УДК 004.415.53

ИСПОЛЬЗОВАНИЕ ИНФОРМАЦИИ О СТОИМОСТИ ТЕСТОВ И СЕРЬЕЗНОСТИ ОШИБОК В ПРОЦЕССЕ ПРИОРИТЕЗАЦИИ ТЕСТОВ

А.Г. МАЛЫШЕВСКИЙ

Рассмотрено использование информации о стоимости тестов и серьезности ошибок в процессе приоритезации в регрессивном тестировании. Описаны способы оценки стоимости тестов и серьезности ошибок. Приведены новые методы приоритезации и метрика для оценки их эффективности. Исследовано применение описанных методов и метрики, а также влияние выбора стоимости тестов и серьезности ошибок на приоритезацию.

ВВЕДЕНИЕ

Одним из методов проверки, удовлетворяет ли программа заданным требованиям и ее спецификации, является тестирование (выполнение программы и проверка ее поведения на соответствие спецификациям). Каждый тест из набора тестов *T*, использующегося в тестировании, состоит из множества входных значений (сценариев тестирования). Обычно набор тестов создается, исходя из некоторого множества правил, называемого критерием адекватности. Этот критерий выражает условия, которым должен удовлетворять набор тестов [1]. На стадии сопровождения программы многократно используется регрессивное тестирование, проверяющее, что внесенные в программу модификации не только изменили программу в соответствии с новыми спецификациями и исправили найденные ошибки, но и не внесли новых ошибок [1].

В регрессивном тестировании растет набор тестов, увеличивая стоимость и продолжительность тестирования. Например, одна из систем ПО из 20 000 строк кода требует семь недель для тестирования при использовании всех тестов в наборе. Во многих случаях в процессе регрессивного тестирования можно использовать только подмножество набора тестов для проверки модифицированной программы. Но иногда бывает сложно или не допускается использование неполного набора тестов, например, для программ, надежность которых является критичной (авионика или управление медицинским оборудованием). В данном случае для уменьшения стоимости регрессивного тестирования может быть применен иной подход: тесты упорядочиваются (приоритезируются) для регрессивного тестирования таким образом, чтобы более важные из них выполнялись в первую очередь.

Вопросам приоритезации уделено много внимания [2–12]. В методах приоритезации тесты сортируют таким образом, чтобы эффективнее достичь заданной цели, например, наиболее быстрого покрытия операторов программного кода, функций программы в порядке частоты их использования или подсистем в порядке частоты их сбоев в прошлом. Возможная цель приоритезации — увеличение скорости выявления ошибок набором тестов в процессе тестирования. Возросшая скорость выявления ошибок может

обеспечить более раннюю обратную связь с регрессивно тестируемой системой и позволить разработчикам начать поиск местонахождения ошибок, а также их исправление раньше, чем это было бы возможно в ином случае. Такая обратная связь обеспечивает выявление ранних признаков того, что заданные цели еще не достигнуты, и позволяет принимать стратегические решения о сроках реализации на ранних этапах. Повышенная скорость обнаружения ошибок увеличивает вероятность того, что в случае преждевременного прекращения процесса тестирования тесты, обеспечивающие наибольшую способность выявлять ошибки в сроки, выделенные на тестирование, уже были выполнены.

В работах [4–6, 8, 12] представлена метрика APFD, которая определяет скорость выявления ошибок во время выполнения набора тестов в заданном порядке, и показано, что она может быть использована для оценки скорости выявления ошибок в наборах тестов числовыми значениями и их последующего сравнения. В этих работах описано несколько методов приоритезации для увеличения скорости выявления ошибок в регрессивном тестировании и эмпирически оценена их эффективность. Результаты оценки показали, что несколько методов могут улучшить значения APFD наиболее простыми (и дешевыми) методами.

Несмотря на то, что разработанные ранее методы приоритезации и метрика APFD успешно применялись, в них содержалось допущение о том, что не только стоимость каждого теста одинакова, но и все ошибки одинаково серьезны. (В работе [3] кратко описывается метод приоритезации, в котором учитывается информация о стоимости тестов.) Такое допущение может быть приемлемо. В некоторых случаях — это чрезмерное упрощение [13,14]. Какие-то тесты просто могут обнаружить ошибку в исходных данных и немедленно прекратить выполнение программы, другие же — выполнить долгие многочасовые вычисления. Аналогично в некоторых случаях один тест требует значительных ресурсов (оборудования, расходных материалов или времени программистов), тогда как другой не требует ничего, кроме компьютерного времени. По иному сценарию выполнение всех тестов может быть коротким, но затраты на проверку результатов работы программы значительно различаться. Таким образом, оценивая сравнительную ценность тестов, необходимо учесть эти различия в их стоимости. Как и в случае с тестами, ошибки могут быть различными по серьезности. Незаметная для пользователей грамматическая ошибка в интерфейсе программы может привести к неправильному функционированию управляемого устройства, что, в свою очередь, привести к катастрофе. Известными примерами являются потеря космических аппаратов для исследования Марса (Mars Polar Lander, Mars Climate Orbiter) и ракетоносителя Ariane, а также получение смертельной дозы радиации на аппарате лучевой терапии Therac-25. Серьезность ошибок также может быть важным компонентом ценности выявляющего их теста.

На практике стоимость тестов и серьезность ошибок могут значительно различаться, а методы, разработанные для улучшения очередности выполнения тестов и сама метрика APFD могут не дать удовлетворительных результатов. Поэтому в данной статье рассматриваются не только новые методы приоритезации, учитывающие как стоимость тестов, так и серьезность

ошибок, но и новая обобщенная метрика для измерения скорости их выявления ~~ошибок~~, учитывающая различающиеся стоимости тестов и серьезности ошибок, а также приводятся результаты применения этих методов для разных распределений стоимости тестов и серьезности ошибок.

Прежде всего для анализа приоритезации необходимо оценить количественно ее эффективность.

МЕТРИКА APFD

В работах [4–6, 12] использовалась метрика APFD (weighted average of the percentage of faults detected), оценивающая скорость выявления ошибок набором тестов в интервале от 0 до 100. Чем больше значение метрики, тем быстрее выявляются ошибки. Однако данная ~~Ограниченность Метрики APFD~~

~~М~~метрика ~~APFD~~ основывается на двух допущениях: 1) все ошибки идентичны по серьезности и 2) все тесты идентичны по стоимости. Эти допущения выражаются в том, что данная метрика просто определяет процент выявленных ошибок для выполненной части набора тестов. Следующие примеры поясняют проблемы с этими двумя допущениями.

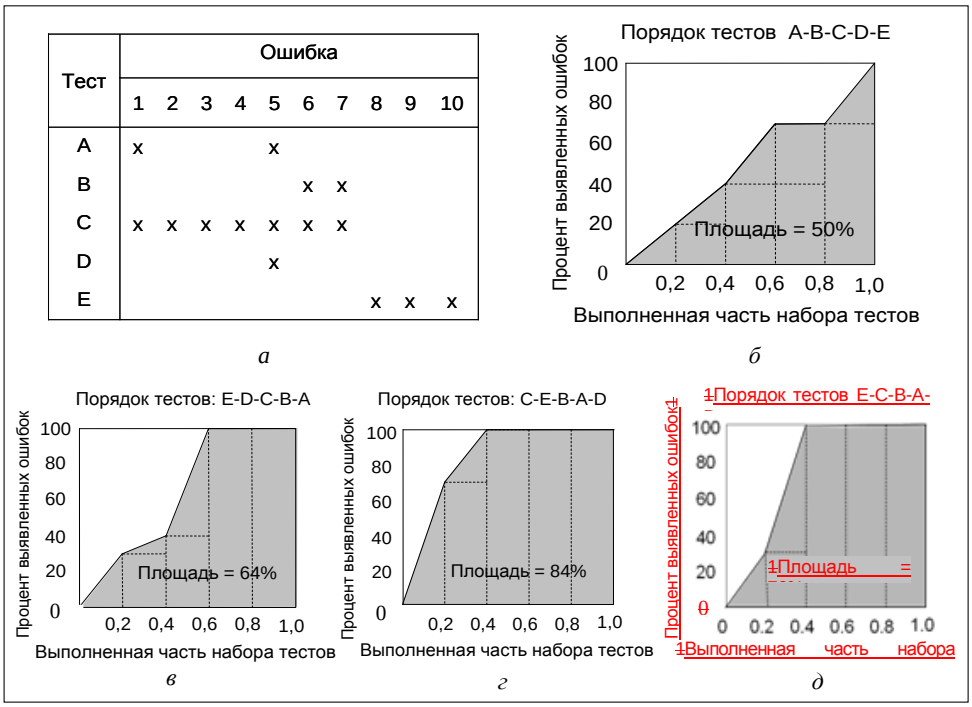

Рис. 1. Примеры, иллюстрирующие метрику APFD: *а* — тесты и выявленные ими ошибки; *б* — APFD для приоритезированного набора тестов T1; *в* — ~~APFD~~ для ~~приоритезированного набора тестов~~ T2; *г* — ~~APFD~~ для ~~приоритезированного набора тестов~~ T3; *д* — ~~APFD для приоритезированного набора тестов~~ T4

Пример 1. Рассмотрим сценарий, показанный на рис. 1. В метрике APFD, когда все десять ошибок одинаково серьезны и все пять тестов равны по стоимости, порядки A–B–C–D–E и B–A–C–D–E являются эквивалентными с точки зрения скорости выявления ошибок. Т.е., если поменять

местами тесты A и B, скорость выявления ошибок не изменится. Эта равноценность отражается в эквивалентных значениях метрики APFD (50%). (Значение метрики соответствует площади, ограниченной кривой.) Допустим, что стоимость теста B в два раза превосходит стоимость теста A, требующего два часа машинного времени, тогда как тест A — один час. С точки зрения ошибок, выявленных за час, порядок тестов A–B–C–D–E предпочтительней порядка B–A–C–D–E, который выявляет ошибки быстрее. Однако метрика APFD считает эти два порядка равноценными.

Пример 2. Работая опять со сценарием, показанным на рис. 1, предположим, что все пять тестов имеют равную стоимость и что ошибки 2…10 имеют значение серьезности, равное k, тогда как ошибка 1 имеет серьезность $2k$. В данном случае тест A выявляет одну более серьезную ошибку и одну менее серьезную, тогда как тест B — только две менее серьезные ошибки. С точки зрения скорости выявления суммарной серьезности ошибок порядок тестов A–B–C–D–E предпочтительней порядка B–A–C–D–E. Но снова метрика APFD оценивает эти два порядка одинаково.

Пример 3. Примеры 1 и 2 демонстрируют ситуации, в которых метрика APFD оценивает два порядка тестов, как эквивалентные, а интуиция показывает, что они не должны быть таковыми. Также возможна ситуация, когда при неодинаковой стоимости тестов или серьезности ошибок, метрика APFD дает более высокую оценку худшему порядку тестов. Предположим (рис. 1), что все десять ошибок равнозначны по серьезности и что каждый из тестов A, B, D и E требует один час для выполнения, но тест C — десять часов. Метрика APFD порядку тестов C–E–B–A–D ~~order~~ присвоила значение APFD 84% (рис. 1, *г*). Рассмотрим иной порядок ~~test case order~~ E–C–B–A–D (рис. 1, *д*). Так как данная метрика не дифференцирует тесты согласно их стоимости, то все вертикальные столбики на графике (индивидуальные тесты) имеют одинаковую ширину. Значение APFD для данного порядка 76% ниже значения для порядка C–E–B–A–D. Однако с точки зрения ошибок, выявленных за единицу времени, второй порядок (E–C–B–A–D) предпочтительнее: он выявляет три ошибки в течение первого часа и остается лучшим, чем первый порядок, до конца выполнения второго теста. Аналогичный пример может быть приведен для использования ошибки с неравной серьезностью при равной стоимости тестов.

НОВАЯ ~~COST-COGNIZANT~~ МЕТРИКА $APFD_C$

~~Примеры и Мотивация~~

~~The я~~Примеры подсказывают, что метрика, которая предполагает равную стоимость тестов и равную серьезность ошибок, может давать неудовлетворительные результаты. Важно понимать: существует компромисс между стоимостью тестов и стоимостью невыявленных в программе ошибок. Поэтому~~,~~ следует учесть этот компромисс в процессе приоритезации тестов. Метрика для оценивания порядков тестов должна содержать факторы, лежащие в основе компромисса. В данной работе такая метрика оценивает порядки тестов, пропорционально скорости выявления единиц серьезности

выявленных ошибок на единицу стоимости тестов. Автором создана такая метрика (адаптированная APFD). Назовем ее $APFD_C$.

Создание новой метрики требует двух изменений (рис. 1). Во-первых, на горизонтальной оси на графике заменим «Выполненная часть набора тестов» на «Процент суммарной затраченной стоимости тестов». Теперь каждый тест в наборе представлен интервалом вдоль горизонтальной оси, и длина его пропорциональна доли стоимости данного теста в суммарной стоимости тестов в наборе. Во-вторых, на вертикальной оси графика заменим «Процент выявленных ошибок» на «Процент суммарной серьезности выявленных ошибок». Теперь каждая ошибка, выявленная набором тестов, представлена интервалом вдоль вертикальной оси, и высота его пропорциональна доли ее серьезности в общей сумме серьезности ошибок. Здесь стоимость теста и серьезность ошибки могут быть интерпретированы и измерены по-разному. Если время выполнения теста (подготовки, самого выполнения и проверки результатов) является основной составляющей стоимости теста, то оно может быть достаточным для ее измерения. Однако стоимость теста может также базироваться на таких факторах, как стоимость оборудования и зарплата персонала. Аналогично серьезность ошибки также может быть измерена соответственно времени, необходимому для выявления и исправления ошибки, либо можно учесть стоимость потери бизнеса, судебных исков или ущерб, причиненный людям или собственности, и т. д. В любом случае метрика $APFD_C$ позволяет учесть такие интерпретации. (Заметим, что в метрике $APFD_C$ мы не пытаемся предсказать стоимость тестов и ошибок, что может быть достаточно сложно, а пытаемся лишь измерить их постфактум для оценки различных порядков тестов.)

С учетом этой новой интерпретации на графиках вклад теста взвешивается в горизонтальном направлении по его стоимости и вдоль вертикального направления по суммарной серьезности выявленных им ошибок. В таких графиках кривая ограничивает большую площадь для порядка тестов, который демонстрирует больше единиц серьезности ошибок, выявленных на единицу стоимости теста. Эта площадь и составляет нашу новую метрику $APFD_C$.

На рис. 2 показаны графики для каждого из трех примеров, описанных выше. Пара графиков, расположенная слева (рис. 2,*а*), соответствует примеру 1: верхний график — метрика $APFD_C$ для порядка тестов ~~test case order~~ A–B–C–D–E, нижний — $APFD_C$ для порядка ~~order~~ B–A–C–D–E. Отметим, что оригинальная метрика APFD не различила бы эти два порядка~~orders~~, а $APFD_C$ отдает предпочтение порядку~~faster detecting order,~~ A–B–C–D–E, который быстрее выявляет ошибки. Другие пары графиков иллюстрируют применение метрики $APFD_C$ в примерах 2 и 3. Пара графиков на рис. 2, *б*, соответствующая примеру 2, показывает, что новая метрика дает более высокую оценку порядку тестов, который раньше выявляет более серьезную ошибку (A–B–C–D–E), при допущении, что ошибкам 2…10 присвоено значение серьезности 1 и ошибке 1 — значение серьезности 2. Пара графиков на рис. 2, *в*, соответствующая примеру 3, показывает, что

новая метрика различает порядки тестов, где ~~test case orders~~ тест C имеет высокую стоимость: вместо недооценивания порядка~~order~~ E–C–B–A–D метрика теперь присва~~ей~~ивает е~~й~~му большее значение, чем порядку~~order~~ C–E–B–A–D.

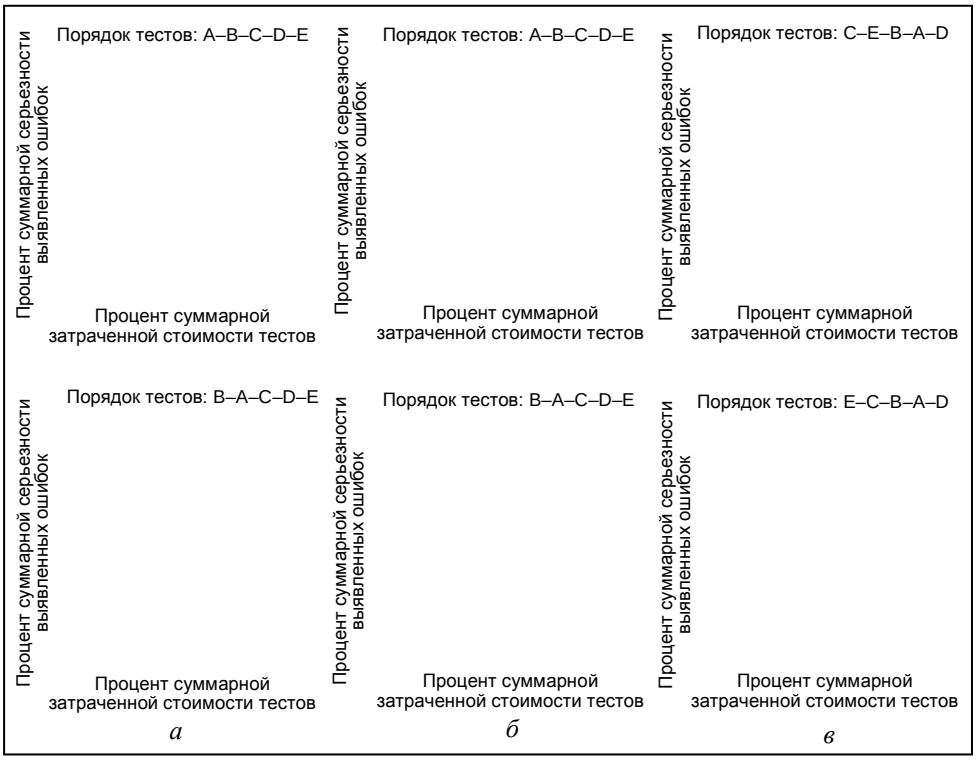

Рис. 2. Примеры, иллюстрирующие метрику APFD_C: *а* — для примера 1; *б* — для примера 2; *в* — для примера 1

Формула для новой метрики

Пусть T будет набором, содержащим n тестов со стоимостями $t_1, t_2, \ldots, t_n$; F — множеством m ошибок, выявленных набором тестов T; $f_1, f_2, \ldots, f_n$ — значениями серьезности этих ошибок. Пусть TF_i будет первым тестом в порядке T' набора T, который выявляет ошибку i. Тогда формула для метрики APFD_C будет иметь следующий вид [15]:

$$\mathrm{APFD}_C = \frac{\sum_{i=1}^{m}\left(f_i \left(\sum_{j=TF_i}^{n} t_j - \frac{1}{2} t_{TF_i} \right) \right)}{\sum_{i=1}^{n} t_i \sum_{i=1}^{m} f_i}.$$

ОЦЕНКА СТОИМОСТИ ТЕСТОВ

Существует две задачи, связанные со стоимостью тестов: 1) измерение или оценка стоимости с целью вычисления значения метрики $APFD_C$ для порядка тестов и 2) оценивание стоимости для использования при приоритезации тестов. Стоимость теста связана с ресурсами, затраченными на его выполнение и проверку результатов. Возможны различные объективные метрики, например: когда основную часть стоимости составляет машинное время или время персонала, стоимость теста может измерять фактическое время, затраченное на тестирование программы заданным тестом. Другая метрика учитывает денежные затраты на выполнение теста и проверку результатов. Она может отражать амортизированнуюamortized стоимость оборудования, зарплату, стоимость материалов для тестирования, потери дохода от задержки выпуска ПО, от срыва сроков релизации и т. д.

Определить стоимость тестов относительно просто после тестирования, что приемлемо для метрики $APFD_C$. Для этого следует знать, observe какие ресурсы затрачены на каждый тест. Однако гораздо сложнее оценить стоимость перед началом тестирования, а это неужно для использования ее в процессе приоритезации тестов, т.е. необходимо предугадать predict стоимость тестов. Одним из подходов является анализ теста и программного кода, выполненного тестом. Другим подходом, который используется в этой работе, является использование данных о стоимости тестов на предыдущих сессиях тестирования (что представляется возможным при регрессивном тестировании). Полагаем, что стоимость тестов значительно не меняется от одной версии программы к другой.

ОЦЕНКА ESTIMATING SEVERITYСЕРЬЕЗНОСТИ ОШИБОК

Как и со стоимостью тестов, существует два подхода к серьезности ошибок: 1) их измерение или оценка в метрике $APFD_C$ для определения порядка тестов и 2) оценка ошибок для использования информации при приоритезации тестов. Серьезность ошибки связана с понесенными затратами, если она осталась в программе после реализации. Возможны различные подходы к измерению такой серьезности.

- Измерение серьезности ошибки как суммы средств, потерянных в результате сбоя, вызванного данной ошибкой (с учетом его вероятности). Такой подход можно применять в ПО, где сбой приводит к катастрофическим последствиям, например, человеческим жертвам, судебным искам, потере оборудования.
- Оценка влияния ошибки на надежность ПО. Применяется к ПО (например, текстовый редактор для ПК), где ошибки вызывают всего лишь неудобство для пользователей, и маловероятно, что сбой будет иметь серьезные последствия (например, снижение надежности ПО, приводящее к потере клиентов).

Аналогично ситуации со стоимостью тестов, нас интересует как оценка серьезности ошибок до их обнаружения, так и оценка их серьезности после обнаружения. Когда тестирование закончено и уже есть информация об

ошибках, можно оценить их серьезность для использования в метрике $APFD_C$ (хотя это не так просто, как со стоимостью тестов). С другой стороны, перед началом тестирования нам необходимо оценить серьезность потенциальных ошибок для использования данной информации в процессе приоритезации, а это намного сложнее, чем оценка стоимости тестов. Таким образом, требуется метрика, связывающая тесты с серьезностью выявленных ошибок.

Если бы мы знали, какие ошибки выявляет каждый тест и их серьезность, было бы несложно связать тесты с серьезностью ошибок. Однако на практике эта информация недоступна до завершения тестирования. Существует два подхода к такой оценке: 1) оценка критичности модулей и 2) оценка критичности тестов. При оценивании критичности модуля необходимо связать ~~severity~~серьезность ошибки с критичностью модуля (или любого другого компонента программного кода, например, основного~~basic~~ блока, функции, файла или объекта), в котором данная ошибка может содержаться. При оценивании критичности теста необходимо связать тесты

с ~~severity~~серьезностью ошибок, которые они могут выявить напрямую. Оценив критичность модулей или тестов, мы надеемся включить ~~incorporate~~ серьезность ошибок в процесс при~~о~~рит~~ие~~зации тестов, прежде чем начнется сам процесс тестирования. В данной работе используется оценка критичности модулей.

ПРИМЕР ИСПОЛЬЗОВАНИЯ ИНФОРМАЦИИ О СТОИМОСТИ ТЕСТОВ И СЕРЬЕЗНОСТИ ОШИБОК

~~Введение~~

Для практического применения предложенной метрики и некоторых модификаций ~~ramifications~~ ее использования нами проведено исследование, цель которого изучить, ~~investigate~~ как различные распределения ~~distributions~~ стоимости тестов и ~~severity~~серьезности ошибок могут влиять на скорость их выявления измеряемой метрикой $APFD_C$.

Понятие критичности модуля применялось для оценки стоимости ошибок в приоритезации.~~The study~~ Рассмотрено ~~examined~~ влияние различных распределений ~~distributions~~ стоимости тестов, ~~severity~~серьезности ошибок и их комбинаций на относительную эффективность методов приоритезации.

Объект использования

Как объект данного исследования использовалась программа *space*, разработанная Европейским космическим агентством и состоящая из 6218 строк кода (*space* — это интерпретатор языка для задания конфигурации массива ADL), а также 50 наборов тестов, адекватных покрытию ветвлений. Создано и применено 29 версий данной программы с некоторым количеством ошибок в каждой [15]~~.~~.

Методы п~~П~~риоритезации

Выбраны следующие методы приоритезации [14, 15]:

• **fn-cov-ccmult-fb**– сортирует тесты по дополнительному покрытию критичности функций. Другими словами, ценность тестов вычисляется как сумма критичности покрытых ими функций (но при этом еще покрытых ранее упорядоченными тестами). Если более чем один тест имеет наибольшую сумму, то тест, покрывающий наибольшее количество непокрытых функций, считается лучшим.

• **st-cov-ccmult-fb**ые ещм ссьгде аиЖ похож на ые ещм ссьгде аи **fn-cov-ccmult-fb**, но вместо покрытия по функциям используется покрытие по операторам (критичность операторов равна критичности содержащих их функций).

• \techfnficovccmultfb\ (**fn-fi-cov-ccmult-fb** похож на **fn-cov-ccmult-fb**, но вместо суммирования критичности покрытых функций каждое слагаемое еще умножается на индекс ошибки [16], аппроксимирующий уровень склонности функции к содержанию ошибок [17, 18].),

• **а также random** метод упорядочивает тесты случайным образом.

Распределения стоимости тестов

Мы случайным образом присвоили стоимость тестам соответственно пяти распределениям.

1. **Unit**. Стоимость каждого теста равна единице, что соответствует ситуации, в которой стоимость тестов не учитывается.

2. **Random**. Стоимость тестов равномерно распределена на интервале от 1 до 10.

3. **Normal**. Стоимость тестов нормально распределена с $\mu = 5$ $\mu = 5$ и $\sigma = 5$ $\sigma = 5$, но ограничена интервалом [1, 10].

4. **Mozilla**. Распределение стоимости тестов соответствует ее распределению по четырем категориям в программе Mozilla (табл. 1Таблицу \ref{table_cost_cognizance_mozilla}).(Mozilla — это иИнтернет-браузер с открытым кодом. См. www.mozilla.org. и budzilla. mozilla.org.)

5. **QTB**. Распределение стоимости тестов соответствует ее распределению по двум категориям в программе QTB (табл. 2Таблицу \ref{table_cost_cognizance_stk}) [19].

Таблица 1. {table_cost_cognizance_mozilla}Распределение стоимости тестов в программе Mozilla

NameНазвание	Уровеньlevel	Описаниеdescription	ПроцентPercentage
HTML	1	Наименьшая стоимостьLeast expensive	87
Printing	2	Большая — // — //more expensive	1
Smoke tests	3	Высокая — //— //expensive	2
Buster	4	Наибольшая — //— //most expensive	10

Таблица 2. \label{table_cost_cognizance_stk}Распределение стоимости тес-

тов в программе QTB

Уровень~~level~~	Описание~~description~~	Процент~~Percentage~~
1	Низкая стоимость~~not expensive~~	88
10	Высокая —//— //~~expensive~~	12

Для того чтобы задействовать ~~apply~~ каждо~~гое~~ распределени~~яе~~ стоимости тестов (кроме **unit**) сгенерировано множество стоимостей, элементы которых были случайным образом присвоены тестам.

Распределение с~~Severity~~серьезности ~~О~~ошибок

Мы использовали три следующих распределения ~~severity~~серьезности ошибок:

1. **Unit**. Все ошибки имеют серьезность, ~~severities~~ равную~~ые~~ единице, что соответствует случаю, в котором серьезность ~~severities~~ ошибок не учитывается.

2. ~~**Линейное**~~ **Mozilla-lin.** ~~severities~~ Соответствует распределению в программе Mozilla (табл. 3~~Таблицу \ref{table_cost_cognizance_mozillafaults}~~) по шести уровням. Значения серьезности присвоены по линейной шкале от 1 до 6.

3. ~~**Экспоненциальное**~~ **Mozilla-exp.** По~~д~~обно ~~**линейному**~~ **Mozilla-lin**, но в нем значения серьезности присвоены по экспоненциальной шкале от 2^0 до 2^5.

Таблица 3. ~~label{table_cost_cognizance_mozillafaults}~~Распределение серьезности~~Severity~~ ошибок в ~~П~~программе Mozilla

Уровень по линейной шкале	Уровень по экспоненциальной шкале	Серьезность	Процент
1	1	Тривиальная	2
2	2	Мелкая	11
3	4	Средняя	6
4	8	Крупная	76
5	16	Критическая	4
6	32	Блокирующая	2

Использование ~~Applying~~ распределения серьезности ошибок **unit** ~~severity ошибок~~ тривиально по сравнению с ~~applying два~~ распределениями **Mozilla-lin** и **Mozilla-exp** ~~труднее~~. Сложность состояла в том, что наши методы приоритезации содержали информацию о критичности модулей, но не имели никаких исторических данных для оценки ~~estimation~~ критичности модулей. Таким образом, требовалось сгенерировать как критичность модулей, так и ~~severity~~серьезность ошибок. Если присвоить критичность модулям и серьезность ~~severities~~ ошибкам независимо, то взаимосвязь м~~ас~~жду ~~severities~~ ними не будет отражена. Существование такой взаимосвязи является необходимым требованием ~~prerequisite~~ для методов приоритезации, которые используют критичность модулей ~~для~~ в предсказани~~ия~~ ~~severity~~серьезности ошибок. Вместо этого в н~~в~~ашем подходе мы до-

пустили, что существует корреляция между критичностью модулей и серьезностью содержащихся в них severities ошибок. Затем, полагаясь на это допущение, сгенерировали значения критичности модулей и severityсерьезности ошибок. Для использования каждого распределения severityсерьезности ошибок (кроме распределения **unit**) сгенерировано множество значений критичности для каждого заданного распределения, и случайным образом эти значения были randomly присвоены модулям, после чего мы considered каждойую ошибкуе f ибыло присвоено значение severityсерьезности, равное критичности содержащего ее модуля.

Такой подход не позволяет анализировать и объективно сравнивать методы приоритизации fairly, если, конечно, unless наши исследования не ограничены conditional следующей гипотезой: существует значительнаяclose корреляция между критичностью модуля и severityсерьезностью содержащейся в нем ошибкин. Однако данная работа направлена не на оценку эффективности performance методов приоритезации, а на оценку влияния распределений серьезности ошибок severityна значения метрики $APFD_C$.Поэтому, представляя результаты, мы condition our conclusions to reflect this methodology.

Комбинации рРаспределений Сстоимости Ттестов и серьезности Severity Ооошибок

При пяти различных распределениях стоимости тестов и трех распределениях severityсерьезности ошибок можно создать пятнадцать комбинаций. Однако ограничимся девятью наиболее интересными (табл. 4, «X» указывает Indicate на Ррассмотренные в Иисследованиях Ккомбинации\ref{table_cost_cognizance_combinations}).

Entries }**Таблица 4.** Комбинации \label{table_cost_cognizance_combinations}распределения Severityсерьезности Ооошибок (слева) и versus РСстоимости Ттестов (сверху)

	Unit	Random	Normal	Mozilla	QTB
Unit	X	X	X	X	X
Mozilla-lin	X			X	
Mozilla-exp	X			X	

Результаты Иисследованийд

Сгруппируем результаты исследований в три этапа. Сначала проанализируем влияние распределений стоимости тестов, используя различные методы приоритезации, при unit распределении severityсерьезности ошибок **unit**. Затем effects of — влияние распределений severityсерьезности ошибок, используя различные методы приоритезации, при unit распределении стоимости тестов **unit**. В конце проанализируем последствия комбинаций effects of распределений для стоимости тестов и severityсерьезности ошибок вместе.

Варьирование Рраспределения Сстоимости тестов

На рис. 3~~Рисунок \ref{figure_cost_cognizance_study1_resfig1}~~ показаны значения $APFD_C$ для различных распределений стоимости тестов. Здесь видны пять групп столбиков — одна соответствует значениям $APFD_C$, усредненным для всех методов приоритезации (слева), а четыре группы — усредненным значениям $APFD_C$, по одной группе на каждый метод приоритезации. Группа содержит пять индивидуальных~~individual~~ столбиков — по одному на распределение. Высота столбика определяет ~~denotes~~ среднее значение $APFD_C$, измеренное для наборов тестов, приоритезированных соответствующим методом и соответственно заданному распределению [13].

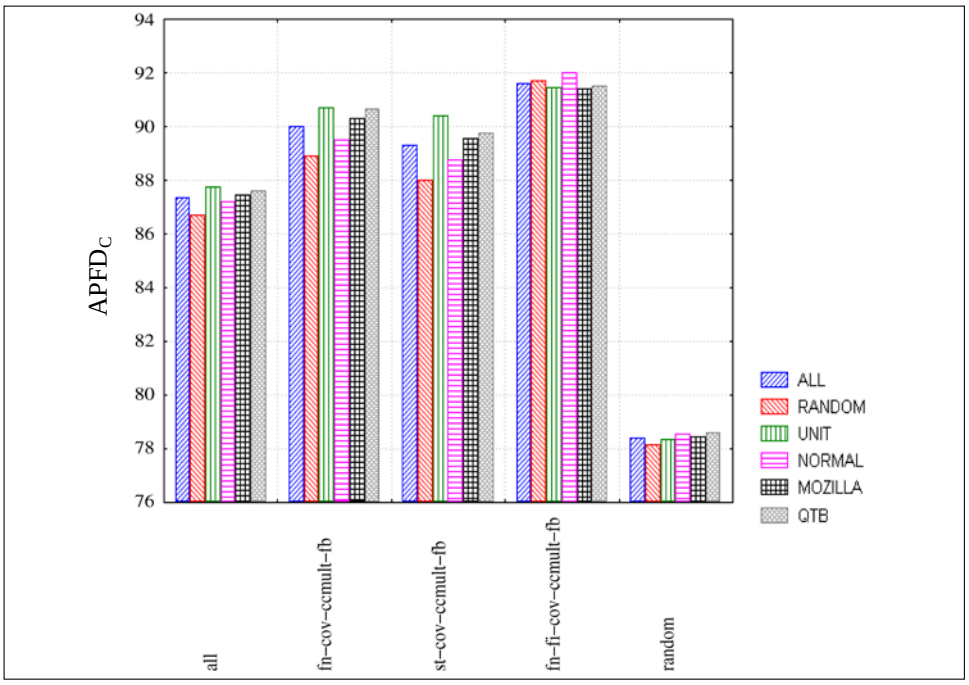

Рис. 3. Средние значения $APFD_C$ для ~~per~~ каждого распределени~~е~~я и каждого ~~, per~~ метод~~а~~ приоритезации

Как видно из рис. 3, распределение стоимости тестов влия~~ет~~ ~~impact~~ на скорость выявления ошибок приоритезированного набора тестов в соответствии с метрикой $APFD_C$ для всех методов приоритезации собранных вместе (группа столбиков слева). Эти различия были статистически значимыми, однако не оказались такими большими, как ожидалось: средние значения $APFD_C$ для различных распределений отличались не более чем на один процент. Также видно (внутри каждой из ~~within each of the~~ четырех групп ~~four rightmost sets~~ столбиков справа), что степень влияния~~extent of the impact~~ для разных методов был~~а~~ неодинаков~~а~~. Например, для метода **st-cov-ccmult-fb** различия между средними значениями $APFD_C$ для разных распределений были статистически значимыми, тогда как для метода **fn-fi-cov-ccmult-fb** — нет.

Анализ был выполнен для средних значений $APFD_C$. Исследование же индивидуальных различий в значениях $APFD_C$ показывает ~~yields~~ иную картину. Графики на рис. 4~~Рисунке \ref{figure_cost_cognizance_study1_resfig2}~~ дают абсолютную разницу в значениях $APFD_C$ приоритезированных наборов тестов с ~~unit~~ распределением стоимости тестов **unit** и приоритезированных наборов тестов с остальными ~~unit~~ четырьмя распределениями стоимости тестов (графики от A до D, соответственно). На каждом графике горизонтальная ось содержит 2000 наблюдений $APFD_C$ — одно на каждый из 50 приоритезированных наборов тестов для каждой из десяти версий и для каждого из четырех методов приоритезации. Наблюдения отсортированы по методам в следующем порядке: **fn-cov-ccmult-fb**, **st-cov-ccmult-fb**, **fn-fi-cov-ccmult-fb**, **random**, а затем по набору тестов и версиям. (Сплошные вертикальные линии на графиках разделяют наблюдения по четырем мето-

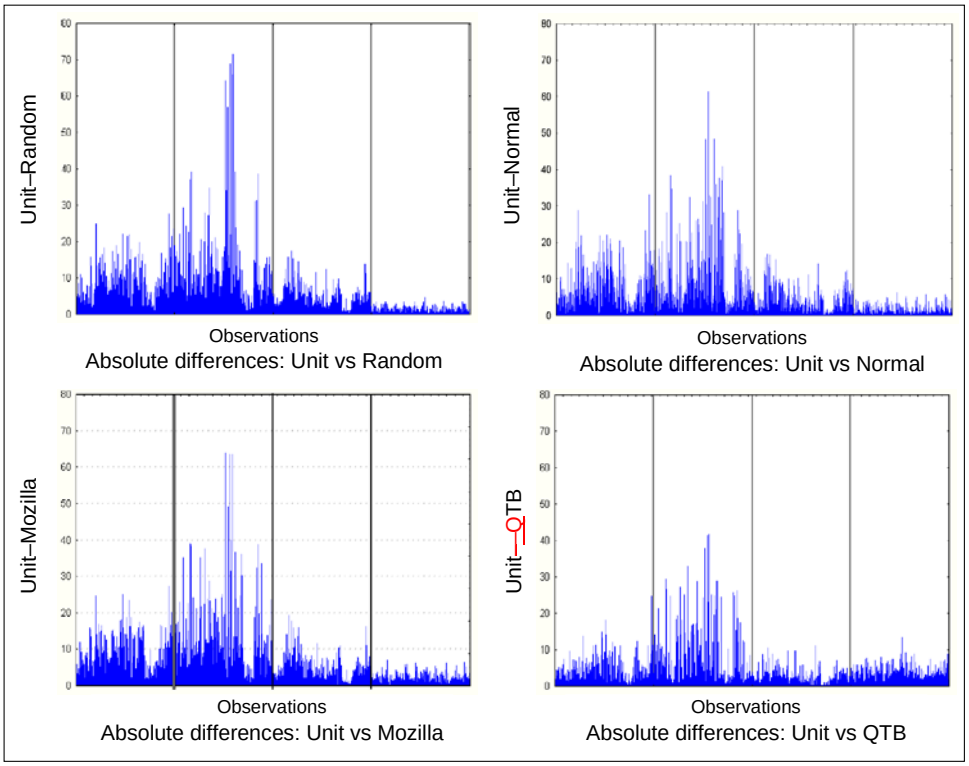

Рис. 4. Абсолютные различия в значениях $APFD_C$ по всем наблюдениям между распределением **unit** и каждым из четырех оставшихся распределений стоимости тестов

дам.)

На рис. 4~~Рисунок \ref{figure_cost_cognizance_study1_resfig2}~~ видно, насколько для индивидуальных приоритезированных наборов тестов значение $APFD_C$ для распределения стоимости тестов **unit** отличается от значений $APFD_C$ каждого другого распределения. Во многих случаях разница в значениях $APFD_C$ превышает 20%, а в некоторых — 50%. Это дает дополнительные аргументы в поддержку необходимости использования распределения стоимости тестов как составной части метрики $APFD_C$ (в против-

ном случае это может привести к неэффективной приоритезации). Кроме того, при ~~unit~~ распределении стоимости тестов **unit** значения $APFD_C$ эквивалентны значениям оригинальной метрики APFD, что показывает величину различий метрик APFD и $APFD_C$. ~~Рисунок \ref{figure_cost_cognizance_study1_resfig2}~~ При использовании методов **fn-cov-ccmult-fb** и **st-cov-ccmult-fb** распределения стоимости тестов ~~exhibited~~ более непостоянны в значениях $APFD_C$, чем при других методах (**st-cov-ccmult-fb** проявил наибольшее непостоянство). Отсюда следует, что для некоторых методов поведение распределений более предсказуемо, ~~predictable~~ чем для других.

В~~Рисунку \ref{figure_cost_cognizance_study1_resfig1}~~ методе **random** в среднем для каждого распределения стоимости тестов все методы приоритезации дали ~~provided~~ значительные улучшения в оценках значений $APFD_C$ (см. рис. 3). Таким образом, независимо от распределения стоимости тестов, приоритезация улучшила скорость выявления ошибок~~rate of fault detection~~.

~~Varying Распределение Severity Ошибок~~

Варьирование распределения серьезности ошибок

Проведенный анализ распределений ~~severity~~серьезности ошибок показывает, что распределение имеет значительное влияние на значения $APFD_C$ для всех методов приоритезации (рис. 5)~~Рисунком \ref{figure_cost_cognizance_study1_resfig3}~~. На рис. 5 изображены три группы диаграмм размаха (по одной на метод). В группе содержится одна диаграмма размаха для каждого из трех распределений ~~severity~~серьезности ошибок. Индивидуальная ~~individual~~ диаграмма размаха показывает распределение значений $APFD_C$ для всех наборов тестов, приоритезированных со-

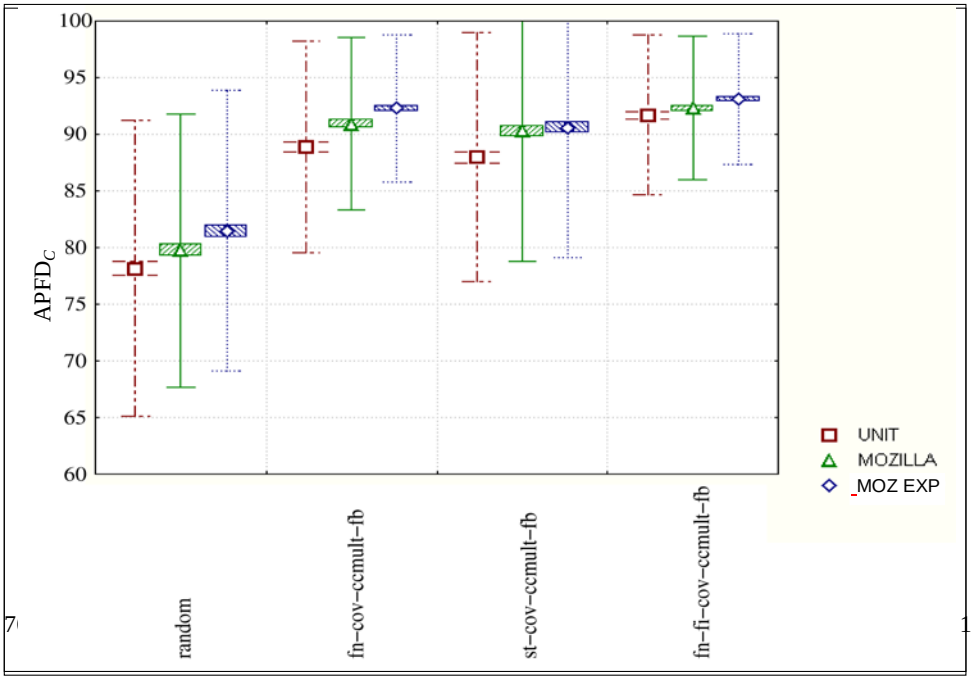

Рис. 5. Распределения значений ~~icse01/newFigs/fs.enc.eps \caption{\label{figure_cost_cognizance_study1_resfig3}~~$APFD_C$ (по одному на рас-

ответствующим методом и соответствующим распределением ~~severity~~серьезности ошибок.

Как видно из рис. 5, **fn-fi-cov-ccmult-fb** показал наиболее стабильное поведение среди всех распределений, а также ~~exhibited~~ более высокую эффективность приоритезации. Однако метод **r~~R~~andom** наиболее склонен к вариациям распределений и показал наихудшие результаты.

Варьирование распределения стоимости тестов и серьезности ошибок

Проанализировав влияние вариации распределений стоимости тестов и ~~severity~~серьезности ошибок в отдельности, рассмотрим результаты, полученные варьированием обеих распределений одновременно.

Для каждой из интересующей нас комбинации распределений применим метод **st-cov-ccmult-fb** и представим значения $APFD_C$ лишь 50 случайным образом выбранных наблюдений (из 500).

На рис. 6~~Рисунок \ref{figure_cost_cognizance_study1_resfig4}~~ показана диаграмма рассеяния для представления трех комбинированных распределений: 1) стоимость тестов **unit** и ~~severity~~серьезность ошибок **unit**, 2) стоимость тестов **Mozilla-lin** и ~~severity~~серьезность ошибок **Mozilla**, 3) стоимость тестов **Mozilla-exp** и ~~severity~~серьезность ошибок **Mozilla**. У каждой отображенной точки значение x~~*~~ соответствует значению $APFD_C$ при распределении **unit-unit** и значение y соответствует значению $APFD_C$ при одном из двух распределений. Значения $APFD_C$ при распределении **unit-unit** существенно отличаются от значений $APFD_C$ при других распределениях. Это очевидно из большо~~г~~го разброса на графиках для обоих распределений **Mozilla** (рис. 6). Выбор различных комбинаций распределений стоимости тестов и

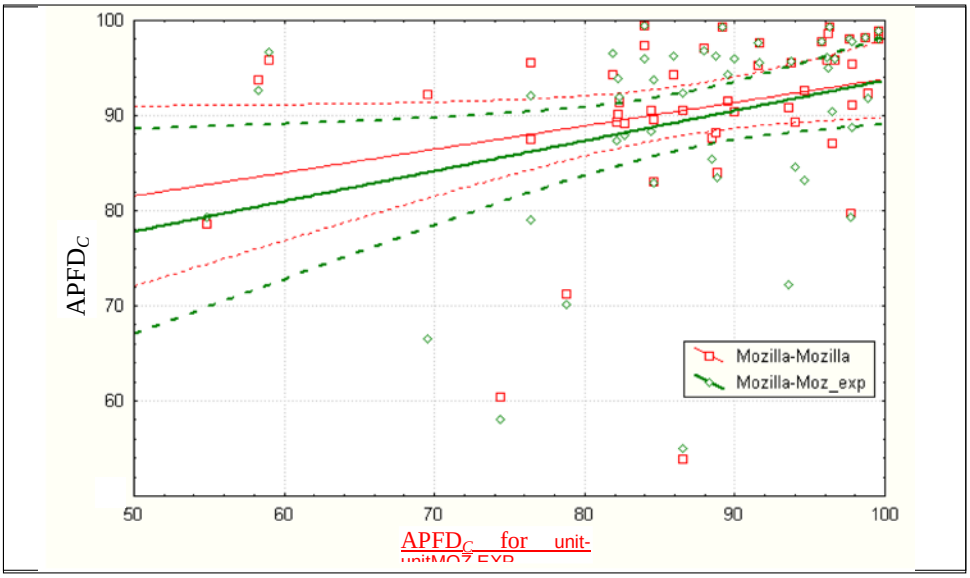

Рис. 6. Диаграмма рассеяния ~~icse01/newFigs/a3b_eps\label{figure_cost_cognizance_study1_resfig4}~~значений ~~severity~~серьезности ошибок имеет влияние ~~impact~~ на метрику $APFD_C$.

Для дальнейшей иллюстрации различий между распределениями на рис. 6 также показаны линии регрессии. Можно заметить, что комбинация

распределений **Mozilla** и **Mozilla-exp** дает наивысшие значения $APFD_C$, но с приближением значений $APFD_C$ к 100 обе комбинации распределений, характеризующих программу **Mozilla,** сходятся.

ВЫВОДЫ

Предложены методы приоритезации и метрика $APFD_C$, учитывающие стоимость тестов и серьезность ошибок. ~~It~~ Исследованы влияние вариации комбинации распределений и
~~scale~~ шкал (например, линейной или экспоненциальной), применяющихся для представления информации о стоимости тестов и серьезности ошибок~~information~~, механизмы ~~путей~~ использования ~~incorporating~~ такой информации в методах приоритезации, а также подходы к получению этой ~~estimating~~ информации. Описано, как различные методы и распределения влияют на метрику $APFD_C$ и как эти методы сравниваются друг с другом ~~compare to each other~~ в зависимости от различных распределений~~scales~~. Показано, как распределения, шкалы ~~scales~~ и другие факторы влияют на исход приоритезации. Определено, что выбор распределения и шкалы ~~scale~~ должен быть сделан на основе анализа, содержащего оценку серьезности ~~severities~~ оши~~в~~бок и стоимости тестов в реальной среде (которые нам пока еще недоступны).

Автор благодарит Г. Ротермела и С. Элбаума за участие в проведении описанных исследований.

ЛИТЕРАТУРА

1. *Ghezzi C., Jazayeri M., Mandrioli D.* Fundamentals of Software Engineering. — Upper Saddle River: Prentice Hall, 1991. — 573 p.
2. *Avritzer A., Weyuker E.J.* The automatic generation of load test suites and the assessment of the resulting software // IEEE Transactions on Software Engineering. — 1995. — **21**, № 9. — P. 705–716.
3. *Wong W., Horgan J., London S., Agrawal H.* A study of effective regression testing in practice // In Proceedings of the Eighth International Symposium on Software Reliability Engineering. — Albuquerque, NM, USA. — 1997. — P. 230–238.
4. *Rothermel G., Untch R., Chu C., Harrold M.J.* Test case prioritization: an empirical study // In Proceedings of the International Conference on Software Maintenance. — Oxford, England, UK. — 1999. — P. 179–188.
5. *Elbaum S., Malishevsky A., Rothermel G.* Prioritizing test cases for regression testing // In Proceedings of the International Symposium on Software Testing and Analysis. — Portland, Oregon. — 2000. — P. 102–112.
6. *Rothermel G., Untch R.H., Chu C., Harrold M.J.* Test case prioritization // IEEE Transactions on Software Engineering. — 2001. — **27**, № 10. — P. 929–948.
7. *Jones J.A., Harrold M.J.* Test-suite reduction and prioritization for modified condition/decision coverage // In Proceedings of the International Conference on Software Maintenance. — Florence, Italy. — 2001. — P. 92–101.
8. *Elbaum Sebastian, Malishevsky Alexey G., Rothermel Gregg.* Test Case Prioritization: A family of empirical studies // IEEE Transactions On Software Engineering. — 2002. — **28**, № 2. — P. 159–182.

9. *Srivastava A., Thiagarajan J.* Effectively prioritizing tests in development environment // In Proceedings of the International Symposium on Software Testing and Analysis. — Via di Ripetta, Rome – Italy. — 2002. — P. 97–106.
10. *Kim J.-M., Porter A.* A history-based test prioritization technique for regression testing in resource constrained environments // In Proceedings of the International Conference on Software Engineering. — Orlando, Florida, USA. — 2002. — P. 119–129.
11. *Srikanth H.* Value-driven system level test case prioritization // Ph.D. Dissertation — North Carolina State University, Releigh, NC. — 2005. — 92 p.
12. *Малышевский А.Г.* Приоритезация тестов в регрессивном тестировании // Системні дослідження та інформаційні технології. — 2006. — № 4. — С. 16–32.
13. *Elbaum S., Malishevsky A., Rothermel G.* Incorporating Varying Test Costs and Fault Severities into Test Case Prioritization // Technical Report 00-60-09. — Computer Science Department. — Oregon State University. — August 2000. — 14 p.
14. *Elbaum S., Malishevsky A., Rothermel G.* Incorporating varying test costs and fault severities into test case prioritization // In Proceedings of the 23rd International Conference on Software Engineering. — Toronto, Ontario, Canada. — May 2001. — P. 329–338.
15. *Malishevsky A.G.* Test case prioritization // Ph.D. Dissertation. — Oregon State University, Corvallis, Oregon, USA. — 2003. — 291 p.
16. *Elbaum S.G., Munson J.C.* Code churn: A measure for estimating the impact of code change // In Proceedings of the International Conference on Software Maintenance. — Bethesda, MD, USA. — 1998. — P. 24–31.
17. *Khoshgoftaar T.M., Munson J.C.* Predicting software development errors using complexity metrics // Journal on Selected Areas in Communications. — 1990. — **8**, № 2. — P. 253–261.
18. *Munson J.C.* Software measurement: Problems and practice // Annals of Software Engineering. — 1995. — **1**, № 1. — P. 255–285.
19. *Elbaum S.G., Munson J.C.* Software evolution and the code fault introduction process // Empirical Software Engineering Journal. — 1999. — **4**, № 3. — P. 241–262.

~~*Ghezzi C., Jazayeri M., Mandrioli D.* Fundamentals of Software Engineering. — Upper Saddle River: Prentice Hall, 1991. — 573 p.~~

~~*Avritzer A., Weyuker E.J.* The automatic generation of load test suites and the assessment of the resulting software // IEEE Transactions on Software Engineering. — 1995. — **21**, № 9. — P. 705–716.~~

~~*Wong W., Horgan J., London S., Agrawal H.* A study of effective regression testing in practice // In Proceedings of the Eighth International Symposium on Software Reliability Engineering. — Albuquerque, NM, USA. — 1997. — P. 230–238.~~

~~*Rothermel G., Untch R., Chu C., Harrold M.J.* Test case prioritization: an empirical study // In Proceedings of the International Conference on Software Maintenance. — Oxford, England, UK. — 1999. — P. 179–188.~~

Elbaum S., Malishevsky A., Rothermel G. Prioritizing test cases for regression testing // In Proceedings of the International Symposium on Software Testing and Analysis. Portland, Oregon. 2000. P. 102–112.

Rothermel G., Untch R.H., Chu C., Harrold M.J. Test case prioritization // IEEE Transactions on Software Engineering. 2001. **27**, № 10. P. 929–948.

Jones J.A., Harrold M.J. Test suite reduction and prioritization for modified condition/decision coverage // In Proceedings of the International Conference on Software Maintenance. Florence, Italy. 2001. P. 92–101.

1. *Elbaum Sebastian, Malishevsky Alexey G., Rothermel Gregg.* Test Case Prioritization: A family of empirical studies // IEEE Transactions On Software Engineering. February 2002. **28**, № 2. P. 159–182.

Srivastava A., Thiagarajan J. Effectively prioritizing tests in development environment // In Proceedings of the International Symposium on Software Testing and Analysis. Via di Ripetta, Rome Italy. 2002. P. 97–106.

2. *Kim J. M., Porter A.* A history based test prioritization technique for regression testing in resource constrained environments // In Proceedings of the International Conference on Software Engineering. Orlando, Florida, USA. 2002. P. 119–129.

Srikanth H. Value driven system level test case prioritization // Ph.D. Dissertation North Carolina State University Releigh, NC. 2005. 92 p.

3. *Малышевский А.Г.* Приоритезация тестов в регрессивном тестировании // Системні дослідження та інформаційні технології. 2006. № 4. С. 16–32.

4. *Elbaum S., Malishevsky A., Rothermel G.* Incorporating Varying Test Costs and Fault Severities into Test Case Prioritization // Technical Report 00-60-09, Computer Science Department, Oregon State University, August, 2000. 14 p.

5. *Elbaum S., Malishevsky A., Rothermel G.* Incorporating varying test costs and fault severities into test case prioritization // In Proceedings of the 23rd International Conference on Software Engineering. Toronto, Ontario, Canada. May 2001. P. 329–338.

6. *Malishevsky A.G.* Test case prioritization // Ph.D. Dissertation. Oregon State University, Corvallis, Oregon, USA. 2003. 291 p.

Elbaum S.G., Munson J.C. Code churn: A measure for estimating the impact of code change // In Proceedings of the International Conference on Software Maintenance. Bethesda, MD, USA. 1998. P. 24–31.

Khoshgoftaar T.M., Munson J.C. Predicting software development errors using complexity metrics // Journal on Selected Areas in Communications. 1990. **8**, № 2. P. 253–261.

Munson J.C. Software measurement: Problems and practice // Annals of Software Engineering. 1995. **1**, № 1. P. 255–285.

7. *Elbaum S.G., Munson J.C.* Software evolution and the code fault introduction process // Empirical Software Engineering Journal. 1999. **4,** № 3. P. 241–262.

Поступила 10.11.2007