Зорин Даниил Александрович
Студент 4 курса, кафедра АСВК
Научный руководитель: Волканов Дмитрий Юрьевич
juan@lvk.cs.msu.su
8-906-703-85-43

Исследование применимости моделей оценки надёжности для разработки программного обеспечения с открытым исходным кодом

1. Введение

В настоящее время в мире разрабатывается большое количество программных продуктов с открытым исходным кодом (Open Source). В таких проектах участвует множество разработчиков, добавляющих свои фрагменты кода в общий репозиторий (хранилище, где находятся различные версии файлов, из которых состоит программа).

Проектом будем считать одну или несколько программ, созданием которых управляет одна команда разработчиков. Большой проект может состоять из подпроектов, реализующих отдельные части, работающие независимо друг от друга. Подпроекты могут состоять из компонент, которые выполняют определённый набор функций, и не используются отдельно. В проекте участвуют программисты, работающие над одним или несколькими подпроектами, тестировщики, и руководители проекта. Одна из основных задач руководителя проекта — определить, когда программу можно предоставлять конечным пользователям.

Основные особенности проектов с открытым кодом следующие:

- Команда тестировщиков непостоянна и работает не в установленные часы, а по мере своих возможностей. Отчёты об ошибках могут поступать в любое время суток (из-за того, что отправители могут находиться в разных городах мира).
- Не гарантируется, что количество времени и усилий, затраченных на тестирование, равномерно.
- Параллельно с тестированием продолжается разработка, поэтому количество ошибок может увеличиться.

В результате, перед руководителями проекта стоят следующие задачи:

- Оценить количество ошибок, оставшихся в проекте, и их критичность.
- Оценить количество ошибок, оставшихся в выбранном компоненте, и их критичность.
- Оценить время, когда текущая версия станет стабильной, то есть количество ошибок и их критичность будет меньше заданного порога.

Для решения этих задач можно использовать два подхода: первый основан на изучении исходного кода программы, второй — на использовании моделей оценки надёжности (МОН) (англ. software reliability growth model, SRGM) — на данных, полученных при тестировании [1]. В данной работе подробно рассматривается второй подход.

Основным результатом расчётов в этом случае является *функция количества ошибок* или *функция количества* — неубывающая функция времени, показывающая количество ошибок, найденных к заданному моменту времени. При получении этой функции возникают задачи, описанные в следующем разделе.

2. Постановка задачи

Введем следующие обозначения:

- *f(t)* – количество ошибок, найденных к заданному моменту: данные, полученные из статистики тестирования.
- *m(t)* – функция количества ошибок, полученная при применении какой-либо модели.

- $m(t)_x$ - функция количества ошибок, полученная в момент времени *x* (на основании статистики до момента *x*).

Изначально задана статистика тестирования: множество ошибок, для каждой из которых обязательно задано время обнаружения и компонент, где она была обнаружена.

Для получения функции *m(t)* из исходных данных результатов тестирования требуется решить следующие задачи:

1. Определить период времени, через который необходимо строить новую функцию *m(t)* для получения более точных оценок, учитывающих вновь поступившие данные.
2. Выбрать наиболее подходящую к данному проекту МОН из нескольких возможных.
3. Оценить, какой объем исходных данных необходим для того, чтобы модели давали результаты с точностью не ниже заданной.

В качестве критериев качества моделей предлагается использовать следующие:

- Сходимость: возможность построения *m(t)* по исходным данным результатов тестирования и выбранной МОН.
- Точность: среднеквадратичное отклонение полученной функции количества ошибок от реальной: $acc(t) = \frac{\sum_{t_i} \frac{m(t_i)}{f(t_i)}}{t} * 100\%$
- Устойчивость: оценка в текущий период мало отличается от оценки за предыдущий период: $st(t) = \left| \frac{m(\infty)_{t_n}}{m(\infty)_{t_{n-1}}} * 100\% - 100 \right| < k, k = 10$

В данной работе решаются задачи 1 - 3 для некоторых конкретных проектов с открытым исходным кодом.

3. Исследуемые модели оценки надёжности

Суть МОН заключается в том, чтобы построить математическую модель, описывающую процесс обнаружения ошибок. Для этого применяются различные вероятностные модели случайных процессов, при этом основной случайной величиной, распределение вероятностей которой необходимо найти, является либо количество ошибок в некоторый момент или за определённый период, либо время между двумя ошибками. Выбрав общий вид используемой модели (например, неоднородный пуассоновский процесс или цепь Маркова), можно различными статистическими методами оценить его параметры и в результате получить функцию, показывающую наиболее вероятное количество ошибок в некоторый момент, либо оценку для времени до обнаружения следующей ошибки. После этого можно сравнивать функции, полученные в результате применения различных моделей, между собой и выбирать наилучшую. На основании выбранной модели можно строить оценки различных величин, например, ожидаемое количество ошибок в ближайший месяц или общее количество оставшихся ошибок, на основании которых можно принимать решение о готовности программы к выпуску.

В данном исследовании были проанализированы наиболее подробно описанные в литературе и часто используемые модели оценки надежности [2]:

- Goel-Okumoto (GO) [1]
- S-Shaped (SS) [1]
- Logarithmic (Log) [4]
- Jelinski-Moranda (JM) [3]
- Littlewood-Verrall (LV) [5]

4.Алгоритм выбора модели оценки надёжности

В этом разделе предложен алгоритм, основанный на алгоритме из [6], позволяющий сравнивать несколько заданных МОН. Алгоритм состоит из следующих шагов:

1. На первом шаге рассчитываются параметры для нескольких моделей (методом максимального правдоподобия). После первого шага остаются только те модели, для которых метод сходится. Сходимость может отсутствовать из-за того, что модель в принципе не подходит.
2. Исследуется точность моделей. Для этого применяется минимизация среднеквадратичного отклонения.
3. Проверка на устойчивость (по формулам из раздела 2).
4. Если на предыдущем шаге не оказалось ни одной устойчивой модели, то необходимо продолжить тестирование и сбор данных ещё на один отчётный период. Если же есть устойчивые модели, то на основе их можно уже решать, является ли текущая версия стабильной.

Описанный алгоритм носит эмпирический характер: модель выбирается после некоторого периода, в течение которого производятся различные промежуточные расчёты, тогда как изначально по свойствам проекта никаких предположений не делается. Для того, чтобы проверить работоспособность алгоритма и адекватность выбранных характеристик точности, необходимо испытать модели и алгоритм на реальных данных. Для этих целей было создано программное средство, позволяющее делать расчёты, и с его помощью проведены эксперименты.

5. Экспериментальное исследование

Задача исследования — на данных тестирования реальных проектов исследовать различные описанные в литературе модели, и получить решение для задач 1-3 из раздела 2:

1. Определить период времени, через который необходимо строить новую функцию *m(t)* для получения более точных оценок, учитывающих вновь поступившие данные.
2. Выбрать наиболее подходящую к исследуемому проекту МОН из нескольких возможных.
3. Оценить, какой объем исходных данных необходим для того, чтобы модели давали результаты с точностью не ниже заданной.

Проекты выбирались исходя из следующих требований:

- Большой объем программы (сотни тысяч строк)
- Длительное время тестирования, обширная база данных о тестировании (несколько лет).
- Большое количество разработчиков (несколько сотен)
- Состоит из отдельных компонент.

Для практического исследования были выбраны несколько проектов с открытым исходным кодом. Так как модуль, собирающий статистику, ориентирован на систему Bugzilla [7], выбирались проекты, использующие эту систему для хранения статистики тестирования. В качестве основного был взят проект Eclipse [8], в качестве дополнительных — Mozilla [9] и Linux Kernel [10].

Методика исследования следующая:

1. По выбранным проектам из системы отслеживания ошибок загружается статистика тестирования.
2. Для этих проектов и, выборочно, их компонент и подмножеств ошибок одинаковой критичности, получены следующие данные:
   - Оценка количества ошибок на ближайший период и сравнение этой оценки с реальными данными (по периодам за некоторое время).

- Точность и устойчивость рассматриваемых сходящихся моделей, в соответствии с алгоритмом из раздела 4.

3. Используя эти данные, с помощью алгоритма из раздела 4 получается ответ на задачи 2 и 3. Оценивая точность полученных результатов на разных промежутках времени между применениями алгоритма выбора модели, можно получить ответ на задачу 1.

Эксперименты показали, что характеристики моделей Goel-Okumoto, Jelinski-Moranda и Logarithmic в целом довольно близкие. Проведенные расчеты свидетельствуют, что спустя некоторое время (те же 10-15 месяцев) разница между оценками по этим моделям оказывается не выше 5%. Минусом GO модели является то, что она иногда не сходится — постоянно применять только ее не всегда возможно. Модель JM можно считать наиболее устойчивой, но не самой точной, кроме того, она делает скорее заниженные оценки, тогда как из практических соображений более приемлемыми следует считать несколько завышенные.

Наиболее подходящей кривой для проектов с открытым исходным кодом, как можно было предположить, является модель S-Shaped, что хорошо видно на статистике по отдельным модулям. В более глобальном масштабе, на уровне целого проекта (количество ошибок за весь период имеет порядок нескольких десятков тысяч) S-Shaped модель работает плохо, что можно объяснить неравномерностью разработки по разным модулям.

6. Заключение

В рамках данной работы получены следующие результаты:

- Предложен алгоритм сравнения МОН.
- Реализовано программное средство для сравнения моделей и выбора наиболее подходящей.
- Показана общая эффективность МОН применительно к выбранным проектам с открытым кодом.
- Выбран наиболее подходящий промежуток времени для разбиения данных — один месяц.

Сформулированы следующие гипотезы о применении МОН к проектам с открытым исходным кодом:

- Приемлемая точность и устойчивость достигается через 10-15 месяцев после начала тестирования.
- Наиболее подходящей для проектов с открытым исходным кодом является модель S-Shaped.

В качестве дальнейшего развития данного подхода можно выделить следующие направления:

- Автоматизация выбора наилучшего периода для группировки данных (с возможностью выбора произвольного периода, а не только «круглого»).
- Определение периода, на котором оценки верны с точностью не ниже заданной.

Литература

1. M.Lyu Software Reliability Engineering: A Roadmap // 2007 Future of Software Engineering. Washington, DC, USA, 2007. p. 153 — 170.
2. Волканов Д.Ю., Зорин Д.А. Исследование применимости моделей оценки надёжности для разработки программного обеспечения с открытым исходным кодом // Программные системы и инструменты. Тематический сборник №10б М.: Изд-во факультета ВМиК МГУ, 2009. С.125-134.
3. S.Alam Software Reliability Using Markov Chain Usage Model. Dhaka, Bangladesh: Department of Computer Science and engineering Bangladesh University of Engineering and Technology, 2005
4. A.Wood Software Reliability Growth Models. Technical Report, 1996
5. M.Limnios, N.Nikulin Recent Advances in Reliability Theory. Birkhauser, 2000

6. C.Stringfellow An Empirical Method for Selecting Software Reliability Growth Models // Empirical Software Engineering 2002. 7. №4. p. 319 — 343.
7. Mozilla project. Bugzilla bug tracker [HTML] (http://www.bugzilla.org/about/)
8. Eclipse project. http://bugs.eclipse.org/bugs/
9. Mozilla Project. https://bugzilla.mozilla.org/
10. Linux Kernel project. http://bugzilla.kernel.org/