МЕТРИКИ АБРЕУ

Теоретические сведения

Свой подход к оценке качества объектно-ориентированного программирования разработал Фернандо Абреу (Fernando Brito e Abreu), предложив в 1994 г. набор метрик *MOOD* (Metrics for Object Oriented Design). Основными целями предложенного Абреу набора метрик являются следующие:

- учет базовых механизмов объектно-ориентированного подхода к созданию ПО (инкапсуляция, наследование, полиморфизм, посылка сообщений);
- формальное определение такого набора метрик, который позволит исключить или значительно снизить субъективность определения показателей качества ПС;
- независимость от размера оцениваемого программного продукта;
- независимость от языка программирования, который использовался при создании оцениваемого ПС.

Набор метрик Абреу включает в себя следующие показатели качества программного продукта:

- фактор закрытости метода (*MHF*);
- фактор закрытости свойства (*AHF*);
- фактор наследования метода (*MIF*);
- фактор наследования свойства (*AIF*);
- фактор полиморфизма (*POF*);
- фактор сцепления (*COF*).

Каждая метрика Абреу относится к одному из основных механизмов объектно-ориентированного подхода к программированию: инкапсуляции (метрики *MHF* и *AHF),* наследованию (метрики *MIF* и *A IF),* полиморфизму (*POF)* и отсылке сообщений (*COF*). В определениях набора *MOOD* не используются специфические конструкции языков программирования, что дает возможность исключить зависимость от языка программирования, применяемого при разработке программного средства.

Фактор закрытости метода $\boldsymbol{MHF}$ (Method Hiding Factor). Закрытость метода характеризует процентное количество классов, из которых конкретный метод невидим.

Для характеристики данной метрики вводятся следующие обозначения:

- $M_v(C_i)$ – интерфейс класса, определяющий количество видимых методов в классе C_i;
- $M_h(C_i)$ – реализация класса, характеризующая количество скрытых методов в классе C_i;
- $M_d(C_i) = M_v(C_i) + M_h(C_i)$ – общее количество методов, определенных в классе C_i, причем в общем количестве методов не учитываются унаследованные методы.

Тогда формула определения значения метрики MHF будет иметь следующий вид:

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

где TC – количество классов в создаваемой программной системе.

Если уровень видимости m-го метода i-го класса из j-го класса определить как

$$is\_visible(M_{mi}, C_j) = \begin{cases} 1, \text{ если } j \neq 1 \text{ и } C_j \text{ может вызвать } M_{mi} \\ 0 \text{ в других случаях} \end{cases},$$

а процентное количество классов, которые способны видеть m-й метод i-го класса, вычислить следующим образом:

$$V(M_{mi}) = \frac{\sum_{i=1}^{TC} is\_visible(M_{mi}, C_j)}{TC - 1},$$

то значение метрики закрытости метода MHF можно рассчитать с помощью следующего выражения:

$$MHF = \frac{\sum_{i=1}^{TC} \sum_{m=1}^{M_d(C_i)} (1 - V(M_{mi}))}{\sum_{i}^{TC} M_d(C_i)}.$$

Числитель последней формулы содержит сумму закрытости всех методов во всех классах программного продукта, а знаменатель этого выражения - общее количество методов, определенных в

рассматриваемой системе. С ростом значения метрики MHF уменьшается плотность дефектов в системе, что означает и сокращение затрат, необходимых для их устранения. Обычно разработка класса программы представляет собой пошаговый процесс, при котором к каждому классу постепенно добавляется все больше и больше деталей (которые означают скрытые методы). Такая схема разработки способствует росту как значения метрики MHF, так и качества класса.

Фактор закрытости свойства AHF (Attribute Hiding Factor). Закрытость свойства представляет собой процентное количество классов, из которых данное свойство невидимо. По аналогии с предыдущей метрикой вводятся следующие обозначения:

$A_v(C_i)$- интерфейс класса, означающий количество видимых свойств в классе C_i;

$A_h(C_i)$ - реализация класса, которая характеризует количество скрытых свойств в классе C_i;

$A_d(C_i) = A_v(C_i) + A_h(C_i)$ – общее количество свойств, определенных в классе C_i, причем унаследованные свойства не учитываются.

При таких обозначениях выражение для определения метрики *AHF* примет следующий вид:

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

где TC - количество классов в создаваемой программной системе.

Если видимость m-го свойства i-го класса из j-го класса вычислить в соответствии с выражением

$$is\_visible(A_{mi}, C_j) = \begin{cases} 1, \text{ если } j \neq 1 \text{ и } C_j \text{ может вызвать } A_{mi} \\ 0 \text{ в других случаях} \end{cases},$$

а процентное количество классов, которые видят m-в свойство i-го класса, определить следующим образом:

$$V(A_{mi}) = \frac{\sum_{i=1}^{TC} is\_visible(A_{mi}, C_j)}{TC - 1},$$

то выражение для определения значения метрики AHF можно записать так:

$$AHF = \frac{\sum_{i=1}^{TC} \sum_{m=1}^{A_d(C_i)} (1 - V(A_{mi}))}{\sum_{i=1}^{TC} A_d(C_i)}.$$

В числителе этой формулы AHF расположена сумма закрытости всех свойств во всех классах, а в знаменателе - общее количество свойств, определенных в анализируемой системе.

В идеальном случае все свойства должны быть скрыты и доступны только для методов соответствующего класса, т. е. наилучшим показателем является значение $AHF = 100\ \%$.

Фактор наследования метода MIF (Method Inheritance Factor). Для определения этой метрики Абреу вводит следующие обозначения:

$M_i(C_i)$ – количество унаследованных и не переопределенных методов в классе C_i;

$M_0(C_i)$ - количество унаследованных и переопределенных методов в классе C_i;

$M_n(C_i)$ - количество новых (не унаследованных и переопределенных) методов в классе C_i;

$M_d(C_i) = M_n(C_i) + M_0(C_i)$ – количество методов, определенных в классе C_i;

$M_a(C_i) = M_d(C_i) + M_i(C_i)$ – общее количество методов, доступных в классе C_i.

Тогда формула для определения метрики MIF будет такой:

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

Числителем MIF является сумма унаследованных (и непереопределенных) методов во всех классах рассматриваемой системы, в знаменателе – общее количество доступных методов (локально определенных и унаследованных) для всех классов. Нулевое значение метрики MIF говорит о том, что в анализируемой системе отсутствует эффективное наследование (например, все унаследованные методы переопределены). С ростом значения метрики MIF уменьшается плотность дефектов, следовательно, можно ожидать сокращения затрат на

исправление ошибок. Высокие значения метрики MIF (на уровне 70-80%) приводят к обратному эффекту, однако это не всегда однозначно, потому необходима дополнительная экспериментальная проверка. Можно сделать более мягкий вывод: умеренное использование наследования является весьма подходящим средством для снижения плотности дефектов и сокращения затрат на доработку программного средства.

Фактор наследования свойства AJF (Attribute Inheritance Factor). Для определения этой метрики вводятся следующие обозначения:

$A_i(C_i)$ - количество унаследованных и непереопределенных свойств в классе C_i;

$A_0(C_i)$ - количество унаследованных и переопределенных свойств в классе C_i;

$A_n(C_i)$ - количество новых (неунаследованных и переопределенных) свойств в классе C_i;

$A_d(C_i)= A_n(C_i)+A_0(C_i)$ - количество свойств, определенных в классе C_i

$A_a(C_i)= A_d(C_i)+A_i(C_i)$ - общее количество свойств, доступных в классе C_i.

Тогда формула для расчета метрики AIF примет следующий вид:

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

Числитель AIF образует сумма унаследованных (и непереопределенных) свойств во всех классах рассматриваемой системы, а знаменатель — общее количество доступных свойств (локально определенных и унаследованных) для всех классов.

Фактор полиморфизма POF (Polymorphism Factor) определяется с учетом следующих обозначений:

$M_0(C_i)$ - количество унаследованных и переопределенных методов в классе C_i;

$M_n(C_i)$ - количество новых (неунаследованных и переопределенных) методов в классе C_i;

$DC(C_i)$- количество потомков класса C_i;

$M_d(C_i) = M_n(C_i) = M_0(C_i)$ - количество методов, определенных в классе C_i.

Тогда формула метрики POF примет вид:

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

Числитель метрики POF указывает на реальное количество возможных полиморфных ситуаций. Очевидно, что сообщение, посланное в класс C_i статически или динамически связывается с реализацией именуемого метода. А этот метод, в свою очередь, может или быть представлен несколькими «формами», или переопределяться (в потомках C_i). Знаменатель метрики POF представляет максимальное количество возможных полиморфных ситуаций для класса C_i. Это происходит тогда, когда все новые методы, определенные в классе C_i, переопределяются во всех его потомках.

Умеренное использование полиморфизма уменьшает как плотность дефектов, так и затраты на доработку ПС. Однако при значениях метрики $POF > 10\ \%$ возможен обратный эффект, когда дефекты и затраты могут возрасти.

Фактор сцепления COF (Coupling Factor). В наборе метрик Абреу сцепление характеризует наличие между классами отношения «клиент-поставщик». Отношение «клиент-поставщик» ($C_c \to C_s$) здесь означает, что класс-клиент содержит по меньшей мере одну не унаследованную ссылку на свойство или метод класса-поставщика.

Если наличие отношения «клиент-поставщик» определять по формуле

$$is\_client(C_c, C_s) = \begin{cases} 1, \text{ если } C_c \to C_s \cap C_c \neq C_s, \\ 0 \text{ в других случаях} \end{cases}$$

то формула для вычисления метрики COF примет следующий вид:

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

Знаменатель COF соответствует максимально возможному количеству сцеплений в системе с TC-классами, поскольку каждый класс может быть потенциальным поставщиком для других классов. Из приведенного рассмотрения исключены рефлексивные отношения, когда класс является собственным поставщиком. Числитель COF характеризует реальное количество сцеплений, не относящихся к наследованию.

С увеличением сцепления классов плотность дефектов и уровень затрат на доработку программы также возрастают. Сцепления отрицательно влияют на качество программного средства, надо добиваться, чтобы их было как можно меньше. Практическое применение этой метрики еще раз доказывает, что сцепление увеличивает сложность ПО, уменьшает инкапсуляцию и возможность повторного использования программных компонентов, затрудняет понимание и усложняет сопровождение ПС.

Задача «Платеж за электроэнергию»

При решении задачи необходимо разработать исходный код программы, а также определить оценки характеристик программы на основе объектно-ориентированных метрик Абреу.

Реализация программы

Текст программы на языке C# для реализации возможного алгоритма решения поставленной задачи представлен на рис. 4.1.

Оценка характеристик программы

В соответствии с теорией Абреу необходимо определить значения следующих характеристик:

- фактора закрытости метода (MHF);
- фактора закрытости свойства (AHF);
- фактора наследования метода (MIF);
- фактора наследования свойства (AIF);
- фактора полиморфизма (POF);
- фактора сцепления (COF).

Каждая метрика Абреу относится к одному из основных механизмов объектно-ориентированного подхода к программированию: инкапсуляции (метрики MHF и AHF), наследованию (метрики MIF и AIF), полиморфизму (POF) и отсылке сообщений (COF). В определениях набора метрик Абреу не используются специфические конструкции языков программирования, что дает возможность исключить зависимость от языка программирования, применяемого при разработке программных средств.

Фактор закрытости метода MHF (Method Hiding Factor). Закрытость метода определяется количеством классов, из которых указанный метод

невидим. Значение данной метрики Абреу предлагает определять по соотношению (4.4.1).

В исходном коде решаемой задачи (см. рис. 4.1) определено три класса:

- class Электро;
- class Запрос;
- class Платежи;

Класс Электро в своем составе имеет четыре метода:

- public Электро (string фамилия);
- public Электро (string фамилия, int текущее, int предыдущее);
- public double Сумма();
- public string Инфо().

Все методы открыты, так как определены модификатором public. Следовательно, $M_h(\text{Электро}) = 0$, $M_d(\text{Электро}) = 4$.

Класс Запрос в своем составе имеет два метода:

- public static void Заполнить (Электро[ ] платеж);
- public static void Вывести (Электро[ ] платеж).

Все методы открыты, так как определены модификатором public. Следовательно, $M_h(\text{Запрос}) = 0$, $M_d(\text{Запрос}) = 2$.

Класс Платежи в своем составе имеет один метод Main(), который по умолчанию является открытым. Следовательно, $M_h(\text{Платежи}) = 0$, $M_d(\text{Платежи}) = 1$.

Таким образом, значение характеристики:

$$MHF = \frac{\sum_{i=1}^{TC} M_h(C_i)}{\sum_{i=1}^{TC} M_d(C_i)} = \frac{0+0+0}{4+2+1} = 0.$$

Решение задачи не имеет закрытых методов.

Фактор закрытости свойства AHF (Attribute Hiding Factor). Закрытость свойства (поля) представляет собой количество классов, из которых данное свойство (поле) невидимо.

В классе Электро определено четыре поля:

- private static double тариф = 1.84;
- private static int потреблениеСреднее = 300;
- private string фамилия;
- private int потребление.

Все перечисленные поля являются закрытыми, так как определены модификатором private. Следовательно, $A_h(\text{Электро}) = 4$, $A_d(\text{Электро}) = 4$,.

В классах Запрос и Платежи поля не определены, следовательно, для них $A_h = 0$ и $A_d = 0$. В этом случае

$$AHF = \frac{\sum_{i=1}^{TC} A_h(C_i)}{\sum_{i=1}^{TC} A_d(C_i)} = \frac{4+0+0}{4+0+0} = 1.$$

Закрытость полей всех классов программы составляет 100%.

Фактор наследования метода MIF (Method Inheritance Factor). В решении задачи принцип наследования не реализован. Следовательно, расчет метрики MIF для данной задачи не имеет смысла.

Фактор наследования свойства AIF (Attribute Inheritance Factor). В решении задачи принцип наследования не реализован. Следовательно, расчет метрики AIF не имеет смысла.

Фактор полиморфизма POF (Polymorphism Factor). В реализации данной задачи принцип наследования не реализован. Следовательно, расчет метрики POF не имеет смысла.

Фактор сцепления COF (Coupling Factor).

Для класса Электро возможными классами-поставщиками могут быть классы Запрос и Платежи. В классе Электро не содержится ни одной ссылки на свойства и методы этих классов, поэтому $is\_client(\text{Электро.Запрос}) = 0$; $is\_client(\text{Электро, Платежи}) = 0$.

Для класса Запрос возможными поставщиками могут быть Электро и Платежи. Класс Запрос содержит ссылки на два поля и три метода класса Электро. Ссылок на класс Платежи нет. Следовательно, is_client(Запрос, Электро)=1; is_client (Запрос, Платежи) = 0.

Для класса Платежи: is_client (Платежи, Электро) = 1; is client (Платежи, Запрос) = 1, так как в этом классе содержатся ссылки на поля и методы класса Электро и ссылки на методы класса Запрос.

Таким образом, получаем:

$$COF = \frac{\sum_{i=1}^{TC}\left[\sum_{j=1}^{TC} is\_client(C_i, C_j)\right]}{TC^2 - TC} = \frac{0+1+2}{3^2-3} = \frac{3}{6} = 0{,}5.$$

Фактор сцепления между классами невысок. Следовательно, возможная плотность дефектов ожидается низкой, уровень затрат на доработку программы будет небольшой.