УДК 004.519.217
**Д.А. Маевский**, канд. техн. наук,
**С.А. Яремчук**

# ОЦЕНКА КОЛИЧЕСТВА ДЕФЕКТОВ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ НА ОСНОВЕ МЕТРИК СЛОЖНОСТИ

***Аннотация.*** *Проведен анализ существующих моделей оценки количества программных дефектов, выполнен анализ программной сложности, ее взаимосвязей с метриками сложности и дефектами программного обеспечения. Разработана методика подсчета значений метрик. Предложены математическая модель и методика оценки количества программных дефектов.*

***Ключевые слова****: метрики сложности, модель оценки количества программных дефектов, методика оценки количества программных дефектов.*

**D.A.Maevsky**, PhD.,
**S.A.Yaremchuk**,

# THE ESTIMATION OF NUMBER OF DEFECTS IN THE SOFTWARE ON THE BASIS OF COMPLEXITY METRICS

***Abstract.*** *We have analyzed the existing models of software defect number evaluation, program complexity, its correlations with software metrics and defects. The methods of metrics calculation have been developed. We proposed a mathematical model and a method of software defect number evaluation.*

***Keywords****: complexity metrics, models of software defect number evaluation, method of software defect number evaluation.*

**Д. А. Маєвський**, канд. техн. наук,
**С. О.Яремчук**

# ОЦІНКА КІЛЬКОСТІ ДЕФЕКТІВ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ НА ОСНОВІ МЕТРИК СКЛАДНОСТІ

***Анотація*** *Проведено аналіз існуючих моделей оцінки кількості програмних дефектів, виконано аналіз програмної складності, її взаємозв'язків з метриками складності і дефектами програмного забезпечення. Розроблено методику підрахунку значень метрик. Запропоновано математичну модель і методику оцінки кількості програмних дефектів.*

***Ключові слова****: метрики складності, модель оцінки кількості програмних дефектів, методика оцінки кількості програмних дефектів.*

## Актуальность проблемы и постановка задач исследования

Программное обеспечение (ПО) современных информационных систем (ИС) представляет собой чрезвычайно сложное изделие, в создании которого могут принимать участие десятки и сотни специалистов. Многие исследователи называют сложность современного ПО ключевым фактором возникновения дефектов [1]. С ростом сложности и количества дефектов ПО увеличиваются размеры потерь в случае отказов и зависимость конечных пользователей ИС от его безотказной работы.

В настоящий момент существует проблема достоверности и точности оценки количества дефектов. Поэтому разработка моделей и методик оценки количества дефектов ПО является актуальной научно-технической задачей, решение которой позволит получить достоверную и точную оценку количества дефектов, оптимальным образом спланировать необходимые ресурсы для устранения дефектов и достижения требуемой надежности, повысить эффективность процессов разработки и эксплуатации ПО.



**Цель исследования.** На основе метрик сложности разработать математическую модель и методику оценки количества дефектов ПО.

**Задачи исследования:**

1) выполнить анализ точности оценки существующих моделей;

2) провести анализ программной сложности с использованием классификации метрик, и методику расчета значений по метрикам;

3) разработать математическую модель

оценки количества дефектов ПО, выполнить верификацию модели;

4) разработать методику оценки количества дефектов ПО.

**1. Анализ точности существующих моделей**

Априорные модели оценки количества дефектов используются на этапе разработки до начала тестирования и эксплуатации ПО. Они позволяют оценить количество дефектов, внесенных разработчиками на этапе проектирования и кодирования. Для оценки количества дефектов априорные модели используют показатель количества строк программного кода (Lines of Code – LOC). LOC-оценка является наиболее простой и распространенной метрикой размера ПО. В стандартизованном словаре терминологии программной инженерии *метрика* ПО определяется как мера, позволяющая получить численное значение некоторого свойства программного обеспечения. Определенный вид программной сложности количественно выражается значением оценки по конкретной метрике.

Имеющиеся априорные модели оценки количества дефектов Гафнии и Акиямы исследованы на предмет достоверности и точности оценки количества дефектов в работе [7]. Обнаружена закономерность превышения расчетного количества дефектов над фактическим в несколько раз для модели Акиямы и в десятки раз для модели Гафнии.

Снижение уровня дефектов в современном ПО объясняется прогрессом методов, средств и технологий программной инженерии, увеличением степени повторного использования бездефектного кода, возможностью современных сред разработки автоматически генерировать бездефектный код. Поэтому эти модели потеряли свою актуальность, их оценка является недостоверной.

Физический размер ПО в виде LOC-оценки не содержит информации о его сложности, поэтому не дает достоверной оценки количества дефектов. В то же время многие исследователи отмечают высокую корреляцию между сложностью программы, частыми ошибками и низкой надежностью. Для количественной оценки сложности программного кода разработано более полусотни различных метрик сложности[2,3,6]. Имеющиеся *модели сложности* ПО основаны на предположении, что количество дефектов может быть предсказано с помощью метрик сложности, поскольку, чем сложнее программа, тем выше вероятность ошибки программиста при ее написании.

Наиболее известной и классической является система метрик Холстеда, предложенная автором в работе [6] и исследованная на предмет точности оценки количества дефектов в работе [7]. Наибольшее абсолютное отклонение составило 87 %, среднее отклонение составило 49 % как в сторону занижения, так и в сторону завышения количества дефектов. Высокие отклонения можно объяснить следующими недостатками данной модели:

1) линейные, циклические, условные операторы в системе метрик Холстеда имеют одинаковую сложность, что не соответствует действительности;

2) модель не учитывает межмодульную структурную сложность ПО, порождающую значительное количество дефектов;

3) модель не учитывает факт повторного использования бездефектного кода.

В [4] показано, что для количественной оценки сложности кода можно использовать метрики МакКейба, Джилба и Чепина. Их преимуществами являются: метрики разработаны еще в 80 годы, хорошо изучены и часто применяются; расчет значений по метрикам легко автоматизируется.

Для оценки количества дефектов фирмой TRW предложено использовать многофакторную *модель сложности* [5]:

$$B = L_{tot}k_1 + 0.1C_{\inf}k_2 + 0.2C_ck_3 + 0.4C_{io}k_4 + (-0.1)U_{read}k_5, \quad (1)$$

где $L_{tot}$ – сложность управляющей логики программы (количество операторов циклов, условий, переходов); $C_{\inf}$ – сложность взаимосвязей (количество системных и прикладных программ, вызываемых из модуля); $C_c$ – сложность вычислений (количество вычислительных операторов); $C_{io}$ –

сложность ввода-вывода (количество операций ввода-вывода); $U_{read}$ – читабельность ПО, оцениваемая как количество комментариев в нем; $k_i$ – коэффициент корреляции количества дефектов с $i$-м показателем сложности. Полученные авторами [5] коэффициенты корреляции показателей сложности различаются для разных подсистем проекта, например, $0{,}63 \le k_1 \le 0{,}91$, $0{,}43 \le k_2 \le 0{,}76$, что в проведенных расчетах количества дефектов дает наибольшее абсолютное отклонение 44 %, среднее отклонение – 31 %. Данная модель не учитывает фактора повторного использования бездефектного кода. Для получения значимых коэффициентов корреляции модели необходим представительный экспериментальный ряд показателей сложности и дефектов программных проектов, что затрудняет использование модели.

Таким образом, существующие априорные модели сложности, оценивающие количество дефектов ПО, являются либо не достоверными (модели Акиямы и Гафнии [7]), либо не достаточно точными (модели Холстеда и фирмы TRW). Они показывают значительные отклонения расчетного количества дефектов от действительного.

**2. Анализ программной сложности и методика расчета значений по метрикам**

В наиболее общем виде представим сложность программного кода следующими основными категориями: 1) сложность конструкций данных; 2) сложность циклических, условных и безусловных структур; 3) сложность вычислений и преобразований; 4) сложность вызовов модулей/классов. В зависимости от специфики ПО этот список может быть изменен или дополнен. На рис.1 представлены составляющие программной сложности и их взаимосвязи с метриками и дефектами.

С одной стороны, каждая категория сложности учитывается соответствующей метрикой и численно измеряется ее значением. С другой стороны, сложность кода является основной объективной причиной возникновения программных дефектов [1]. Чем выше сложность программного кода, тем выше оценка сложности по метрике и больше количество дефектов. Множество рассчитанных значений оценки сложности по метрике $M = \{M_i | i = 1...n\}$ будет количественно выражать различные категории сложности программного кода. Поскольку различные виды сложности можно количественно измерить метриками сложности, логично предложить использовать числовые значения оценок по метрикам для оценки количества дефектов ПО.

При разработке модели оценки дефектов на основе метрик сложности необходимо учитывать факт использования заведомо бездефектного системного и служебного кода. Кроме того, зачастую в ПО модули/классы используются повторно, будучи бездефектными после отладки в предыдущих проектах. Во избежание искажения соответствий значений по метрикам и количества дефектов определим понятие уникального кода, под которым будем понимать созданный разработчиком впервые и с определенной вероятностью содержащий дефекты программный код. Поскольку степень использования уникального и бездефектного кода в различных модулях проекта может быть разной, для количественного выражения этой степени введем понятие коэффициента уникальности кода модуля $k_u$ ($0 \le k_u \le 1$) в виде отношения

$$k_u = \frac{LOC_{new}}{LOC_{tot}}, \qquad (2)$$

где $LOC_{new}$ – LOC-оценка (без комментариев и пустых строк) нового уникального кода, созданного разработчиком в программном модуле; $LOC_{tot}$ – LOC-оценка всего кода программного модуля.

Для более точного определения значений оценки сложности кода по метрикам и корректности создаваемой модели предлагается учитывать коэффициент уникальности кода модуля $k_u$ умножением значений оценки сложности по метрике на коэффициент $k_u$ по следующей методике:

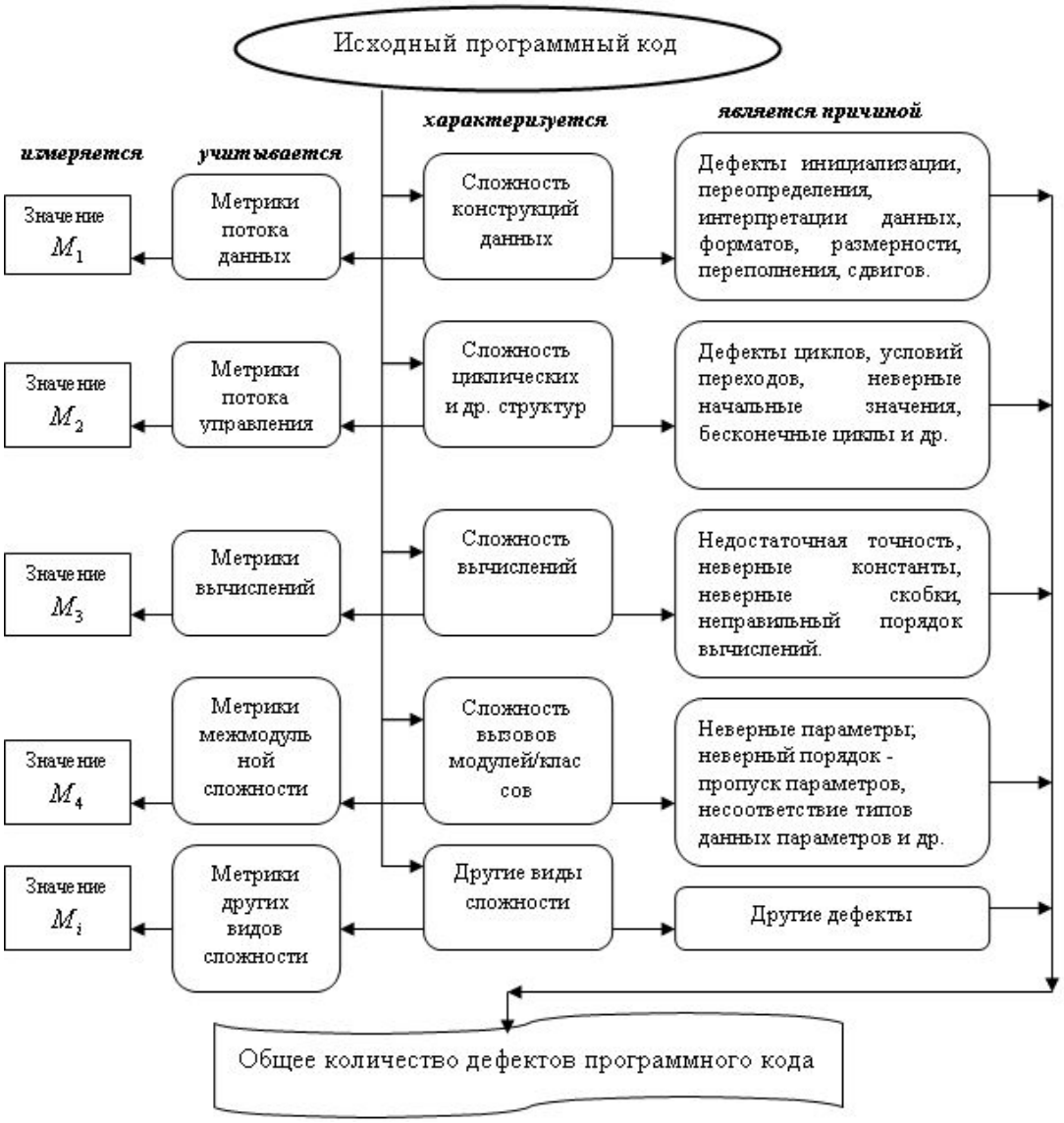



Рис. 1. Составляющие программной сложности и их взаимосвязи с метриками и дефектами

1) если исходный код полностью уникальный, значения по метрикам подсчитываются для всего кода умножением на $k_u = 1$;

2) если в исходном коде используются ранее разработанные модули/классы без внесения изменений, и эти модули представляется возможным выделить отдельно, значения по метрикам для них не подсчитываются, $k_u = 0$;

3) если в коде используются фрагменты уникального и бездефектного кода, метрическая оценка производится умножением значений по метрике на коэффициент $k_u$, определяемый по формуле (2). При невозможности подсчета $k_u$, связанной с высокой фрагментацией уникального и бездефектного кода, определять $k_u$ методом экспертных оценок;

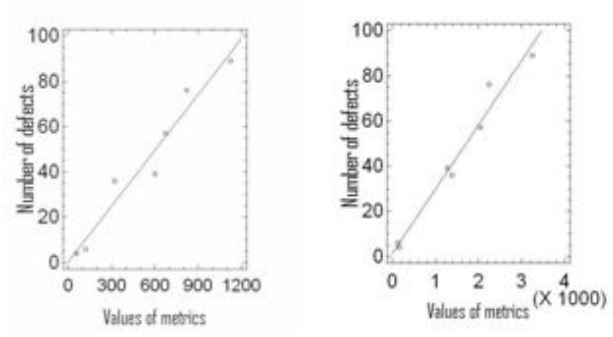
4) количественные значения по каждой выбранной метрике подсчитываются отдельно для каждого модуля программного кода с учетом коэффициента $k_u$, затем суммируются согласно выражению

$$M_{total} = \sum_{i=1}^{k} M_{\mathrm{mod}\,ul} \qquad (3)$$

где $M_{\mathrm{mod}\,ul}$ – значение по конкретной метрике для одного модуля; $i$ –количество модулей в ПО; $M_{total}$ - общее значение по конкретной метрике для кода в целом.

### 3. Математическая модель оценки количества дефектов ПО, верификация модели

Высокий коэффициент корреляции $R$ между количеством дефектов и значениями оценок сложности по метрикам в интервале $0{,}92 < R < 0{,}98$ свидетельствует о тесной линейной связи этих показателей [7], что подтверждается также исследованиями в работах [1,5]. Графики зависимостей количества дефектов от значений по метрикам МакКейба и Чепина представлены на рис.2.




а б

Рис.2. Графики зависимостей количества дефектов от значений по метрикам:
а – МакКейба; б – Чепина.

Графики отражают очень близкую к линейной зависимость этих показателей, что дает основание предположить линейную интегральную (аддитивную) зависимость между значениями метрик и количеством дефектов ПО в виде выражения

$$N_d = a_1 M_1 + a_2 M_2 + \ldots + a_n M_n, \quad (4)$$

где $M_1, M_2, \ldots M_n$ – известные значения по метрикам программного проекта, подсчитанные с учетом коэффициента $k_u$ $N_d$ – известное экспериментальное количество дефектов кода, обнаруженных при тестировании; $a_1, a_2, \ldots a_n$ – неизвестные коэффициенты, которые назовем коэффициентами дефектосложности ПО. Для определения коэффициентов необходима система уравнений

$$\begin{cases} a_1 M_{11} + a_2 M_{12} + \ldots + a_n M_{1n} = N_1 \\ a_1 M_{21} + a_2 M_{22} + \ldots + a_n M_{2n} = N_2 \\ \ldots. \\ a_1 M_{n1} + a_2 M_{n2} + \ldots + a_n M_{nn} = N_n \end{cases}, \quad (5)$$

где $M_{11}, M_{21}, \ldots M_{nn}$ – известные значения по метрикам, полученные с учетом коэффициента $k_u$ для $n$-го количества программных проектов; $N_1, N_2 \ldots N_n$ – известное количество дефектов программных проектов, обнаруженных при тестировании. Для корректного решения системы (5) необходимо, чтобы количество программных проектов было не меньше количества выбранных метрик.

Выражение (4) является математической моделью оценки дефектов программного кода на основе значений по метрикам сложности. Решение (5) позволяет определить значения коэффициентов дефектосложности в виде вектора $A = \{a_i | i = 1, \ldots, n\}$. Полученные коэффициенты и значения по метрикам будут использоваться разработчиками для оценки количества дефектов в последующих программных проектах согласно (4). Данная модель предполагает, что при разработке программных проектов интенсивность внесения разработчиками дефектов в код оставалась постоянной, т.е. не менялись квалификация программистов, знания предметной области, уровень организации процессов и Case-технологии.

Для верификации модели использовались исходные коды семи программных проектов учетно-аналитического характера, разработанные IT-компанией (г. Измаил). Программные проекты различаются своей функциональностью и имеют различный размер: от 1 до 30 тысяч строк кода.

Для количественной оценки сложности кода использованы метрики МакКейба, Джилба и Чепина. Результаты расчетов приведены в таблице 1.

**Цель расчетов** – на основании значений по метрикам и количества дефектов составить систему уравнений (5) и определить коэффициенты дефектосложности. Используя полученные коэффициенты и имеющиеся значения по метрикам, оценить количество дефектов согласно зависимости (4) для других программных проектов, сравнить его с фактическим количеством. Расчеты выполнены для двух, трех и четырех метрик для выяснения влияния количества выбранных метрик на точность оценки количества дефектов по модели. Представим эти расчеты.

1. Экспериментальное и расчетное количество дефектов.

| Программные проекты | Кол. дефектов | Значение оценки по метрикам | | | | $k_{unic}$ |
|---|---|---|---|---|---|---|
| | | МакКейба | Чепина | Джилба | Сложности вычислений | |
| 1-й | 4 | 58 | 157 | 8 | 161 | 1 |
| 2-й | 6 | 120 | 128 | 5 | 121 | 1 |
| 3-й | 36 | 320 | 1379 | 126 | 664 | 0,6 |
| 4-й | 39 | 601 | 1274 | 86 | 1213 | 1 |
| 5-й | 57 | 673 | 2039 | 191 | 1773 | 0,85 |
| 6-й | 76 | 819 | 2246 | 278 | 1812 | 1 |
| 7-й | 89 | 1123 | 3242 | 276 | 2463 | 1 |

1. Использованы значения по метрикам МакКейба, Чепина, Джилба для проектов 1, 2, 3. Для проектов 4,5,6,7 максимальное отклонение составило +7,71 %, дисперсия – 2,15.

2. Использованы значения по метрикам МакКейба, Чепина, Джилба для проектов 4, 5, 6. Максимальное отклонение для проектов 1,2,3,7 составило – 4,23 %, дисперсия – 27,57.

3. Использованы значения по метрикам МакКейба, Чепина и метрики сложности вычислений для проектов 1, 2, 3. Максимальное отклонение для проектов 4,5,6,7 составило -**7**,3 %, дисперсия – 0,005.

4. Использованы значения по метрикам МакКейба, Чепина, Джилба и сложности вычислений для проектов 1, 2, 3, 4. Для проектов 5,6,7 максимальное отклонение составило – 3,82 %, дисперсия – 28,80.

5. Использованы значения по метрикам МакКейба, Чепина, Джилба и сложности вычислений для проектов 4, 5, 6, 7. Для проектов 1,2,3 максимальное отклонение составило -5,67%, а дисперсия – 30,25.

6. Использованы значения по метрикам МакКейба, Чепина, Джилба и сложности вычислений для проектов 2, 3, 4, 5. Для проектов 1,6,7 максимальное отклонение составило – 1,40%, дисперсия – 12,69.

7. Выполнены три расчета оценки дефектов с использованием двух метрик для проектов 1,2 с последующей оценкой количества дефектов для проектов 3,4,5,6,7. Получены результаты: для метрик МакКейба и Чепина максимальное отклонение – 15,24 %, дисперсия 0,006; для метрик МакКейба и Джилба: максимальное отклонение +12,06 %, дисперсия 0,002; для метрик МакКейба и сложности вычислений: максимальное отклонение –16,31 %, дисперсия 0,006.

Расчеты п.7 показали меньшую точность оценки. Это можно объяснить тем, что использования только двух метрик недостаточно для представления комплексной сложности кода и точной оценки количества дефектов.

**4. Методика оценки количества дефектов**

Данная методика разработана на основании предложенной модели и методики расчета значений количества дефектов по метрикам. Для оценки количества дефектов необходимо:

1) на основе анализа исходного программного кода определить различные категории его сложности и выбрать множество необходимых метрик $M = \{M_i | i = 1...n\}$;

2) по выбранным метрикам рассчитать количественные значения для модулей/классов программного кода с учетом коэффициента уникальности кода $k_u$ согласно приведенной выше методике;

3) просуммировать значения соответствующих метрик для всех модулей/классов согласно (3) и получить значения для программного кода в целом

4) пункты 1 – 3 повторить для других программных проектов. Общее количество проектов не должно быть меньше количества выбранных метрик;

5) найти коэффициенты дефектосложности как решение системы (5);

6) для прогнозной оценки количества дефектов другого программного проекта

данного разработчика подсчитать значения по выбранным в п.1 метрикам $M = \{M_i | i = 1...n\}$ для модулей/классов с учетом коэффициента уникальности кода $k_u$ согласно приведенной выше методике;

7) согласно (3)получить суммарные значения по выбранным метрикам для программного кода;

8) используя суммарные значения метрик и полученные в п. 5 коэффициенты дефектосложности, рассчитать прогнозное количество дефектов по формуле (4).

**Выводы**

В работе выполнен анализ сложности ПО и анализ взаимосвязей показателей сложности, метрик и дефектов. Для повышения точности подсчета значений по метрикам предложена методика определения коэффициента уникальности кода $k_u$. Разработана математическая модель оценки количества дефектов программного кода на основе метрик сложности.

Выполнена верификация модели, результаты представлены в табл. 2. Предложенная модель оценивает количество дефектов более точно, чем модель Холстеда и модель фирмы TRW. По сравнению с моделями Акиямы и Гафнии точность оценки количества дефектов выше в несколько раз [7].

На основе предложенной математической модели разработана методика оценки количества программных дефектов с использованием метрик сложности, ориентированная на практическое применение софтверными компаниями.

Преимущества методики заключаются в повышении точности оценки, в сравнительно небольшом объеме необходимых исходных данных, а также в простоте математических методов решений.

Практическое использование предложенной методики оценки дефектов позволяет повысить эффективность разработки ПО за счет оптимизации процесса тестирования и возможности своевременного принятия решений об окончании отладки.

Дальнейшие исследования должны быть направлены на проверку предложенной методики в программных проектах других разработчиков.

2. Сравнительная оценка количества дефектов

| Набор метрик | Отклонение | |
|---|---|---|
| | % | Дисперсия |
| 2-е | -15,24 | 0,0060 |
| | 12,06 | 0,0020 |
| | -16,31 | 0,0060 |
| Макс. | **-16,31** | 0,0060 |
| Среднее | **-6,50** | 0,0050 |
| 3-и | 7,71 | 2,1500 |
| | -4,23 | 27,5700 |
| | -7,30 | 0,0050 |
| Макс. | **7,71** | 27,5700 |
| Среднее | **-1,27** | 14,3238 |
| 4-и | -3,82 | 28,8000 |
| | -5,67 | 30,2500 |
| | -1,40 | 12,6900 |
| Макс. | **-5,67** | 30,2500 |
| Среднее | **-3,63** | 25,4975 |
| Метрики Холстеда | | |
| Макс. | 87,00 | |
| Среднее | 49,00 | |
| Модель фирмы TRW | | |
| Макс. | 44,00 | |
| Среднее | 31,00 | |

Список использованной литературы

1. Макконнелл С. Совершенный код. Мастер-класс. / С. Макконел. – СПб. : Питер, 2005. – 896 с.

2. Метрики сложности программного обеспечения [Электронный ресурс] – Режим доступа: http://metrix.narod.ru/page1.htm.

3. Программный код и его метрики [Электронный ресурс] – Режим доступа: http://habrahabr.ru.

4. Современные информационные и электронные технологии: сб. науч. трудов / Д. А.Маевский, С. А.Яремчук. Соответствие метрик сложности и количества дефектов в программном обеспечении. – Одесса: СИЭТ, 2012. – 25 с.

5. Тейер Т. Надежность программного обеспечения. / Т. Тейер, М. Липов, Э. Нельсон. М.: Мир, 1981. – 326 с.

6. Холстед М.Х. Начала науки о программах. / М. Х. Холстед. – М.: Финансы и статистика, 1981. – 128 с.

7. Яремчук С.А. Метод оценки количества программных дефектов с использованием метрик сложности. / С.А.Яремчук // Радиоэлектронные и компьютерные системы. – 2012. – № 5. – С. 212–218.

Получено 03.07.2012

References

1. McConnell S. Code Complete. Masterclass. – Sankt-Petersburg: Piter, 2005. – 896 p.[rus]

2. Software Complexity Metrics [Internet source] – available: http://metrix.narod.ru/page1.htm [rus]

3. Software code and metrics [Internet source] – available: http://habrahabr.ru [rus]

4. Modern information and electronic technologies.: scientific works collection / D. A. Mayevskiy, S. A. Yaremchuk. The correlation between complexity metrics and the number of software defects. – Odessa: MIET, 2012. – 25 p.[rus]

5. Teyer T., Lipov M., Nelson E,. Software reliability. – Moscow: Mir, 1981. – 326p. [rus]

6. Halstead M. H. Elements of software science. –Moscow: Finances and Statistics, 1981. – 128 p. [rus]

7. Yaremchuk S. A. The method of software defects number evaluation using complexity metrics / S. A. Yaremchuk // Radioelectronic and computer systems. – 2012. – № 5. – P. 212–218. [rus]

Маевский
Дмитрий Андреевич, канд. техн. наук, зав. каф. теорет. основ и общ. электротехники Одесского нац. политехн. у-та,
тел. (048) 734-84-54.

Яремчук
Светлана Александровна, ст. пр. каф. информ. управл. систем и технологий Измаильского и-та водного транспорта,
тел.(04841) 2-02-80.