# Метрики программных систем

Технологии разработки программного обеспечения

Виноградова М.В.
МГТУ им. Н.Э. Баумана
Кафедра СОИУ (ИУ5)

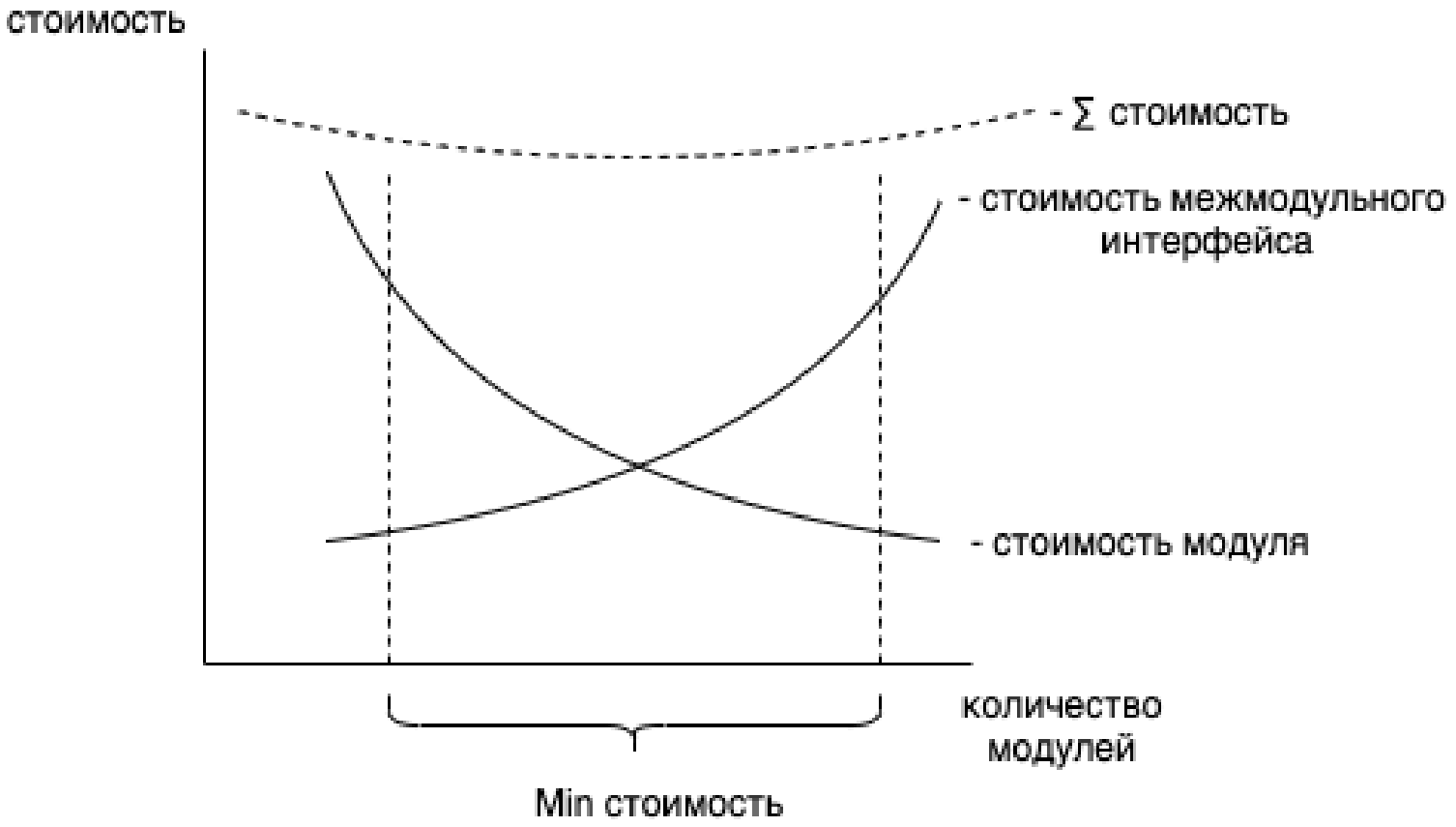
# Декомпозиция подсистем на модули

- Модель потока данных
  - Разбиение по функция
- Модель объектов
  - Разбиение на слабо сцепленные сущности, которые имеют свои данные, состояния и операции.
- Выбор модели определяется сложностью разрабатываемого ПО

- **Модуль** - фрагмент программного кода, являющегося строительным блоком для физической структуры системы
  **модуль = интерфейс + реализация**

- **Модульность** - свойство системы, которая может быть декомпозирована на множество внутренне связанных и слабо зависящих другой друга модулей (возможность создания сколь угодно сложной системы)

# Затраты при разработке модулей



# Информационная закрытость

- Модули независимы, обмен только информацией, необходимой для работы.
- Доступ к операциям и структурам данных модуля ограничен.
- Возможность разработки независимыми коллективами.
- Легкость модификации (ошибки не распространяются)

# Характеристики модуля

- Связность
  - Мера зависимости частей модуля;
  - Внутренняя характеристика;
  - Степень «черноты» ящика;
  - Определение в силе связанности СС (от 0 до 10);
  - Цель: ↑
- Сцепление
  - Мера взаимозависимости модулей по данным;
  - Внешняя характеристика;
  - Определение в силе связанности СЦ (от 0 до 9);
  - Цель: ↓

# Связность модуля

- Функциональная (CC = 10)
  - Выполняется 1 проблемная задача;
  - Ничего лишнего
    - «Вычислить sin угла» - просто
    - «Вычислить зарплату сотрудника» - сложно, могут быть подфункции
  - «Черный ящик», оптимален
- Информационная (CC=9)
  - Конвейер для обработки данных
  - «прием и проверка записи»
  - **(+)** Почти «черный ящик» - хорошо сопровождать
  - (−) Хуже повторно использовать, так как последовательность действий нужна не всегда

# Связность модуля - 2

- Коммуникативная (CC=7)
  - Части модуля работают с одними и теми же данными
  - «Отчет и средняя з/п»
  - (−) Серый ящик, избыточность данных для повторного использования.
  - **(+)** Хорошо сопровождается, т.к. Минимальное количество внешних данных.
- Процедурная (CC=5)
  - Порядок выполнения действий реализует некий сценарий поведения.
  - Действия независимы, но связаны порядком выполнения.
  - «Вычисление средних значений»
  - (−) очень трудно модифицировать.
  - (−) хуже сопровождать «белый ящик».
  - Нет связи по данным, есть общий порядок передачи управления.

# Связность модуля - 3

- Временная (CC = 3)
  - Части модуля не связаны, но необходимы в 1 момент времени.
  - «Инициализация средних значений»

    Текущая таблица = NULL;

    Счетчик = ø;

    ∑ = ø;

    функция ввода = false

    функция инициализации = true;

    КОНЕЦ
  - (−) «белый ящик»:
    - При программной оптимизации сложно использовать повторно.
    - Необходимо дублировать код при необходимости инициализировать часть системы или добавить влаги.
    - Сложные внешние связи.

# Связность модуля - 4

- Логическая (CC = 1)
  - Объединение действий по функциональному подобию, например обработки ошибок. При использовании выполняется 1 из пунктов программы.
  - «Пересылка сообщений»
  - Отличия: 1 интерфейс, № функций;
  - Смысл параметра зависит от действия => пустые параметры с учетом типа.
  - Много параметров; сложный интерфейс; сложная и запутанная внутренняя структура.
  - Сложно понять; Сложно сопровождать.
- По совпадению (CC = 0)
  - Нет явных внутренних связей.
  - «Разное» (параметры).
  - Нет связей по дополнительному времени, управлению.
  - «Абсолютно белый ящик».
  - Проще без модульности.
  - Могут появиться из временной связности, при их усложнении.

# Определение связности модуля

- Если модуль - единственная проблемно-ориентированная функция, то функциональная иначе
- Если действия связаны:
  - Если связаны по данным
    - Если порядок действий важен, то информационная,
    - иначе - коммуникативная
  - Если связаны по управлению и порядок действий важен, то процедурная
    - Если порядок действий не важен, то временная.
- Если действия не связаны и $\in$ 1 категории, то логическая.
- Если действия не связаны и $\notin$ 1 категории - то по совпадению.

# Сцепление модулей

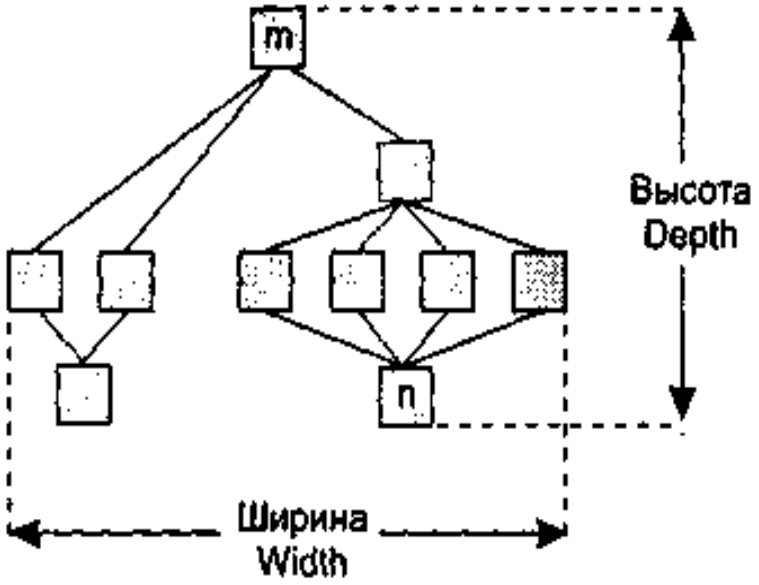
- По данным (СЦ=1)
  - Модуль А вызывает модуль В.
  - Входные и выходные параметры вызываемого модуля - простые элементы данных.
- По образцу (СЦ=3)
  - В качестве параметров используются структуры данных.
- По управлению (СЦ=4)
  - Модуль А явно управляет работой модуля В, посылая ему управляющие данные (флаг или переключатель).

# Сцепление модулей - 2

- По внешним ссылкам (СЦ=5)
  - Модули A и B ссылаются на 1 глобальный элемент данных.
- По общей области (СЦ=7)
  - Модули A и B разделяют одну глобальную структуру данных.
- По содержанию (СЦ=9)
  - Модуль A прямо ссылается на содержание модуля B (не через точку входа).
  - Например, их коды перемежаются.

# Иерархическая структура программной системы

- Количество модулей (вершин).
- Количество ребер (связей между модулями).
- **высота** - количество уровней управления (≈4).
- **ширина** - мах количество модулей на 1 уровне управления (≈6).



# Сложность модуля

- **мера длины модуля** (Холстед, 1977)

  **N = n1 x log2(n1) + n2 x log2(n2)**

  n1 - количество операторов

  n2 - количество операндов

- **объем модуля** (количество символов для записи всех операций и операндов)

  **V = N * log2(n1+n2) (бит)**

  n1 - количество операторов

  n2 - количество операндов

- **метрика цикломатической сложности**

  **V(g) = E - N + 2**

  E - количество дуг;

  N - количество вершин в управляющем графе ПС

# Невязка

- Отличие структуры от дерева
- Лучшая структура- дерево - Et.
- Худшая – полный граф – Ec.
- Невязка - величина расхождения (отличия от дерева).
- Невязка: от 0 до 1.
- (n – кол.вершин, e – кол.ребер)

$$E_c = \frac{n * (n-1)}{2}$$

$$E_t = n - 1$$

$$Nev = \frac{e - e_t}{e_c - e_t} = \frac{(e-n+1)\times 2}{n\times(n-1) - 2\times(n-1)} = \frac{2\times(e-n+1)}{(n-1)\times(n-2)}$$

# Дополнительные характеристики иерархической структуры

- коэффициент объединения по входу Fan_in(i)
  - количество модулей, которые управляют i-ым модулей.
- коэффициент объединения по выходу Fan_out(i)
  - количество модулей, которыми прямо управляет i-ый модуль.

  - Большое значение Fan_in(i) — свидетельство высокого сцепления, так как является мерой зависимости модуля.
  - Большое значение Fan_out(i) говорит о высокой сложности вызывающего модуля.

# Дополнительные характеристики иерархической структуры - 2

- Информационный коэффициент ifan_in(i)
  - количество элементов и структур данных, из которых i-й модуль берет информацию
- Информационный коэффициент ifan_out(j)
  - количество элементов и структур данных, которые обновляются j-м модулем.

# Метрика общей сложности структуры ПС

$$S = \sum_{i=1}^{n} \text{length}(i) \text{ x } (\text{Fan\_in}(i) + \text{Fan\_out}(i))^2,$$

Fan_in (i) = Fan_in (i) + ifan_in (i),

Fan_out (j) = Fan_out (j) + ifan_out (j).

length(i) — оценка размера i-го модуля
(в виде LOC- или FP-оценки)

# Метрики объектно-ориентированные

- метрики Чидамбера и Кемерера, 1994г.
  - проектные метрики, ориентированные на классы
- Метрики Лоренца и Кидда
  - результат практического, промышленного подхода к оценке ОО-проектов.
- Метрики Фернандо Абреу

# Метрики Чидамбера и Кемерера

- Взвешенные методы на класс - WMC (Weighted Methods Per Class)
  - считать только методы класса, без унаследованных.
  - считать все методы, в т.ч. числе унаследованные.
  - промежуточный. Например методы класса и его родителя.
  - WMC - относительная мера сложности классов.
- Высота дерева наследования - DIT(Depth of Inheritance Tree)
  - Для отдельного класса - это длина максимального пути от данного класса до корневого класса в иерархии классов.
  - У нижнего уровня много методов => трудно предсказать поведение класса. Возрастает сложность проекта, многие методы могут выполняться многократно.

# Метрики Чидамбера и Кемерера

- Количество детей - NOC (Number of children)
  - NOC = количество непосредственных наследников класса.
  - ↑NOC = ↓абстракции родительского класса (-> часть детей может быть использована неправильно).
  - ↑количества тестов для проверки ребенка.
  - ↑многократного использования.
- DIT и NOC определяют форму и размер структуры классов.
- Следует строить сбалансированные по высоте и ширине структуры наследования: обычно не выше, чем 7 ± 2 уровня, и не шире, чем 7 ± 2 ветви.

# Метрики Чидамбера и Кемерера

- Сцепление между классами объектов - CBO (Coupling between object classes)
  - CBO = количество сцеплений класса (вызов метода или свойства другого класса)
  - ↑CBO = ↓повторное использование класса.
  - Высокое значение CBO усложняет модификацию и тестирование. Минимизация межобъектных сцеплений улучшает модульность и содействует инкапсуляции проекта
- Отклик для класса - RFC (Response For a Class)
  - RFC = количество методов в множестве отклика.
  - Множество отклика - это множество методов класса и вызовов методов других классов из данного класса.
  - С ростом RFC увеличивается сложность класса.
  - Определяет временя тестирования.

# Метрики Чидамбера и Кемерера

- Недостаток связности в методах - LCOM (Lack of Cohesion in Methods)
  - Определяет, насколько методы не связаны друг с другом через свойства (переменные). Если все методы обращаются к одинаковым свойствам, то LCOM = 0.
  - N - количество пар методов без общих экземплярных переменных;
  - S - количество пар методов с общими экземплярными переменными.
  - Если LCOM имеет высокое значение, то методы слабо связаны друг с другом через свойства. Это увеличивает сложность, в связи с чем возрастает вероятность ошибок при проектировании.

$$LOCM \begin{cases} N - S, \text{если } N > S \\ 0, \text{иначе} \end{cases}$$

# Метрики Лоренца и Кидда

- Коллекция метрик Лоренца и Кидда — результат практического, промышленного подхода к оценке ОО-проектов.

- Размер класса CS (Class Size)
  - общее количество операций (вместе с приватными и наследуемыми экземплярными операциями), которые инкапсулируются внутри класса;
  - количество свойств (вместе с приватными и наследуемыми экземплярными свойствами), которые инкапсулируются классом.
  - Большие значения CS указывают, что класс имеет слишком много обязанностей. Они уменьшают возможность повторного использования класса, усложняют его реализацию и тестирование.
  - Рекомендуемое значение CS <= 20 методов.

# Метрики Лоренца и Кидда

- Количество операций, переопределяемых подклассом, NOO (Number of Operations Overrided by a Subclass)
  - Большие значения NOO обычно указывают на проблемы проектирования, так же нарушается абстракция суперкласса, ослабляется иерархия классов, усложняет тестирование и модификацию программного обеспечения.
  - Рекомендуемое значение NOO <= 3 методов.
- Количество операций, добавленных подклассом - NOA (Number of Operations Added by a Subclass)
  - С ростом NOA подкласс удаляется от абстракции суперкласса.
  - Обычно при увеличении высоты иерархии классов (увеличении DIT) должно уменьшаться значение NOA на нижних уровнях иерархии.
  - Для рекомендуемых значений CS = 20 и DIT = 6 рекомендуемое значение NOA <= 4 методов (для класса-листа).

# Метрики Лоренца и Кидда

- Индекс специализации SI - (Specialization Index)

$$SI = (NOO \times уровень) / Мобщ,$$

  - где уровень — номер уровня в иерархии, на котором находится подкласс, Мобщ — общее количество методов класса.
  - Чем выше значение SI, тем больше вероятность того, что в иерархии классов есть классы, нарушающие абстракцию суперкласса.
  - Рекомендуемое значение SI <= 0,15.

# Метрики Лоренца и Кидда

- Средний размер операции OSAVG (Average Operation Size)
  - количество строк программы.
  - Альтернативный вариант — «количество сообщений, посланных операцией».
  - Рост значения означает, что обязанности размещены в классе не очень удачно.
  - Рекомендуемое значение OSAVG <= 9.
- Среднее количество параметров на операцию NPavg (Number of Parameters average)
  - Чем больше параметров у операции, тем сложнее сотрудничество между объектами. Поэтому значение NPavg должно быть как можно меньшим.
  - Рекомендуемое значение NPavg = 0,7.

# Метрики Лоренца и Кидда

- Сложность операции ОС (Operation Complexity)
  - LOC- или FP-оценок, метрики цикломатической сложности, метрики Холстеда.
  - желательно уменьшать ОС.
  - Рекомендуемое значение ОС <= 65

| Параметр | Вес |
|---|---|
| Вызовы функций API | 5,0 |
| Присваивания | 0,5 |
| Арифметические операции | 2,0 |
| Сообщения с параметрами | 3,0 |
| Вложенные выражения | 0,5 |
| Параметры | 0,3 |
| Простые вызовы | 7,0 |
| Временные переменные | 0,5 |
| Сообщения без параметров | 1,0 |

# Метрики Лоренца и Кидда

- Количество описаний сценариев NSS (Number of Scenario Scripts)
  - Рекомендуется — не менее одного сценария на публичный протокол системы, отражающий основные функциональные требования к подсистеме.
- Количество ключевых классов NKC (Number of Key Classes)
  - Ключевой класс прямо связан с проблемной областью.
  - 20-40% от общего количества классов. Оставшиеся классы реализуют общую инфраструктуру.
  - Рекомендуемое значение: если NKC < 0,2 от общего количества классов системы, следует углубить исследование проблемной области.
- Количество подсистем NSUB (NumberofSUBsystem)
  - размещение ресурсов, планирование (с акцентом на параллельную разработку), общие затраты на интеграцию.
  - Рекомендуемое значение: NSUB > 3.

# Проектные метрики Лоренца и Кидда

- NSS, NKC, NSUB формируют метрический базис фирмы, в который также включаются метрические значения по классами и операциям.
- Эти исторические данные могут использоваться для вычисления метрик производительности (среднее количество классов на разработчика или среднее количество методов на человеко-месяц).
- Совместное применение метрик позволяет оценивать затраты, продолжительность, персонал и другие характеристики текущего проекта.

# Метрики Фернандо Абреу

- MOOD – Metrics of OO Design
- Покрытие базовых механизмов объектно-ориентированной парадигмы, таких как инкапсуляция, наследование, полиморфизм, посылка сообщений;
- Формальное определение метрик, позволяющее избежать субъективности измерения;
- Независимость от размера оцениваемого программного продукта;
- Независимость от языка программирования, на котором написан оцениваемый продукт.

# Метрики Фернандо Абреу

- Фактор закрытости метода (MHF) Method Hiding Factor
  - Mv (Ci) — количество видимых методов в классе Ci (интерфейс класса);
  - Mh (Ci) — количество скрытых методов в классе Ci (реализация класса);
  - Md (Ci) = Mv (Ci) + Mh (Ci) — общее количество методов, определенных в классе C, (унаследованные методы не учитываются).
  - С увеличением MHF уменьшаются плотность дефектов в системе и затраты на их устранение.
  - Возрастание значения MHF и увеличение качества класса.

$$MHF = \frac{\sum_{i=1}^{TC} M_h(C_i)}{\sum_{i=1}^{TC} M_d(C_i)}$$

# Метрики Фернандо Абреу

- Фактор закрытости свойства (MHA) Attribute Hiding Factor
  - Av (Ci) — количество видимых свойств в классе Ci (интерфейс класса);
  - Ah(Ci) — количество скрытых свойств в классе Ci (реализация класса);
  - Ad(Ci) = Av (Ci) + Ah(Ci) — общее количество свойств, определенных в классе Ci (унаследованные свойства не учитываются)
  - В идеальном случае все свойства должны быть скрыты и доступны только для методов соответствующего класса
  - (AHF = 100%).

$$AHF = \frac{\sum_{i=1}^{TC} A_h(C_i)}{\sum_{i=1}^{TC} A_d(C_i)}$$

# Метрики Фернандо Абреу

- Фактор Наследования метода (MHF) Method Inheritance Factor
  - Mi (Ci ) — количество унаследованных и не переопределенных методов в классе Ci;
  - M0(Ci ) — количество унаследованных и переопределенных методов в классе Ci;
  - Mn(Ci ) — количество новых (не унаследованных и переопределенных) методов в классе Ci;
  - Md(Ci )= Mn(Ci ) + M0(Ci ) — количество методов, определенных в классе Ci;
  - Ma(Ci )= Md(Ci)+ Mi(Ci) — общее количество методов, доступных в классе Ci.
  - MIF = 0 указывает, что в системе нет эффективного наследования

$$MIF = \frac{\sum_{i=1}^{TC} M_i(C_i)}{\sum_{i=1}^{TC} M_a(C_i)}$$

# Метрики Фернандо Абреу

- Фактор Наследования свойства (AHF) Attribute Inheritance Factor
  - *Ai (Ci)* — количество унаследованных и не переопределенных свойств в классе *Ci*;
  - *A0(Ci)* — количество унаследованных и переопределенных свойств в классе *Ci*;
  - *An(Ci)* — количество новых (не унаследованных и переопределенных) свойств в классе *Ci*;
  - *Ad(Ci) = An(Ci) + A0(Ci)* — количество свойств, определенных в классе *Ci*;
  - *Aa(Ci) = Ad(Ci)+ Ai (Ci)* — общее количество свойств, доступных в классе *Ci*.

$$AIF = \frac{\sum_{i=1}^{TC} A_i(C_i)}{\sum_{i=1}^{TC} A_a(C_i)}$$

# Метрики Фернандо Абреу

- *фактор полиморфизма (POF) Polymorphism Factor*
  - *M0(Ci)* — количество унаследованных и переопределенных методов в классе *Ci*;
  - *Mn(Ci)* — количество новых (не унаследованных и переопределенных) методов в классе *Ci*;
  - *DC(Ci)* — количество потомков класса *Ci*;
  - *Md(Ci) = Mn(Ci) + M0(Ci)* — количество методов, определенных в классе *Ci*.
  - Умеренное использование полиморфизма уменьшает как плотность дефектов, так и затраты на доработку. Однако при POF > 10% возможен обратный эффект.

$$POF = \frac{\sum_{i=1}^{TC} M_0(C_i)}{\sum_{i=1}^{TC} [M_n(C_i) \times DC(C_i)]}$$

# Метрики Фернандо Абреу

- фактор сцепления (COF) Coupling Factor.
  - наличие между классами отношения «клиент-поставщик» (client-supplier).
  - класс-клиент содержит по меньшей мере одну не унаследованную ссылку на свойство или метод класса-поставщика.
  - Числитель COF фиксирует реальное количество сцеплений, не относящихся к наследованию.
  - С увеличением сцепления классов плотности дефектов и затрат на доработку также возрастают.
  - Сцепления отрицательно влияют на качество ПО.
  - сцепление увеличивает сложность, уменьшает инкапсуляцию и возможности повторного использования, затрудняет понимание и усложняет сопровождение ПО.

$$COF = \frac{\sum_{i=1}^{TC}[\sum_{j=1}^{TC} is\_client(C_i, C_j)]}{TC^2 - TC}$$