УДК 004.052.4

Д.А. МАЕВСКИЙ[1], С.А. ЯРЕМЧУК[2]

[1] ***Одесский национальный политехнический университет, Украина***
[2] ***Аспирант ОНПУ, Одесса, Украина***

АПРИОРНАЯ ОЦЕНКА КОЛИЧЕСТВА ДЕФЕКТОВ В ПРОГРАММНОМ ОБЕСПЕЧЕНИИ ИНФОРМАЦИОННЫХ СИСТЕМ

В работе выполнен анализ существующих моделей и методов оценки количества дефектов в программном обеспечении информационных систем. Предложена априорная логистическая модель оценки количества дефектов на основе комплексных метрик сложности. Сформулированы допущения модели, сделан выбор наиболее информативных метрик и определено их минимально необходимое количество для оценки дефектов объектно - ориентированного программного кода. Доказана работоспособность исследуемой модели для программных проектов различных разработчиков. Показано, что точность оценки количества дефектов зависит от степени выполнения допущений. Предложена численная характеристика выполнения допущений – индекс различия изменений и определен способ его расчета. Предложенная модель может быть использована на этапе тестирования программного обеспечения для оценки имеющегося количества дефектов и планирования на ее основе процесса тестирования.

Ключевые слова*: априорная логистическая модель надежности, комплексная программная сложность, метрики сложности, программные дефекты.*

Введение

В настоящее время информационные системы стали неотъемлемой частью жизни человека. Они охватывают все стороны деловой и личной жизни каждого человека, делая ее динамичней, насыщенней и интересней. Однако негативной стороной этого процесса является возрастающая зависимость человека от надежности, безопасности и корректности работы программного обеспечения (ПО) современных информационных систем.

Современное ПО все более полно отражает предметные области сложного и динамичного реального мира, что объективно ведет к повышению его сложности. Растущая сложность ПО – один из основных факторов увеличения количества программных дефектов и снижения надежности. При этом возникает противоречие между объективным повышением уровня сложности, увеличением количества дефектов и снижением надежности ПО, с одной стороны, и необходимостью обеспечения надежности при одновременном снижении затрат на разработку ПО с другой стороны.

Разрешение этого противоречия невозможно, в частности, без разработки методов и информационных технологий оценки количества дефектов в имеющемся программном продукте. Несмотря на значительные успехи в разработке таких методов, задача еще далека от решения и практической реализации. Особенно это касается методов априорной оценки качества разработки ПО до начала его тестирования или эксплуатации. Наличие такой оценки способствует обоснованному использованию ограниченных ресурсов тестирования, увеличивает эффективность разработки, позволяет определить количество оставшихся дефектов и способствует оптимальной организации сопровождения ПО.

Поэтому задача, связанная с разработкой метода априорной оценки количества дефектов в программном обеспечении информационных систем, является актуальной.

Современное состояние проблемы

Априорная оценка количества дефектов в программном обеспечении информационных систем возможна только на основе косвенных данных – значениях метрик сложности. Интуитивно понятно, что количество дефектов в ПО должно быть пропорционально степени его сложности.

Подход к оцениванию количества дефектов в ПО на основе метрик сложности не является новым. Вот уже более сорока лет в ведущих университетах мира проводятся исследования в этой области. Основная цель исследований – с удовлетворительной точностью оценить количество дефектов и место их нахождения в разработанном ПО до начала его тестирования. Наиболее раннее исследование проведено Ф. Акаямой [1], который показал, что линейные регрессионные модели, использующие количество строк исходного кода L, обеспечивают приемлемые оценки для общего количества дефектов N_d, обнаруженных во время тестирования и в течение двух

первых месяцев эксплуатации, в виде уравнения

$$N_d = 4{,}86 + 0{,}018 \cdot L .$$

В [2] предложено использовать метрику L_i – количество строк исходного кода i -го модуля ПО в виде выражения

$$N_d = 4{,}2 \cdot n + 0{,}0015 \cdot \sum_{i=1}^{n} L_i^{4/3} ,$$

где n – количество модулей в исследуемом ПО.

В [3] получено выражение для количества дефектов в виде полинома:

$$N_d = 0{,}00000047 \cdot L^2 + 0{,}0156 \cdot L + 0{,}069 .$$

В работе [4] исследовано соотношение между количеством дефектов и значениями нескольких различных метрик. Получены следующие регрессионные уравнения:

– с использованием метрик Холстеда:

$$N_d = 0{,}042 \cdot MCI - 0{,}075 \cdot N + 0{,}00001 \cdot HE ,$$

– с использованием метрик МакКейба:

$$N_d = 0{,}25 \cdot MCI - 0{,}53 \cdot DI + 0{,}09 \cdot VG ,$$

где MCI – количество команд машинного кода, N – длина программы по Холстеду (общее количество операторов и операндов), HE – метрика усилия по Холстеду, DI и VG – количество элементов данных и показатель сложности программы по метрике МакКейба.

За последние годы предложено ряд новых регрессионных моделей и методик прогнозирования количества и места нахождения дефектов. В работе [5] авторы представляют всестороннее эмпирическое исследование восьми регрессионных моделей оценки количества дефектов в крупномасштабной индустриальной информационной системе. Другие разработанные методы прогнозной оценки на основе Байесовских сетей доверия требуют получения экспертных оценок условных вероятностей факторов, влияющих на количество дефектов, что означает необходимость привлечения персонала, обладающего специальной квалификацией.

Одним из современных методов оценки количества дефектов является метод машинного обучения. В работе [6] авторы исследуют производительность прогнозирующих моделей для обнаружения склонных к дефектам модулей в исследуемом ПО. Однако для выполнения эффективного прогноза по этим моделям необходим большой объем экспериментальных данных.

Для оценки количества склонных к дефекту программных модулей применялся подход, основанный на использовании нейронных сетей. Однако сложность построения нейронных сетей препятствует их внедрению в инженерную практику, что является существенным недостатком данного подхода. В [7], с целью обнаружения склонных к отказу программных модулей, предложена упрощенная модель нейронной сети и соответствующий ей кластеризирующий генетический алгоритм для извлечения набора правил из обучаемой нейронной сети. Анализ показал, что этот алгоритм обладает меньшей точностью прогноза по сравнению с обучаемой нейронной сетью, однако благодаря своей простоте он получил практическое применение в программной инженерии.

В работе [8] исследуется применение, разработанных в NASA, генетических алгоритмов машинного обучения для наборов экспериментальных данных ПО в среде C++, которые включают различные метрики. Однако за счет значительного объема требуемых исходных данных процессы обучения замедляются.

Таким образом, проведенный анализ существующих методов априорной оценки количества дефектов ПО позволяет выявить следующие их недостатки:

1. Существует большое количество линейных, степенных, полиномиальных и других регрессионных моделей. Однако, за счет их многообразия, ограниченности условий применения и коллинеарности независимых переменных, вызывающей нестабильность коэффициентов регрессии, применение этих моделей затруднено.

2. Байесовские сети доверия требуют экспертных оценок условных вероятностей, что требует привлечения персонала, обладающего специальной квалификацией, и неизбежно приводит к субъективности полученных оценок;

3. Методы машинного обучения с использованием различных генетических алгоритмов настолько сложны, что исследователи готовы пожертвовать точностью ради получения более простых и понятных подходов, доступных для практического применения. Кроме того, генетические алгоритмы требуют большого количества представительных данных.

Поэтому разработка простых инженерных методов точной оценки количества дефектов является актуальной научно – технической задачей. Эти методы должны быть основаны на учете объективно доминирующего фактора – комплексной программной сложности, выраженной посредством значений соответствующих метрик сложности.

В работе [9] предложена априорная логистическая модель надежности для оценки количества дефектов на основе КПС, выраженной посредством значений метрик сложности. В название модели введен термин «логистическая». В широком смысле термин «логистика» определяется как «системный подход к организации жизненного цикла товара ли-

бо изделия» [10]. В более узком смысле, в контексте программной индустрии, этот термин предполагает построение эффективной системы управления ЖЦ продукта, что и является конечным результатом использования модели.

Априорная логистическая модель построена с использованием следующих допущений:

1. Количество дефектов в ПП прямо пропорционально сложности этого проекта.
2. От проекта к проекту квалификация разработчиков не меняется, а эволюция их знаний и опыта не оказывает существенного влияния на количество дефектов в разработанных проектах.
3. От проекта к проекту не изменяются используемые методы, средства и технологии разработки.
4. Временные, финансовые, психологические и другие факторы не оказывают влияния на качество процесса разработки.

В модели [9] программная сложность является комплексным показателем и может быть численно выражена в виде набора различных метрик $M_1, M_2, ... M_n$. Тогда, исходя из допущения 1, количество дефектов в программном проекте (N_d) можно вычислить как:

$$N_d = a_1 \cdot M_1 + a_2 \cdot M_2 + ... + a_n \cdot M_n, \qquad (1)$$

где $a_1, a_2, ... a_n$ – неизвестные коэффициенты, названные ***коэффициентами дефектосложности***. Из допущений 2 – 4 следует, что для одного и того же разработчика (или команды разработчиков) от проекта к проекту коэффициенты дефектосложности не должны меняться. Это дает возможность выполнять априорное прогнозирование количества дефектов в новых проектах, определив коэффициенты дефектосложности на основе нескольких старых.

Коэффициенты дефектосложности можно определить из решения системы:

$$\begin{cases} a_1 \cdot M_{11} + a_2 \cdot M_{12} + ... + a_n \cdot M_{1n} = N_{d1}, \\ a_1 \cdot M_{21} + a_2 \cdot M_{22} + ... + a_n \cdot M_{2n} = N_{d2}, \\ ... \\ a_1 \cdot M_{n1} + a_2 \cdot M_{n2} + ... + a_n \cdot M_{nn} = N_{dn}, \end{cases} \qquad (2)$$

где $N_{d1}, N_{d2}, ..., N_{dn}$ – известное количество обнаруженных дефектов в ранее созданных программных проектах, M_{ij} – значение сложности i-го программного проекта, рассчитанное по j-й метрике. В работе [9] на семи программных проектах показано, что предложенная модель может быть использована на практике для априорной оценки количества дефектов, особенно с использованием предложенной метрики сложности вычислений [9, 11].

Однако, эта проверка выполнена только для программных проектов одного и того же разработчика. В связи с этим возникает закономерный вопрос – применима ли модель и какова точность оценки ею количества дефектов в программных проектах других разработчиков? При проведении подобного исследования необходимо учитывать следующие особенности.

1. Доминирующей парадигмой современной программной инженерии стало объектно-ориентированное программирование (ООП). Поэтому верификацию априорной логистической модели необходимо провести именно на проектах, выполненных с использованием ООП.
2. Поскольку данные софтверных компаний о количестве дефектов в программных проектах являются закрытой корпоративной информацией и недоступны для использования, целесообразно использовать данные о значениях метрик сложности и количестве дефектов в проектах с открытым исходным кодом.

Эти две особенности определили цель настоящего исследования.

Цель и задачи исследования

Целью исследования является верификация априорной логистической модели оценки количества дефектов для ООП проектов с открытым исходным кодом.

Для достижения поставленной цели решены следующие **задачи**:

- выполнен анализ экспериментальных данных на соответствие допущениям модели, на основании чего отобраны проекты для исследования;
- выполнен анализ метрик с целью обоснованного выбора наиболее информативных для оценки количества дефектов;
- выполнено сравнение прогнозных оценок количества дефектов с их экспериментальными значениями и проведен статистический анализ отклонений.

Выбор проектов для исследований

Для исследования априорной логистической модели использованы экспериментальные данные, опубликованные в депозитарии данных на сайте международной конференции PROMISE (PRedictOr Models In Software Engineering) [12].

По каждому программному проекту депозитарий предоставляет данные о количестве выявленных дефектов и значениях сложности, рассчитанных более чем по двадцати различным метрикам. Кроме того, коль скоро все проекты являются открытыми,

можно получить доступ к их исходным кодам. Для исследования отобраны 45 программных проектов, которые разбиты на 12 групп. Все проекты одной группы принадлежат одному и тому же разработчику. Каждая группа содержит от трех до пяти разных проектов. Размеры проектов составляют от одной до четырехсот тысяч строк исходного кода, общий объем по всем отобранным проектам – более пяти миллионов строк. Все проекты соответствуют различным предметным областям. В таблице 1 представлены сведения об источниках использованных экспериментальных данных.

Таблица 1

Сведения об источниках данных

Группа	Кол. ПП	Источник данных
1	3	http://code.google.com/p/promisedata/wiki/log4j
2	3	http://code.google.com/p/promisedata/wiki/synapse
3	3	http://code.google.com/p/promisedata/wiki/lucene
4	4	http://code.google.com/p/promisedata/wiki/poi
5	5	http://code.google.com/p/promisedata/wiki/ant
6	4	http://code.google.com/p/promisedata/wiki/camel
7	4	http://code.google.com/p/promisedata/wiki/xerces
8	4	http://code.google.com/p/promisedata/wiki/xalan
9	3	http://code.google.com/p/promisedata/wiki/forrest
10	3	http://code.google.com/p/promisedata/wiki/ivy
11	5	http://code.google.com/p/promisedata/wiki/jedit
12	3	http://code.google.com/p/promisedata/wiki/velocity

Отобранные экспериментальные данные ПП были проанализированы с точки зрения выполнения первого, основного допущения модели. Для анализа использовался введенный авторами «индекс различия» I, который вычисляется для каждой пары проектов как:

$$I = \frac{M_1 \cdot N_{d2}}{N_{d1} \cdot M_2},$$

где M_1 и M_2 – значения метрик сложности первого и второго проектов, а N_{d1} и N_{d2} – количество дефектов в этих проектах. В результате анализа исследуемые группы проектов были разделены на три категории:

Первая категория включает проекты со значением индекса различия $I \approx 1$, что соответствует прямой пропорциональности между значением метрики и количеством дефектов. В эту категорию попали первые четыре группы из приведенных в таблице 1.

Вторая категория включает проекты, для которых $I \approx 2$, что соответствует наличию пропорциональной связи между значением метрики и количеством дефектов, однако эта связь выражена нелинейной зависимостью. Для данных проектов основное допущение модели выполняется частично. В эту категорию попали вторые четыре группы из таблицы 1.

Третья категория включает проекты, для которых $I >> 2$, что соответствует отсутствию пропорциональности связи между значением метрики и количеством дефектов. Для данных проектов основное допущение модели не выполняется частично. В эту категорию попали последние четыре группы из таблицы 1. Можно предположить, что в этом случае количество дефектов в программном проекте обуславливается не только сложностью, но и какими-то другими факторами, требующими отдельного исследования.

Выбор метрик

Отобранные для исследования данные содержат показатели сложности, рассчитанные с использованием двадцати различных метрик кода, поразному связанные с одним и тем же показателем – количеством выявленных в данном проекте дефектов. С учетом того, что для решения системы (2) достаточно использовать количество метрик, равное количеству программных проектов, а число проектов в каждой группе не превышает пяти, следует выбрать метрики, участвующие в анализе. При этом, естественно, следует использовать только те метрики, которые обеспечивают выполнения первого условия – наличия прямой пропорциональности с количеством дефектов. Для отбора использовались значения парной (R_1) и множественной (R_n) корреляции между значениями метрик и количеством дефектов в каждом программном проекте. Предполагается, что метрики с более высокими значениями коэффициентов корреляции дадут более высокую точность оценки по априорной модели. В результате проведенного анализа были выбраны следующие наиболее информативные для оценки количества дефектов метрики [13, 14, 15] (таблица 2):

1) LCOM (Lack of cohesion in Methods), недостаток связности методов класса;

2) LOC (Lines Of Code), количество строк кода;

3) RFC (Response for a class), отклик для класса – показывает сумму количества методов класса и количества методов других классов, вызываемых из данного класса;

4) WMC (Weighted methods per class), суммарная сложность всех методов класса;

5) NPM (The Number of Public Methods), число общедоступных методов в классе;

6) CE (The ratio of efferent coupling), оценивает сложность категории классов;

7) CBO (Coupling between object classes), сцепление между классами, показ ывает количество классов, с которыми связан исходный класс.

На основании анализа минимальных значений R_1 не удалось установить, каких-либо закономерностей связи между показателями сложности и количеством дефектов.

В целом невысокие средние значения объясняются тем, что исследуемые 44 программных проекта разработаны специалистами различной квалификации, которые допускают в работе разное количество дефектов.

Таблица 2

Результаты ранжирования метрик

Метрика	Коэффициенты парной корреляции				Ранг
	Мин.	Макс.	Средн.	Дисп.	
LCOM	-0,091	0,896	0,316	0,047	1
LOC	-0,046	0,885	0,400	0,036	2
RFC	-0,027	0,865	0,464	0,043	3
WMC	-0,248	0,828	0,374	0,046	4
NPM	-0,497	0,793	0,336	0,060	5
CE	-0,281	0,738	0,374	0,039	6
CBO	-0,257	0,699	0,298	0,037	7

Максимальные значения показателя, больше 0,7, могут свидетельствовать о близкой к прямой пропорциональности связи между показателями сложности и количеством дефектов. Ранг метрики в таблице 2 определялся по этому показателю, на основании которого отобраны показанные семь метрик. Величина дисперсии незначительна (0,04 – 0,06), поэтому метрики можно считать устойчивыми при оценивании количества дефектов.

Однако, количество метрик и количество проектов должно совпадать. Получение таких данных не всегда возможно, поэтому количество реально используемых метрик будет меньшим. Чтобы уменьшить количество проектов для сбора данных, необходимо уменьшить количество метрик, применяемых для оценки. При этом следует избегать существенного падения точности оценивания. Для уменьшения количества метрик и одновременно выбора из них наиболее значимых предложено использовать коэффициенты множественной корреляции R_n. При этом полагается, что метрики с более высокими значениями R_n дадут более высокую точность оценки. С целью выяснения степени уменьшения R_n при уменьшении количества используемых метрик по сорока четырем исследуемым проектам выполнен расчет множественной корреляции для различного числа используемых метрик. Результаты представлены в таблице 3.

Анализ результатов позволил подтвердить интуитивно понятное положение о том, что максимальные значения множественной корреляции формируют метрики с более высоким значением парной. С ростом количества метрик множественная корреляция увеличивается незначительно (0,4795-0,4807).

Таблица 3

Множественная корреляция при различном числе метрик

Показатели	Мин.	Макс.	Средн.	Дисп.
2 метрики	0,337	0,479	0,405	0,0018
3 метрики	0,361	0,479	0,428	0,0011
4 метрики	0,383	0,480	0,438	0,0010
5 метрик	0,414	0,480	0,452	0,0008
6 метрик	0,480	0,480	0,480	0
7 метрик	0,480	0,480	0,480	0

Из этого следует, что для оценки количества дефектов достаточно двух, трех или четырех метрик с наиболее высоким значением R_1. Использование пяти метрик уже не дает увеличения множественной корреляции, а использование шести или семи метрик незначительно (на 0,0001) увеличивают ее. Однако, при этом значительно увеличиваются затраты на сбор экспериментальных данных для шести или семи программных проектов.

Для определения набора метрик, используемого для оценивания количества дефектов в данном наборе программных проектов, предложен следующий алгоритм:

1. Для очередного проекта выполнить ранжирование доступных метрик по коэффициенту парной корреляции.
2. Выбрать метрики с наибольшим рангом в соответствии с количеством проектов.
3. Определить ковариацию между значением каждой отобранной метрики и количеством дефектов. Ранжировать полученные значения ковариации.
4. Получить суммарные ранги метрик одного проекта сложением рангов, полученных на шаге 1 и 3.
5. Выполнить шаги 1 – 4 для других проектов данного разработчика.
6. Получить суммарные ранги по всем проектам сложением суммарных рангов по каждому проекту.
7. Выбрать необходимое количество метрик с наивысшими суммарными рангами.

Результаты априорных оценок количества дефектов

Априорное оценивание количества дефектов выполнялось для каждой группы программных проектов.

Рассмотрим пример оценивания на проектах первой группы, которая включает три проекта. Для

этих проектов по приведенному выше алгоритму были выбраны четыре наиболее значимые метрики с максимальными суммарными рангами: LCOM, WMC, LOC, RFC.

Показатели сложности по этим метрикам и экспериментальное количество дефектов для трех проектов этой группы приведены в таблице 4.

Таблица 4

Данные проектов группы 1

Проект	LCOM	WMC	LOC	RFC	N_d
1	2691	889	21549	2889	61
2	2943	853	19938	2531	86
3	11056	1727	38191	5171	498

Для сравнения точности оценок был выбран третий проект этой группы, расчетное количество дефектов в котором рассчитывалось исходя из данных первых двух проектов. Так как для прогноза на основе двух проектов достаточно только двух метрик, выполнено три прогноза, в которых к первой, самой весомой метрике LCOM добавлялись три оставшихся – WMC, LOC и RFC. Результаты оценивания приведены в таблице 5, в которой для каждого набора метрик приведено расчетное количество дефектов третьего проекта (N_d^*) и процент их отклонений от экспериментального.

Таблица 5

Результаты расчетов для проекта 3

Метрики	N_d^*	% откл.
LCOM, WMC	562	12,86
LCOM, LOC	518	4,05
LCOM, RFC	456	-8,39
Средн.откл.		2,84
Дисперсия		75,99

Для остальных одиннадцати групп проектов выполнены аналогичные расчеты и получены: математическое ожидание M и дисперсия D индекса различия изменений, расчетные оценки количества дефектов и их средние отклонения от экспериментальных. Полученные результаты представлены ниже для каждой из трех категорий проектов в отдельности.

Для проектов первой категории, где допущения модели выполняются, математическое ожидание и дисперсия индекса различия составляют соответственно M=1,38, D=0,26. Результаты расчетов для проектов с полностью выполняющимися допущениями приведены в таблице 6.

Здесь мы видим сравнительно небольшие отклонения предсказанного количества дефектов от их истинного количества.

Таблица 6

Результаты расчетов для проектов первой категории

Группа	% откл.	Дисперсия
1	2,84	75,99
2	22,22	99,21
3	12,67	502,9
4	8,09	2,23
Средние	11,46	170,08

Для проектов второй категории, где допущения модели выполняются частично, математическое ожидание и дисперсия индекса различия составляют соответственно M=1,87, D=0,24.

Результаты расчетов для проектов второй категории приведены в таблице 7.

Таблица 7

Результаты расчетов для проектов второй категории

Группа	% откл.	Дисп.
1	-46,31	2,52
2	49,54	2036,07
3	-34,69	499,43
4	-39,75	275,71
Средние	-17,80	703,43

Для проектов третьей категории, где допущения модели не выполняются, математическое ожидание и дисперсия индекса различия составляют соответственно M=5,49, D=33,18. Результаты расчетов для проектов первой категории приведены в таблице 8.

Таблица 8

Результаты расчетов для проектов третьей категории

Группа	% откл.	Дисп.
1	157	Не определялась
2	более 1000	
3	720	
4	90	
Средние	Оценка не достоверна	

Анализ полученных результатов

Анализ полученных результатов позволяет сделать следующие, приведенные ниже выводы.

Для проектов первой категории, где допущения модели выполняются, среднее отклонение оценок количества дефектов от экспериментальных данных составило 11,46% при использовании двух наиболее информативных метрик сложности. Все полученные отклонения оценок положительны, что говорит о завышении оценки количества дефектов.

С увеличением количества, используемых для оценки, метрик результат оценки можно незначительно улучшить, однако это требует значительного увеличения объема экспериментальных данных при неизменных условиях разработки ПП, что не всегда возможно, и в целом нецелесообразно.

Для проектов второй категории, где допущения модели выполняются частично, точность оценок уменьшается до 17,80% (в 1,55 раза), причем все отклонения отрицательны, что свидетельствует о систематическом занижении оценки.

Для проектов третьей категории допущения модели не выполняются. Здесь имеют место значительные нарушения пропорциональности между программной сложностью и количеством дефектов. При этом наблюдаются очень большие отклонения (от 90% до более чем 1000%), точность оценки снижается многократно, а оценка количества дефектов становится недостоверной.

Полученные результаты свидетельствуют о работоспособности модели для проектов различных разработчиков только при строгом соблюдении ее допущений. При выполнении допущений априорная логистическая модель оценки количества дефектов на основе метрик сложности показывает достаточно точные оценки. Модель вполне применима в практике программной инженерии разработчиками и тестировщиками программного обеспечения.

Актуальными представляются исследования априорной логистической модели оценки количества дефектов на разных структурных уровнях проектов: на уровне подсистем, компонент и отдельных модулей. Ранжирование программных модулей по количеству дефектов, полученных в результате оценки, позволило бы создать приоритетный список первоочередного тестирования, что снизило бы его продолжительность за счет сужения области поиска дефектов в исходном коде, и значительно повысило общую эффективность разработки ПО. Указанные положения должны определить основные направления дальнейших исследований в области априорной оценки количества дефектов программного обеспечения информационных систем.

Литература

1. Akiyama, F. An example of software system debugging [Text] / F. Akiyama // Proceedings of the International Federation of Information Processing Congress, Ljubljana, Yugoslavia. – 1971. – Vol .71. – P. 353–379.

2. Gaffney, J.R. Estimating the Number of Faults in Code [Text] / J.R Gaffney // IEEE Trans. Software Engineering. – 1984. – Vol. 10, No.4. – P. 459–465.

3. Compton, T. Prediction and control of Ada software defects [Text] / T. Compton, C. Withrow // The Journal of Systems & Software. – 1990. – Vol. 12, issue 3. – P. 199–207.

4. Kitchenham, B.A. An evaluation of some design metrics [Text] / B.A. Kitchenham, L.M. Pickard, S.J. Linkman // IEEE Trans. Software Engineering. – 1990. – Vol. 5, No. 1. – P. 50–58.

5. Gao, K. Comprehensive Empirical Study of Count Models for Software Fault Prediction [Text] / K. Gao, T. Khoshgoftaar // IEEE Transactions on Reliability. – 2007. – Vol. 56. – P. 223– 236.

6. Ma, Y. A Statistical Framework for the Prediction of Fault-Proneness [Text] /Y. Ma, L. Guo, B. Cukic // Conference on Machine Learning, Corvalis, Oregon, 2007. – P. 935–942.

7. Wang, Q. Extract rules from software quality prediction model based on neural network [Text] / Q. Wang, B. Yu, J. Zhu // Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2004), Boca Raton, Florida, USA, 15-17 Nov. 2004. – P. 191 – 195.

8. Kaminsky, K. How to Predict More with Less: Defect Prediction Using Machine Learners in an Implicitly Data Starved Domain [Text] /K. Kaminsky, G. Boetticher // The 8th World Multi-Conference on Systemics, Cybernetics and Informatics, Orlando, Fla., 2004. – P. 281 – 293.

9. Маевский, Д.А. Оценка количества дефектов программного обеспечения на основе метрик сложности [Текст] / Д.А. Маевский, С.А. Яремчук // Электротехнические и компьютерные системы. – 2012. – № 07 (83). – С. 113–120.

10. Логистика. [Электронный ресурс]. – Режим доступа: http://ru.wikipedia.org/wiki/Логистика. – 14.11.2012 г.

11. Маевский, Д.А. Соответствие метрик сложности и количества дефектов в программном обеспечении [Текст] / Д.А. Маевский, С.А. Яремчук. // Современные информационные и электронные технологии: сб. науч. трудов. 13-й межд. науч.-практ. конф., Одесса, 4-8 июня 2012 г. – С. 25.

12. The PROMISE Repository of empirical software engineering data [Электронный ресурс] / T. Menzies, B. Caglayan, E. Kocaguneli, J. Krall, F. Peters, B. Turhan. – Режим доступа: http://promisedata.googlecode.com. – 14.11.2012 г.

13. Chidamber, S. A Metrics Suite for Object-Oriented Design [Text] / S. Chidamber, C. Kemerer// IEEE Transactions on Software Engineering. – 1994. – vol. 20, No. 6. – P. 476–493.

14. Lorenz, M. Object-Oriented Software Metrics. A Practical Guide [Text] / M. Lorenz, J. Kidd. – Prentice Hall, Englewood Cliffs, New Jersey, USA, 1994. – 146 p.

15. Martin, R.C. Object Oriented Design Quality Metrics: An analysis of dependencies [Text] / R.C. Martin // Report on Object Analysis & Design. – 1995. – Vol. 2, No 3. – 68 p.

Поступила в редакцию 14.11.2012

Рецензент: д-р техн. наук, проф., зав. каф. компьютерных систем и сетей В.С. Харченко, Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Харьков

АПРІОРНА ОЦІНКА КІЛЬКОСТІ ДЕФЕКТІВ В ПРОГРАМНОМУ ЗАБЕЗПЕЧЕННІ ІНФОРМАЦІЙНИХ СИСТЕМ

Д.А. Маєвський, С.О. Яремчук

У роботі виконано аналіз існуючих моделей і методів оцінки кількості дефектів в програмному забезпеченні інформаційних систем. Запропонована апріорна логістична модель оцінки кількості дефектів на основі комплексних метрик складності. Сформульовані допущення моделі, зроблений вибір найбільш інформативних метрик і визначена їх мінімально необхідна кількість для оцінки дефектів об'єктно - орієнтованого програмного коду. Доведена працездатність досліджуваної моделі для програмних проектів різних розробників. Показано, що точність оцінки кількості дефектів залежить від міри виконання допущень. Запропонована чисельна характеристика виконання допущень - індекс відмінності змін і визначений спосіб його розрахунку. Запропонована модель може бути використана на етапі тестування програмного забезпечення для оцінки наявної кількості дефектів і планування на її основі процесу тестування.

Ключові слова: апріорна логістична модель надійності, комплексна програмна складність, метрики складності, програмні дефекти.

A PRIORI ESTIMATION OF DEFECTS AMOUNT IN SOFTWARE OF INFORMATION SYSTEMS

D.A. Maevsky, S. A. Yaremchuk

The article is devoted by the analysis of existing models and methods of an estimation of program defects. A priori logistic model of estimation of amount of defects offers on the basis of complex birth-certificates of complication. Assumptions of model are set forth, the choice of the most informing birth-certificates is done and their minimally necessary amount is determined for the estimation of defects in the object - oriented programs. The capacity of the investigated model is well-proven for the programmatic projects of different developers. It is shown that exactness of estimation of amount of defects depends on the degree of implementation of assumptions. Numeral description of implementation of assumptions offers is an index of distinction of changes and the method of his calculation is certain. An offer model can be used on the stage of testing of software for the estimation of present amount of defects and planning on her basis of process of testing.

Keywords: aprioristic logistical model of reliability, complex program complexity, complexity metrics, program defects.

Маевский Дмитрий Андреевич – канд. техн. наук, доцент, заведующий кафедрой теоретических основ и общей электротехники Одесского национального политехнического университета, г. Одесса, Украина, E-mail: Dmitry.A.Maevsky@gmail.com.

Яремчук Светлана Александровна – аспирант кафедры ТООЭ Одесского национального политехнического университета, Одесса, Украина, старший преподаватель кафедры информационных систем и технологий Измаильского института водного транспорта, г. Измаил, Одесская область, Украина, e-mail: svetlana397@yandex.ru