электронный журнал
МОЛОДЕЖНЫЙ НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51038.

 # 07, июль 2017

УДК 004.42

Анализ количественных показателей добротности программного кода

Лебедев В.И., *студент*
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Компьютерные системы и сети»

Хромушкин Н.А., *студент*
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Компьютерные системы и сети»

Научный руководитель: Березкин Д.В., к.т.н., доцент
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Компьютерные системы и сети»
v.suzev@bmstu.ru

Само понятие добротности ПО было введено Поттосиным И.В. с целью оценки реализации с технической стороны. Добротность программы заключается в том, что программа разумно и рационально организована, с достаточно продуманной организацией потоков управления и информационных потоков, не слишком переусложнена. Таким образом, подо добротностью программного продукта, подразумевается некая общая характеристика качества технической реализации ПО. Данная характеристика включает в себя различные требования к программному продукту в зависимости от цели разработки, предметной области, требований пользователя [1].

Поттосин И.В. вводит четыре класса критериев добротности программ: количественные, структурные, генетические и прагматические критерии. Эти группы критериев существенно различаются по оцениваемым показателям, методам оценки, используемым механизмам, а также целям и задачам оценки. В этой связи разумно проводить оценку данных критериев раздельно, не пытаясь одним средством оценить, например, качество структурных решений программы и безошибочность написания кода.

При том, что характеристика добротности есть результат внешней оценки программного продукта, следует отметить необходимость контроля и оценки данного характеристики в процессе разработки ПО. Данное утверждение следует из того факта,

что стоимость и трудоемкость исправление или корректировка решений, отрицательным образом влияющих на добротность итогового продукта, в случае их обнаружения на различных стадиях разработки ПО, могут различаться весьма существенно. Так, на рисунке №1 приведена таблица зависимости затрат на устранение дефекта программного кода в зависимости от этапа разработки, на котором данный дефект был обнаружен.

	Время обнаружения дефекта				
Время внесения дефекта	Выработка требований	Проектирование архитектуры	Конструирование (кодирование)	Тестирование	После выпуска ПО
Выработка требований	1	3	5-10	10	10-100
Проектирование архитектуры	-	1	10	15	25-100
Конструирование (кодирование)	-	-	1	10	10-25

Рис. 1. Средняя стоимость исправления дефектов в зависимости от времени их внесения и обнаружения [2]

Для выполнения задачи анализа добротности необходимо рассмотреть критерии, по которым оцениваются программные продукты. Как уже было сказано, данные критерии могут варьироваться, поэтому укажем наиболее общие: защищенность, надежность, устойчивость, понимаемость, тестируемость, адаптируемость, модульность, сложность, переносимость, доступность пользователю, повторная используемость, эффективность, изучаемость и сопровождаемость [1]. Рассмотрим подробнее оцениваемые характеристики с целью определениях требований к ним, методов оценки, а также возможности проверки данной характеристики при помощи автоматического анализатора без участия человека.

Рассмотрим подробнее группу количественных критериев как наиболее полно поддающуюся автоматизированной оценке.

Количественные критерии оценки представляют собой некие метрики, через которые выражается сложность программного продукта. Это могут быть различные количественные показатели, например, метрики Холстеда, метрики цикломатической сложности Мак-Кейба, метрики Хура и другие. При этом для ответа на вопрос, соответствует ли соответствующий показатель критериям положительной оценки добротности, производится сравнение показателей с неким показателем-образцом. Этим образцом могут быть требования заказчика или оценка соответствующих параметров некой схожей программы, или же принятые стандарты в данной области. В этой связи

необходимо помнить, что оценка количественных критериев относительна, т. е. базируется на сравнении метрик программы с образцом или аналогом.

Существующие наборы метрик можно разделить на несколько типов: метрики размера программ; метрики сложности потока управления программ; метрики сложности потока данных программ [3].

Метрики первой группы базируются на определении количественных характеристик, связанных с размером программы, и отличаются относительной простотой. К наиболее известным метрикам данной группы относятся число операторов программы, количество строк исходного текста и метрики Холстеда [4]. Метрики этой группы ориентированы на анализ исходного текста программ. Поэтому они могут использоваться для оценки сложности промежуточных продуктов разработки. При этом, объектом рассмотрения и подсчета данных метрик является размер, или же в случае более совершенных метрик данного типа некий показатель функциональности программы. Простых количественно-ориентированные метрики появились раньше остальных, в частности, первая подобная метрика SLOC, считающая просто количество физических и логических строк кода. Отрицательной стороной подобных метрик, особенно в простом и примитивном варианте является их плохая универсальность, неспособность объективно оценивать затраты на написание кода. В частности, размерности программ могут зависеть от таких факторов как стиль написания, используемый язык и т. д. Так, непродуманные попытки использования подобных метрик для оценки программных могут приводить в т. ч. к феномену «индусского кода» т. е. ситуации, когда избыточная, нерационально написанная программа рассматривается заказчиком как эталон [5].

Метрики второй группы базируются на анализе управляющего графа программы. Представителем данной группы является метрика Маккейба [4]. Управляющий граф программы, который используют метрики данной группы, может быть построен на основе алгоритмов модулей. Поэтому метрики второй группы могут применяться для оценки сложности промежуточных продуктов разработки. Показатель цикломатической сложности позволяет не только произвести оценку трудоемкости реализации отдельных элементов программного проекта и скорректировать общие показатели оценки длительности и стоимости проекта, но и оценить связанные риски и принять необходимые управленческие решения. Как правило, при вычислении цикломатической сложности логические операторы не учитываются. Показатель цикломатической сложности может быть рассчитан отдельно для модуля, метода и других структурных единиц программы. При этом в зависимости от целей оценивания и текущего этапа разработки программы, в

подсчет цикломатической сложности могут вноситься изменения, так подсчет может быть: строгим (цикломатическая сложность включает логические операторы), модифицированным (логические операторы рассматриваются в целом, а не как набор разных ветвлений) и упрощенным (вычисление сводится к подсчету управляющих операторов без построения графа и работы с ним).

Метрики третьей группы базируются на оценке использования, конфигурации и размещения данных в программе. В первую очередь это касается глобальных переменных. К данной группе относятся метрики Чепина.

В приведенной таблице были приведены наиболее известные и используемые метрики кода [6].

Название модели	Формула нахождения	Пояснения
Количество строк кода (SLOC) - физические строки - логические строки - строки комментариев		Физической строкой считается каждая непустая строка текстового файла. Логические строки считаются по количеству операторов в программе.
Метрики Холстеда - длина программы - объем программы - оценка реализации - трудность понимания - трудоемкость кодирования - уровень языка выражения - информационное содержание - оптимальная модульность	$N = n_1 \log_1 n_1 + n_2 \log_2 n_2$ $V = N \log_2 n$ $L^* = (2 n_2)/ (n_1 N_2)$ $E_c = V/ L^*$ $D = (n_1 N_2) (2n_2) = 1/ L^*$ $\lambda^* = V/ D^2 = V/ L^{*2}$ $I = V / D$ $M = n_2^* /6$	n1 - количество различных операторов программы; n2 - количество различных операндов программы; N1 - общее количество операторов программы; N2 - общее количество операндов программы.
Метрики Мак-Кейба - цикломатическое число - цикломатическая сложность	$\lambda (G) = m - n + p$ $\nu (G) = \lambda (G) +1 = m - n + 2$	e – количество дуг n – количество вершин p – число компонентов связности

Таблица (Продолжение)

Метрика Чепина - мера трудности понимания программ на основе входных и выходных данных	H = 0,5T+P+2M+3C	P — вводимые переменные для расчетов и для обеспечения вывода M — модифицируемые, или создаваемые внутри программы переменные C — переменные, участвующие в управлении работой программного модуля (управляющие переменные) T — не используемые в программе («паразитные») переменные
Метрики Джилба - количество операторов цикла - количество операторов условия - число модулей или подсистем - отношение числа связей между модулями к числу модулей - отношение числа ненормальных выходов из множества операторов к общему числу операторов	L_{1oop} L_{IF} L_{mod} $f = N^4_{SV} / L_{mod}$ $f^* = N^*_{SV} / L$	
Метрика Шнадевида - число путей в управляющем графе	S = □□Pi Ci	
Метрика Хансена - пара (цикломатическое число, число операторов)	H = {□ , N}	
Метрика Янгера -логическая сложность с учетом истории вычислений -сложность проектирования -насыщенность комментариями -число внешних обращений -число операторов	□ (□) Cc = □□log2 (i + 1) [□□n Cxy (n)] X = K/C Ci L1	

Таблица (Продолжение)

Группа метрик надежности программы -количество структурных изменений, произведенных с момента прошлой проверки - количество ошибок, выявленных в ходе просмотра кода, - количество ошибок, выявленных при тестировании программы и -количество необходимых структурных изменений, необходимых для корректной работы программы		В большинстве случаев, значение каждой метрики вычисляется как среднее на 1000 строк программного кода.
Метрика Пивоварского	N(G) = v*(G) +СУММАPi	v*(G) — модифицированная цикломатическая сложность, при вычислении которой оператор CASE с n выходами рассматривается как один логический оператор Pi — глубина вложенности i-й предикатной вершины
Метрика граничных значений	S0=1-(v-1)/Sa	S0 — относительная граничная сложность программы Sa — абсолютная граничная сложность программы v — общее число вершин графа потока управления
Метрика Шнейдевинда		Число возможных путей в графе потока управлениz
Метрика спена		Спен(span) — это число обращений к переменной между её первым и последним появлением в программе (переменная, встретившаяся n раз, имеет спен равный n-1).

Таблица (Продолжение)

Мера Овиедо	C = aCF + bDF	CF – сложность потока управления, учитывающая только число дуг графа DF – сложность потока данных, вычисляемая как сумма сложностей потоков данных базовых блоков программы, причем сложность потока данных блока есть число переменных, которые используются, но не определяются в блоке a, b – весовые коэффициенты, которые могут быть приняты равными 1

Отдельно следует упомянуть группу обобщающих метрик, которые позволяют получить некий обобщенный показатель нескольких.

С целью нахождения общего итогового количественного значения качества может быть использована метрика Кокола, определяющая общий показатель нескольких метрик разного приоритета.

$$H\_M = (M + R1 * M(M1) + \ldots + Rn * M(Mn)/(1 + R1 + \ldots + Rn),$$

где M – главная метрика,

Mi – прочие значимые метрики,

Ri – коэфициенты каждой метрики.

Другой обобщающей Метрикой является метрика Зольновского, Симмонса и Тайера. Данная метрика представляет собой взвешенную сумма наборов индикаторов: (структура, взаимодействие, объем, данные) и (сложность интерфейса, вычислительная сложность, сложность ввода/вывода, читабельность). При этом как и в случае с метрикой Кокола, в качестве значений наборов используются показатели других метрик соответствующего типа.

Видно, что каждая метрика (за исключением обобщающих метрик) характеризует определенную значимую характеристику программного кода и при условии использования достаточного их набора можно осуществить достаточно полное описание программы. При этом данный тип критериев оценки качества ПО относительно легко может быть реализован с программно и проверяться без или с минимальным участием человека-

эксперта. Это связано с тем, что сложные с точки зрения программной реализации решения, например, эвристические, для подобных оценок малоиспользуемы, в то время как вычисления и обработка текста программы на вхождения элементов являются сравнительно простой (малозатратной) операцией ЭВМ.

Весьма разумным решением для автоматизации оценки программы с позиции вычисления наборов метрик является использование для этой цели статических анализаторов программного кода. Подсчет метрик кода является одной из традиционно основных задач данного класс ПО.

Статический анализатор обрабатывает исходные тексты программ и выдает программисту рекомендации обратить повышенное внимание на определенные участки кода, а в нашем случае, занимается подсчетом метрик, обнаруживая их по определенным признакам в коде. Конечно, программа может нуждаться в контроле со стороны программиста, причем необходимость этого контроля зависит от надежности обнаружения метрик и количества потенциально возможного «брака», однако соотношение польза/цена делает использование статического анализа весьма полезной практикой, причем не стопроцентная надежность программы в целом компенсируется отсутствием «человеческого фактора» в виде возможной ошибке в подсчете из-за невнимательности, ошибки или другого внешнего фактора. Как и при использовании статического анализатора при обзоре кода на предмет ошибок, данный метод позволяет не заменить полностью работу программиста, но существенно повысить КПД подобного его работы.

К программе оценки количественных показателей добротности необходимо выдвинуть следующие требования (помимо возможности подсчитывать необходимые метрики по колу программы при помощи механизма статического анализа) [7]: возможность настройки искомых метрик (как отдельно каждой метрики, так и в рамках наборов метрик, например метрики Холстеда), подсчет суммарного показателя качества при помощи метрики Кокола или метрик Зольновского, Симмонса и Тайера, возможность задавать приоритет каждой метрики или набора метрик для вычисления метрики Кокола, возможность пользователя видеть результаты каждого подсчета и возможность корректировать его. Последнее требование необходимо с целью возможность контроля работы программы со стороны пользователя, что позволяет дополнительно снизить вероятность ошибки в подсчетах.

Следует заметить, что использование метрик кода для оценки результативности и эффективности работы сотрудников в ряде компаний вызывали критику, основные тезисы

которой стоит привести и разобрать. Основные аргументы скептически настроенных по отношению к метрикам специалистов можно разделить на следующие категории:

1. Этически аргументы — мнения о неэтичности подобного подхода, неучете личных качеств работника, опыта и т. д. Ввиду различных мнений по данному вопросу, а также тому, что вопрос человеческой этики явно выходит за рамки данной статьи, оставим данную группу аргументов без комментариев.

2. Аргументы о возможности искажения — мнение, что оценка кода программы может быть умышленно искажена сотрудниками, знающими, что их труд оценивается тем или иным набором метрик. Предполагается, что сотрудники могут оптимизировать свою работу под ту или иную метрику для увеличения показателя, что никак не означает улучшения качества кода. Примером такой оптимизации в ущерб результату следует ставший широко известным «индусский код» т. е. намеренное раздувание программы и использование неэффективных решений с целью увеличения количества строк кода.

Данный аргумент не лишен смысла, так как практически любая метрика или набор метрик могут быть повышены в ущерб прочим показателям программы при условии, что работник знает критерии оценки. Тем не менее, разные наборы метрик уязвимы в разной степени — простые метрики, например, SLOC сравнительно беззащитны, а потому их использование без каких-то других способов оценки весьма спорно и ненадежно.

Решением указанных проблем будет использование комплексных наборов метрик, а также комплексных оценок качества программы. При этом необходимо следить за сбалансированностью ценности каждой из метрик таким образом, чтобы улучшение одной из них путем существенного ухудшения прочих не давало необъективного повышения итоговой оценки.

Резюмируя вышеизложенное можно сказать, что использование набора разносторонних метрик, представляющих собой оценку различных аспектов качества программного кода, можно с высокой точностью оценивать количественные критерии добротности программного обеспечения. Данная система может быть использована как часть более крупной системы комплексной оценки добротности программного обеспечения, так и представляет самостоятельную ценность в рамках программной разработки, так как позволяет оценивать качество программы, а в ряде случаев и качество труда программиста. При этом, важным условием работоспособности данного программного продукта являются верно подобранные коэффициенты значимости той или иной метрики программы. Такие коэффициенты должны отражать наиболее актуальные

сведения о возможностях и требованиях к программному продукту, а следовательно скорее всего быть корректируемыми пользователем и/или программой в рамках актуализации данных. Весьма перспективным выглядит использование в данном аспекте технологии построения нейронной сети, обучающейся по методу «с учителем».

Список литературы

[1]. Поттосин И.В. Добротность программ и информационных потоков // Открытые системы. 1998. № 6. С. 24-27.

[2]. Макконнелл С. Совершенный код: пер. с англ. СПб.: Питер, 2007, 252 с. [McConnell S.C. Code Complete. Second edition. Microsoft Pres, 2004, 251 p.].

[3]. Изосимов А.В., Рыжко А.Л. Метрическая оценка качества программ / под ред. А.Л. Рыжко. М.: МАИ, 1989. 214 с.

[4]. Рыжков Е.К., Программный код и его метрики. 2014. Режим доступа: http://habrahabr.ru/company/intel/blog/106082/ (дата обращения: 10.12.2016).

[5]. Богданов Д. В. Стандартизация жизненного цикла программных средств. СПб.: Питер, 2006. 140 с.

[6]. Милютин А.Н. Метрики кода программного обеспечения. 2015. Режим доступа: http://www.viva64.com/ru/a (дата обращения 11.12.2016).

[7]. Андреев А.М., Березкин Д.В., Можаров Г.С., Свирин И.С. Математическое моделирование надежности компьютерных систем и сетей // Вестник МГТУ. Сер. Приборостроение. Спец. вып. «Моделирование и идентификация компьютерных систем и сетей». 2012. С. 3–46.