Постоянно возрастающие в сложности реализации нужды общества в новых программных продуктах, очевидно, приводят к усложнению написания надёжного и легко сопровождаемого кода. Метрики качества программного обеспечения призваны облегчить вычисление объективных показателей для этих параметров.

<слайд 1>Метрика качества ПО, или просто метрика кода – это, можно сказать, некоторый протокол для измерения каких-то определённых значений кода программы, который позволяет разработчикам получить более глубокую картину «внутрянки» разрабатываемого ими ПО.</слайд 1>

Благодаря метрикам кода разработчики могут понять, какие типы/классы/методы должны быть переработаны, либо экстенсивно протестированы. Команда разработчиков может зафиксировать потенциальные риски, осознать текущее состояние разрабатываемого проекта и отслеживать прогресс разработки.

В этом докладе я коротко расскажу о метриках кода, которые могут использовать разработчики, выбравшие в качестве своей среды разработки Microsoft Visual Studio.

Перед тем, как вдаваться в определения и суть этих метрик, разберёмся с терминами, которые будут использованы в процессе.

<слайд 2>Объём Холстеда – число, высчитываемое по формуле , где – длина программы, равная количеству всех операторов и операндов в программе, а – словарь программы, равный количеству уникальных операторов и операндов в программе. </слайд 2>

<слайд 3.1>Цикломатическая сложность</слайд 3.1>

Для того, чтобы дать определение термину «Цикломатическая сложность», нужно вспомнить про графы. Любой код можно визуализировать с помощью графа, если предположить, что любой оператор – это узел, а дуги, из него выходящие, символизируют возможные пути, по которым может пойти программа после выполнения этого оператора. Тогда, например, обычный оператор присваивания будет иметь один выход, оператор проверки условия – два выхода, а оператор выбора (также известный как switch) будет иметь столько выходов, сколько различных вариантов обработки (операторов case) им предусмотрено. Цикломатическая сложность <слайд 3.2>высчитывается по формуле , где – количество дуг в таком графе, – количество узлов, – количество компонент связности</слайд 3.2>.

<слайд 4>LOC – Lines of code – количество строк кода в программе.</слайд 4>

С терминами разобрались, перейдём непосредственно к метрикам.

<слайд 5>Метрика 1. Цикломатическая сложность</слайд 5>

Рассмотрим пример простого фрагмента кода, для которого нужно вычислить цикломатическую сложность.

<слайд 6.1>\*скриншот кода 7\*</слайд 6.1>

Граф для этого кода будет выглядеть вот так:

<слайд 6.2>\*скриншот кода 7 + граф 7\*</слайд 6.2>

Пользуемся формулой, о которой говорили ранее. В этом графе 5 дуг, 5 узлов и одна компонента связности. Тогда цикломатическая сложность этого фрагмента кода будет равна 5 – 5 + 2 = 2.

<слайд 7>Метрика 2. Индекс сопровождаемости: </слайд 7>

Раньше для расчёта этой метрики использовалась только та формула, которая выступает вторым аргументом у максимума, без умножения на . Однако её шкала была, скажем так, нестандартной – от 171 баллов до минус бесконечности. Поэтому приняли решение все значения меньше 0 сделать эквивалентными 0, а промежуток от 0 до 171 баллов промасштабировать на промежуток от 0 до 100.

Примера ради вычислим индекс сопровождаемости для того же самого фрагмента кода. Для этого нам нужно знать объём Холстеда, цикломатическую сложность и количество строк этого кода. <слайд 8.1>Цикломатическую сложность уже знаем.</слайд 8.1.> Приложив титанические усилия и выполнив математические вычисления невиданной ранее сложности, видим, что <слайд 8.2>количество строк в этом коде – 6.</слайд 8.2> Осталось высчитать объём Холстеда. Для этого фрагмента кода , = 12. Тогда имеем <слайд 8.3>.</слайд 8.3>3. Высчитывая значение получим примерно 69.6. Вновь проводя математические вычисления просто неприличной сложности, видим, что . Следовательно, индекс сопровождаемости нашего шестистрочного кода – 69.6.

<слайд 9>Метрика 3. Глубина наследования – длина максимально большой цепочки наследующихся классов.</слайд 9>

Здесь всё просто, в общем-то, объяснять даже нечего. Для простоты представления можно визуализировать иерархию классов в программе, условно представив её в виде графа, узлы которого – это классы. Например, для вот такого фрагмента кода: <слайд 10.1>\*скриншот кода 10\*</слайд 10.1>, такой граф будет выглядеть примерно так: <слайд 10.2>\*скриншот кода 10 + граф 10\*</слайд 10.2>. По этому графу можно быстро сделать вывод, что максимально большая цепочка из классов будет иметь длину 6.

<слайд 11>Метрика 4. Связь классов – максимальное количество различных используемых классов произвольной функцией или другим классом.</слайд 11>

Для демонстрации примера применения этой метрики я по мотивам знаменитого художественного фильма воспользовался графическими материалами из документации Microsoft.

Пример первый: просто описание класса. <слайд 12>\*скриншот из документации 1\*</слайд 12>. По определению нетрудно видеть, что значение этой метрики для этого класса будет равно 0, ведь он не использует никаких внешних классов.

Пример второй: описание класса, использующее в себе первый описанный класс. <слайд 13>\*скриншот из документации 2\*</слайд 13>. Для этого класса значение этой метрики уже будет давать 1.

И третий пример: переопределение метода класса на такой, который помимо первого описанного использует ещё и некий третий класс. <слайд 14>\*скриншот из документации 3\*</слайд 14>. В целом в этом методе используется два различных внешних класса, поэтому для него значение этой метрики будет равно 2.

<слайд 15>Метрика 5. Количество строк исходного кода – подсчитывает все строки исходного кода, в том числе пустые строки.</слайд 15>

Для нашего примера уже посчитали ранее – 6. И, наконец,

<слайд 16>Метрика 6. Количество строк исполняемого кода – подсчитывает приблизительное количество строк кода, которые будут исполнены во время программы.</слайд 16>

Для нашего кода значение этой метрики должно быть 5, ведь строчек кода хоть и 6, но исполнено будет только 5, т.к. строчки if(…) … и else … взаимоисключающие и не могут обе быть выполнены за одно выполнение программы.

В завершение хотел бы сказать, что метрик кода существует больше, чем было описано в этом докладе, однако в Visual Studio доступны именно эти, и их, в общем-то, полагаю, вполне достаточно для среднестатистического разработчика. На этом, собственно, всё.

|  |  |
| --- | --- |
| скриншот кода 6 |  |
| граф 6 |  |
| скриншот кода 10 |  |
| граф 10 |  |
| скриншот из документации 1 | C:\Users\konst\Downloads\class-coupling-example-1.png |
| скриншот из документации 2 | C:\Users\konst\Downloads\class-coupling-example-3.png |
| скриншот из документации 3 | C:\Users\konst\Downloads\class-coupling-example-4.png |