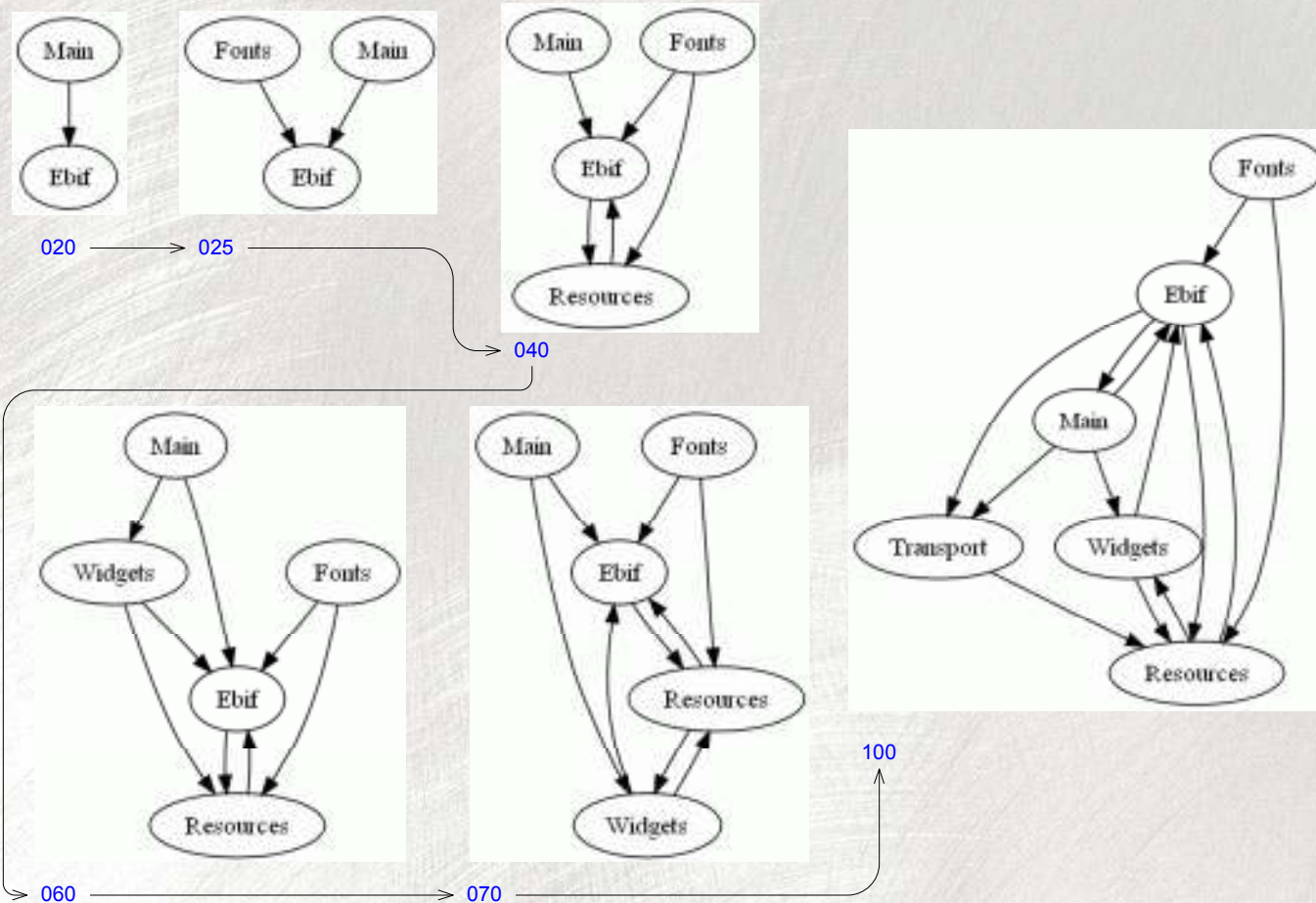


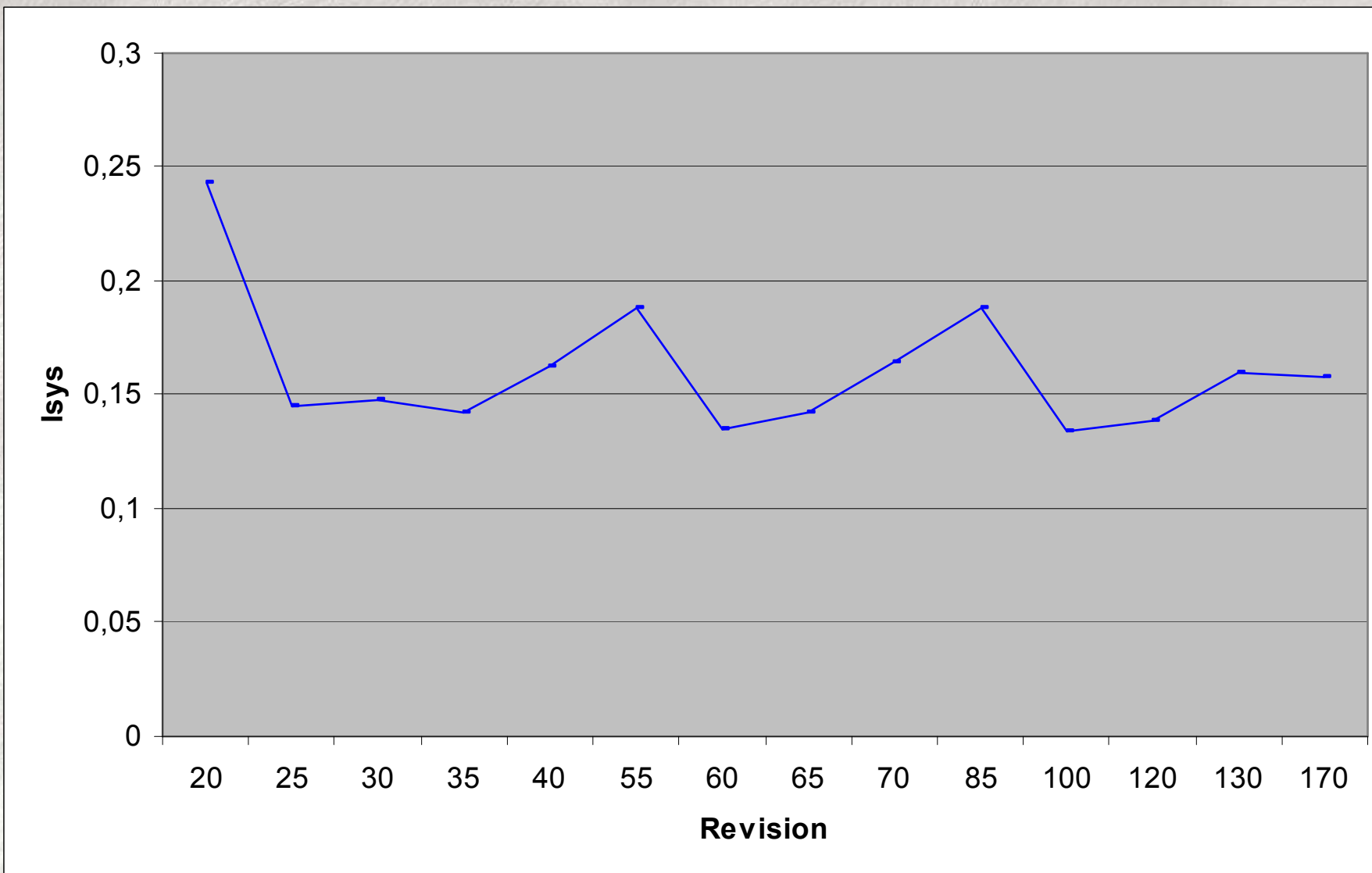
$$I_{sys} = (2/3 + 1/3 + 2/3) / 3 = 5/9$$



$$I_{sys} = (1/3 + 3/3 + 1/3) / 3 = 5/9$$

- Там, где возможно сравнение
 - Мониторинг (эволюция программного проекта)





- Автоматизация

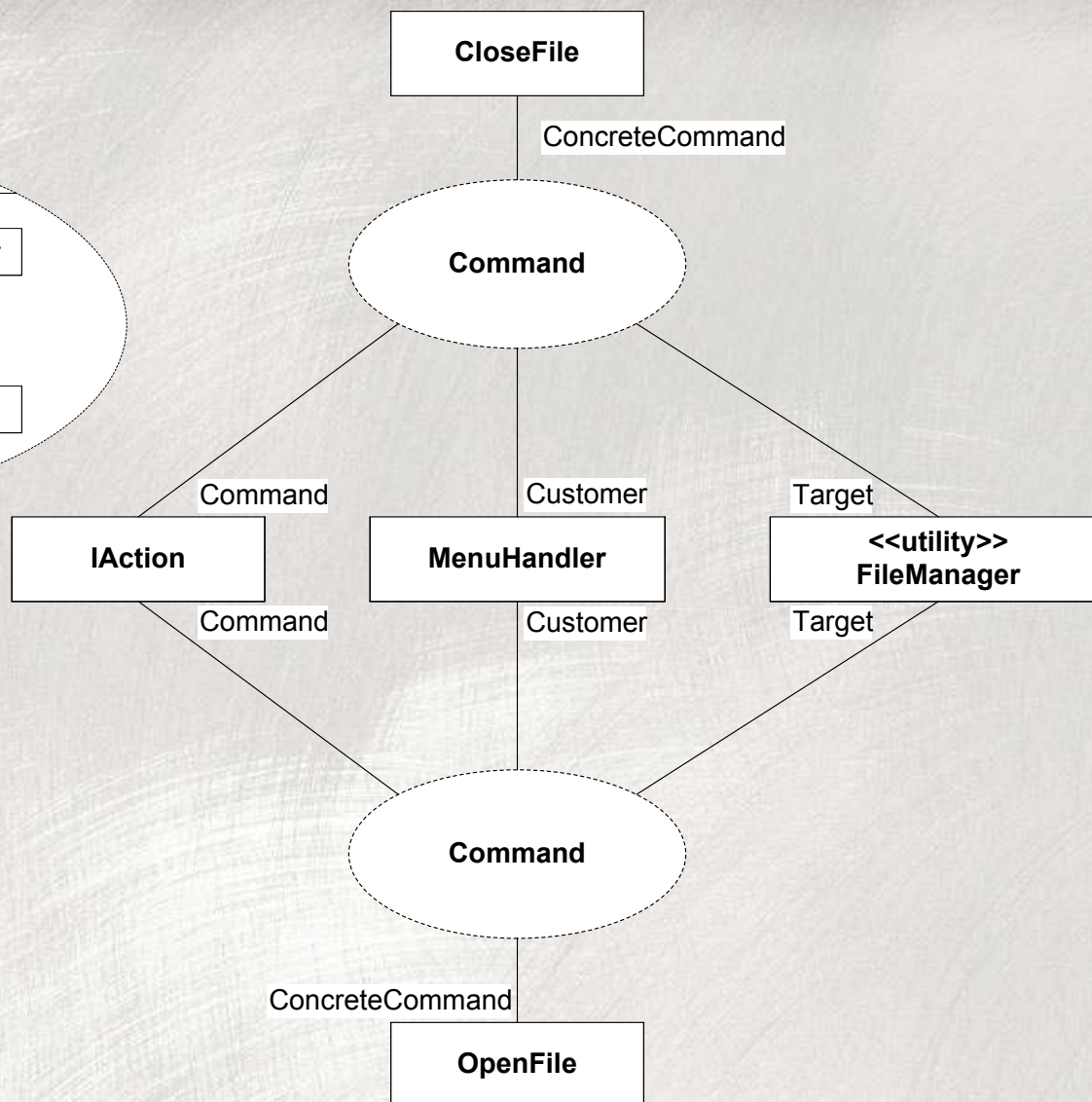
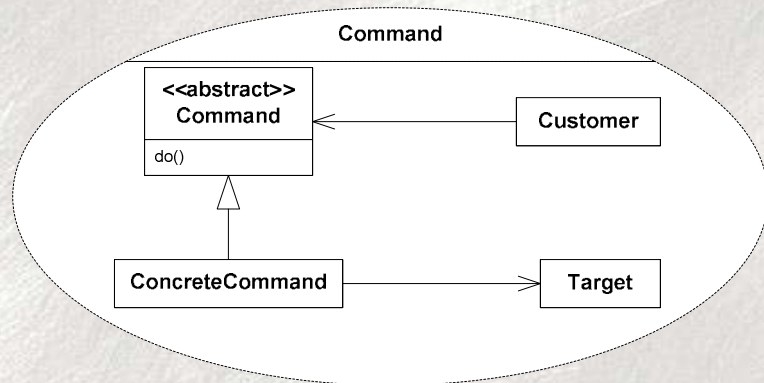
- Оценка - вероятность

- Настройка
 - кооперации/пакеты
 - веса

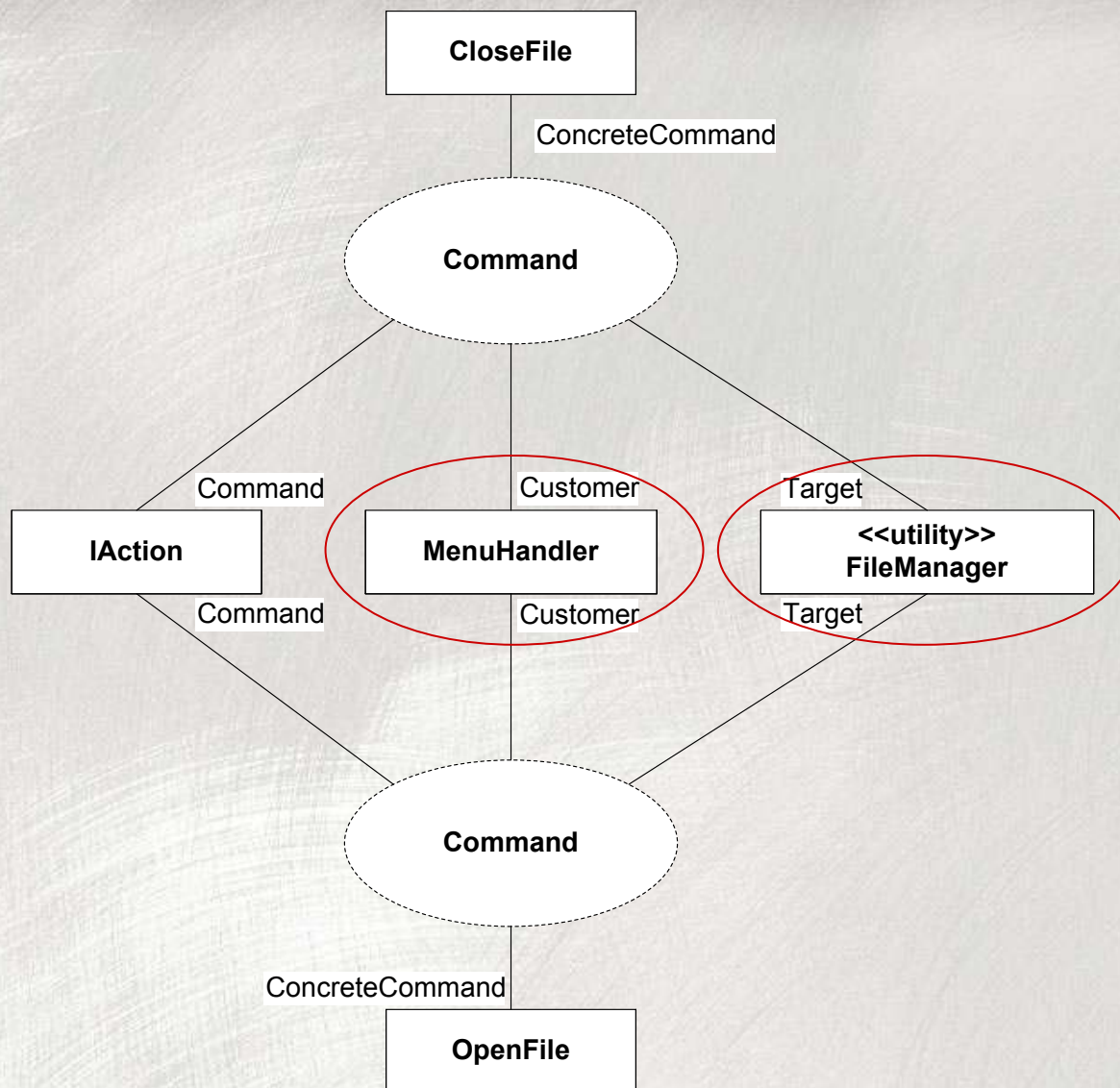
- <http://www.objectmentor.com>
- <http://www.uml.org>
- <http://ru.wikipedai.org>

Спасибо!

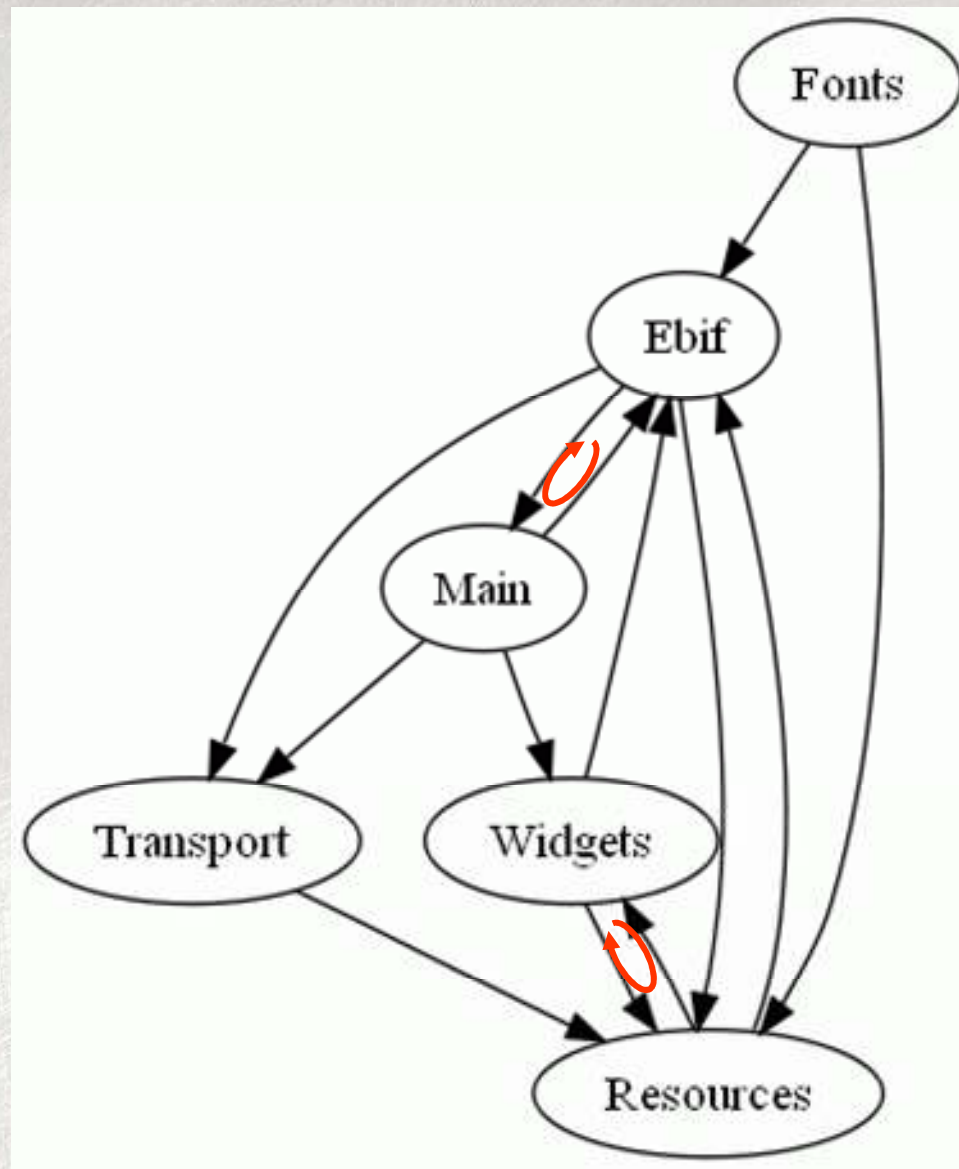
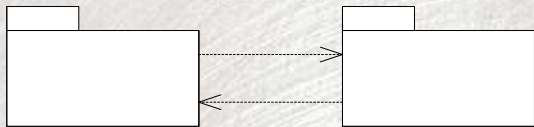
□ Кооперации и пакеты



□ Пакеты



□ Циклы



- Способ оценки качества исходного кода
 - Пояснение к названию доклада
 - Качество исходного кода
 - Оценка качества исходного кода
 - Область применения
 - Теоритические рассуждения
 - Практическая реализация
 - Выводы

