УДК 681.3.06+681.327

ПРОБЛЕМЫ ИЗМЕРЕНИЯ КАЧЕСТВА ПРОГРАММНОГО КОДА

С.В. *Звездин*

PROBLEMS OF MEASUREMENT OF CODE QUALITY

S.V. *Zwezdin*

Контроль качества кода - важная задача программной инженерии. Однако невозможно контролировать качество без числовых показателей. В этой статье предлагается математическая модель для определения показателей качества программного кода.

Ключевые слова: метрики, качество кода, измерение.

Code quality assurance is important problem of software engineering. However, it is impossible to control quality without numerical indicators. In this article the mathematical model for calculation of indicators of quality of a program code is offered.

Keywords: metrics, code quality, measurement.

Введение

Отрасль информационных технологий является одной из наиболее интенсивно развивающихся областей в современной жизни. Быт людей в настоящее время устроен таким образом, что даже далекий от информационных технологий человек ежедневно использует их для достижения целей. Более того, компьютерные программы на сегодняшний день используются даже в тех областях, где цена ошибки очень высока. Безусловно, использование автоматизированных систем является очень удобной заменой повторяющихся операций, выполняемых человеком. Это позволяет увеличить эффективность работы и избежать случайных ошибок в процессе работы. Однако, заменяя ручной труд человека программным кодом, появляется риск возникновения ошибки в программе. Дефекты в программном коде могу иметь разную природу и последствия. Одни ошибки возникают по причине низкого качества программного кода и могут доставить пользователю определенные неудобства, другие - иметь случайный характер. Однако иногда ценой ошибки может быть человеческая жизнь. Как правило, подобный род ошибок фиксируется на этапе разработки и эксплуатации и зачастую постепенно исключается из рабочего кода. Тем не менее, существуют такие дефекты системы, которые проявляются не сразу, а в процессе поддержки и развития программной системы. Постепенно ошибки, которые казались незначительными, могут становиться серьезными дефектами, которые мешают дальнейшему развитию продукта. Обычно такие дефекты связаны с архитектурными изъянами и некорректным построением программного кода. Ошибки подобного рода несут в себе серьезную проблему, которая проявляется только после длительного срока работы над программной системой. Из-за этой проблемы могут прекращаться работы над, казалось бы, успешными проектами. В лучшем случае программный продукт может быть разработан заново, без использования кода, содержащего подобного рода дефекты.

Мировое сообщество серьезно озабочено возможностью появления различного рода дефектов в программном коде. Подтверждением тому может быть факт создания международных стандартов на разработку программных систем (ISO/IEC, CMM/CMMI, др.). Более того, за последние несколько десятилетий было создано большое количество различных методологий и подходов к разработке различных типов программных систем. Каждая такая методология стремится к получению работоспособного продукта после завершения его разработки. Некоторые из методологий делают акцент на последовательность процессов разработки, некоторые - делают процесс максимально простым и гибким. Тем не менее, каждая из таких методологий стремится к созданию качественного продукта.

Однако как известно, мы не можем контролировать то, что не можем измерить. Это же утверждение можно отнести и качеству кода. Трудно принимать какие-либо проектные решения при отсутствии количественных измерений характеристик. Поэтому проблема измерения качества кода на сегодняшний день является актуальной.

Звездин Сергей Владимирович - ассистент кафедры информатики факультета экономики и управления ЮУрГУ; sergey@zwezdin.com

Zwezdin Sergey Vladimirovich - assistant of Informatics department of Economics and management faculty of SUSU; sergey@zwezdin.com

1. Качество программного обеспечения и качество кода

Понятие «качество» является сложным и многогранным. Обычно под качеством понимается соответствие объекта каким-то предъявляемым требованиям [1]. В программной инженерии этот термин можно трактовать по-разному. Можно трактовать качество программного продукта как соответствие его характеристикам, определенных в требованиях к продукту Это означает, что конечный продукт решает поставленные перед ним задачи. Другая интерпретация может состоять в том, что качество продукта - это отсутствие дефектов и ошибок, связанных с аварийной работой продукта. Также можно трактовать понятие качества как способность к легкому изменению программного кода и возможность легкого добавления дополнительных функциональных возможностей к продукту Как видно термин «качество» может использоваться на различных уровнях и использоваться в разных контекстах. Поэтому необходимо различать два главных понятия - «качество программного продукта» и «качество программного кода».

Качество программного продукта определяется тем, насколько он решает задачи конечных пользователей. Серьезное влияние на качество продукта оказывает корректность поставленных требований к продукту на этапе анализа. Если аналитики перед началом работ не смогли корректно определить задачи и проблемы пользователей, то такой программный продукт вряд ли сможет помочь пользователям.

Качество программного кода подразумевает грамотно выдержанный архитектурный стиль программного кода, четкое разделение кода на функциональные блоки и строгая структуризация и т. д. Таким образом, качественный программный код - это код, который легко поддерживать, вносить в него дополнительную функциональность, изменять существующие алгоритмы и т. д. Для получения качественного программного кода на сегодняшний день разработчики пользуются различными методологиями и практиками, применяют шаблоны проектирования, для типовых задач используют готовые библиотеки и алгоритмы.

Таким образом, видна существенная разница между понятиями «качество продукта» и «качество кода». Более того, продукт может иметь качественный код и при этом не решать проблем конечных пользователь. Такой программный продукт считается некачественным. И наоборот, качественный продукт, который успешно справляется с задачами пользователей, может состоять из некачественного кода. Естественно, цель каждой разрабатываемой программной системы - решить задачи пользователя. Зачем же тогда бороться за качество кода? Ответ простой - практически каждая программная система имеет тенденции к дальнейшему развитию и модификации. При этом важными показателями являются количество дефектов в программе и стоимость модификации кода. Если при добавлении новой функциональности возникает неприемлемое число ошибок, то продукт уже не может удовлетворять потребности пользователя. Аналогично, если стоимость добавления новой функциональности в программный продукт слишком высока, то это также негативно сказывается на пользователе. Из этого следует вывод, что качество программного кода - это нефункциональный, но очень важный показатель, который впоследствии оказывает опосредованное влияние на качество конечного программного продукта.

Поскольку качество кода оказывает существенное влияние на качество самого программного продукта, то в данной работе будут рассматриваться проблемы анализа качества программного кода.

2. Метрики **программного** кода

Исходя из приведенных ранее доводов, встает задача контроля качества программного кода и удержания его на необходимом уровне. Однако для того, чтобы контролировать качественные показатели, необходимо иметь численные характеристики этих показателей. Другими словами, для контроля качества программного кода необходимо уметь измерять это качество.

Для получения формальных оценок качества программного обеспечения существуют метрики программного кода. Суть этого механизма заключается в том, что на основе анализа исходного кода программной системы можно получить различные числовые характеристики. Обычно вычисление таких показателей строится на основе анализа графа программного потока или структуры программного кода [2]. На сегодняшний день существует большое количество метрик кода, анализирующих различные аспекты программного кода. К наиболее типичным метрикам можно отнести показатели количества строк кода в системе, сложности графа программы, глубины наследования классов и т. д. Преимущество метрик программного кода состоит в том, что в процессе их вычисления не участвует человеческий фактор, а все вычисления производит компьютер. Это гарантирует факт точности и повторяемости таких измерений для каждой из метрик. Более того, метрики можно вычислять автоматизированно и строить на их основе различные аналитические отчеты. Однако, на сегодняшний день метрики сильно дискредитировали себя в глазах большого количества разработчиков по всему миру. Это произошло из-за того, что каждый разработчик использует сильно ограниченный, не связанный набор метрик, и таким образом, получает однобокое представление о системе.

Для построения объективного представления о программном коде можно использовать связанный набор метрик, которые будут отражать цело-

стное представление о качестве программного кода. Как правило, качественный код представляется как программный код, не наделенный избыточной сложностью и связностью частей системы, хорошо структурированный и имеющий адекватные пропорции для объема (функциональные блоки не должны быть слишком большими или маленькими, что отражает качество декомпозиции в системе). Для того чтобы проанализировать качество программного кода в этой плоскости, предлагается использовать связанную систему метрик, характеризующую четыре этих аспекта качества программного кода. Таким образом, требуется измерить:

- сложность программного кода;
- связность программного кода;
- структурированность программного кода;
- объемные характеристики программного кода.

Для получения объективной информации в каждой группе необходимо использовать набор метрик. После вычисления каждой метрики в группе необходимо объединить эти показатели и получить единый результат для каждой группы. Для этого могут быть использованы различные математические методы и подходы. Одним из наиболее эффективных подходов является комплексная мера П. Кокола. Эта мера берет за основу одну метрику, а также учитывает другие метрики, которые должны оказывать влияние на конечный результат. Общий вид меры Кокола представлен формулой (1).

Мера Кокола содержит базовую метрику M и другие важные метрики M_i. Значения R_i (корректирующие коэффициенты) и $M(M_i)$ (корректирующие функции) рассчитываются методом регрессивного анализа. Таким образом, мера Кокола позволяет получить единственное числовое значение для набора метрик с учетом взвешенных коэффициентов.

Первая группа измерений - это **сложность программного кода.** Эти измерения должны показать насколько сложным является измеряемый код. Поскольку при разработке программного кода программисты очень часто занимаются борьбой со сложностью, то эта группа будет одной из важнейших из представленных выше. Для измерения сложности обычно применяются метрики сложности Мак-Кейба и его модификации. Кроме того, для этого могут использоваться меры Холстеда, отражающие трудоемкость понимания программы и трудоемкость кодирования [3]. Таким образом, можно получить связанную систему метрик для вычисления сложности программного кода, которая представлена формулой (2).

В качество базовой меры будет использоваться цикломатическое число Мак-Кейба. Таким образом, комплексная мера сложности программы высчитывается по формуле (3).

Следующая группа измерений - **связность программного кода.** Она отражает степень зависимости компонентов программного кода друг от друга [4]. Как известно, чем больше таких зависимостей, тем больше сложностей может появлятся при модификации и развитии программной системы. Основными метриками в этой группе являются метрики внешних и внутрених зависимостей (efferent coupling и afferent coupling), глубина наследования и среднее число внутренних связей на тип (relational cohesion). Таким образом, данная группа измерений представляется системой (4).

$$H = \frac{M + R_1 \times M(M_1) + ... + R_n \times M(M_n)}{1 + R_1 + ... + R_n} \tag{1}$$

$$\begin{cases} \lambda(G) = m - n + 2 \text{ (цикломатическое число Мак-Кейба)}, \\ H = 0,5T + P + 2M + 3C \text{ (мера Чепена)}, \\ S = \sum P_i C_i \text{ (мера Шнадевида)}, \\ T = [\upsilon_1 \div \upsilon_2] \text{ (интервальная мера Майерса)}, \\ E_c = V/L^* \text{ (мера Холстеда – трудоемкость понимания)}, \\ D = 1/L^* \text{ (мера Холстеда – трудоемкость кодирования)}. \end{cases} \tag{2}$$

$$H_{compl} = \frac{\lambda(G) + R_1 \times M(H) + R_2 \times M(S) + R_3 \times M(T) + R_4 \times M(E_c) + R_5 \times M(D)}{1 + R_1 + R_2 + R_3 + R_4 + R_5}. \tag{3}$$

$$\begin{cases} Ce = \sum C_{in}(C_{out}) \text{ (внешние зависимости)}, \\ Ca = \sum C_{out}(C_{in}) \text{ (внутренние зависимости)}, \\ H = \frac{R+1}{N} \text{ (среднее число внутренних связей)}, \\ DIT = \sum base(C) \text{ (глубина наследования)}. \end{cases} \tag{4}$$

В качестве базовой меры будет использоваться среднее число внутренних связей. В этом случае комплексная мера Кокола будет определяться по формуле

$$H_{coupl} = \frac{H + R_1 \times M(Ce) + R_2 \times M(Ca) + R_3 \times M(DIR)}{1 + R_1 + R_2 + R_3}. \quad (5)$$

Группа мер **структурированности программного кода** отражает степень разбиения программного кода на функциональные блоки и уровень инкапсуляции. Чем больше программа структурирована, тем лучше она поддается модификации, пониманию и отладке [5]. Кроме того, в хорошо структурированном программном коде повышается возможность повторного использования кода. В качестве основных мер для этой группы используются метрика Хансена и метрика Пивоварского. Обе эти меры являются модификацией меры Мак-Кейба и отражают структурированность программы. Таким образом, эту группу измерений можно представить системой

$$\begin{cases} S = \{m - n + 2\} \text{ (метрика Хансена)}, \\ N(G) = n^*(G) + \sum P_i \text{ (метрика Пивоварского)}. \end{cases} \quad (6)$$

Для вычисления комплексной меры структурированности кода в качестве базовой меры используется метрика Пивоварского. Тогда комплексная мера Кокола для этой группы может будет вычисляться по формуле

$$H_{struct} = \frac{N(G) + R_1 \times M(S)}{1 + R_1}. \quad (7)$$

Наконец, последняя группа измерений - это объемные характеристики кода. Данная группа мер не является прямым показателем качества программного кода. Однако значения данных измерений может быть опосредованным индикатором структурированности и качества кода. Например, эта группа измерений способна информировать разработчиков о том, что какой-либо класс содержит слишком большое или слишком маленькое количество методов и необходимо пересмотреть данный участок кода. В качестве основы для измерения данных характеристик используется метод Холстеда и набор метрик, построенных на основе отношения размеров программных сущностей (8).

Базовой мерой в комплексной мере Кокола будет использоваться отношение количества классов к количеству методов. В этом случае значение комплексной меры будет вычисляться по формуле (9).

Таким образом, мы получили нормализованные характеристики по всем четырем группам характеристик, отражающих качество программного кода. Общий вид системы метрик для оценки качества вычисляется по формуле (10).

Заключение

На сегодняшний день проблема контроля и качества кода является актуальной для большого числа разработчиков. При решении этой задачи требуется наличие инструмента, который позволяет это качество оценить. Для этого требуется на-

$$\begin{cases} V = N \times \log_2 n \text{ (мера Холстеда – объем программы)}, \\ R_{class} = \dfrac{\sum C}{\sum P} \text{ (отношение числа классов к числу пакетов)}, \\ R_{method} = \dfrac{\sum M}{\sum C} \text{ (отношение числа методов к числу классов)}, \\ R_{fields} = \dfrac{\sum F}{\sum C} \text{ (отношение числа полей к числу классов)}, \\ N = n_1 \times \log_2 n_1 + n_2 \times \log_2 n_2 \text{ (мера Холстеда – длина программы)}. \end{cases} \quad (8)$$

$$H_V = \frac{R_{method} + R_1 \times M(N) + R_2 \times M(V) + R_3 \times M(R_{class}) + R_4 \times M(R_{fields})}{1 + R_1 + R_2 + R_3 + R_4} \quad (9)$$

$$\begin{cases} H_{compl} = \dfrac{\lambda(G) + R_1 \times M(H) + R_2 \times M(S) + R_3 \times M(T) + R_4 \times M(E_c) + R_5 \times M(D)}{1 + R_1 + R_2 + R_3 + R_4 + R_5}, \\ H_{coupl} = \dfrac{H + R_1 \times M(Ce) + R_2 \times M(Ca) + R_3 \times M(DIR)}{1 + R_1 + R_2 + R_3}, \\ H_{struct} = \dfrac{N(G) + R_1 \times M(S)}{1 + R_1}, \\ H_V = \dfrac{R_{method} + R_1 \times M(N) + R_2 \times M(V) + R_3 \times M(R_{class}) + R_4 \times M(R_{fields})}{1 + R_1 + R_2 + R_3 + R_4} \end{cases} \quad (10)$$

личие математической модели и соответствующих алгоритмов. При этом на сегодняшний день уже существует несколько сотен различных метрик кода, которые отражают те или иные аспекты программного кода. Однако использование необработанных значений этих метрик не всегда может быть эффективным. Поэтому требуется наличие системы метрик, которая объективно показывала различные свойства программного кода (в данном случае - качество).

В данной работе была сделана попытка построения математической модели, которая способна оценить качество кода по четырем критериям: сложность, связность, структурированность и объем. Для этих целей использовались уже существующие метрики кода, которые объединялись по анализируемой области в систему. Вычисление комплексной меры осуществляется на основе метрики П. Кокола.

Литература

1. Кайгородцев, Г.И. Введение в курс метрической теории и метрологии программ / Г.И. Кайгородцев. -Новосибирск: НГТУ, 2009 - 187 с.

2. Черноножкин, С.К. Меры сложности программ (Обзор) / С.К Черноножкин // Системная информатика. Вып. 5 Архитектурные, формальные и программные модели. — Новосибирск: Наука, 1997. - С 188-227

3. Холстед, МХ. Начало науки о программах / М.Х. Холстед; пер. с англ. В.М. Юрфы. — М.: Финансы и статистика, 1981.

4. Липаев, В.В. Качество программного обеспечения / В.В. Липаев. - М. *Финансы и статистика, 1983.*

5. Шелехов, В. И. Структура программы в языково-ориентированном потоковом анализе / В.И. Шелехов // Программирование. - Новосибирск: Наука, 1998. - № 6.

***Поступила в редакцию 22 октября 2009* г.**