УДК 004.9 DOI: 10.30748/zhups.2018.56.17

О.В. Шматко, М.І. Мироненко

Національний технічний університет «Харківський політехнічний інститут», Харків

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ВІДСЛІДКОВУВАННЯ ПОМИЛОК ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розглядається інформаційна технологія тестування, яка здійснює автоматичне сортування та категоризацію звітів про помилки програмного забезпечення. Новизна технології полягає в створенні програмної системи відслідковування помилок, яка дозволяє здійснювати аналіз звітів про помилки з метою їх автоматичного сортування, і, при цьому, оцінювати надійність програмного забезпечення за Коркореном. За результатами базового навантажувального тестування web-компонентів було підтверджено високу функціональну ефективність розробленої програмної системи.

***Ключові** слова: програмне забезпечення, система відслідковування помилок, надійність, тестування, автоматичне сортування.*

Вступ

У зв'язку з широким застосуванням засобів обчислювальної техніки практично у всіх галузях промисловості, бізнесу та в багатьох сферах повсякденного життя, все більшого значення набувають питання якості та надійності програмного забезпечення ЕОМ. Якщо надійність функціонування апаратних засобів обчислювальної техніки неухильно підвищується, то все частіше причинами різних відмов і збоїв у роботі обчислювальних систем стають помилки в програмах, які не були своєчасно виявлені на етапах автономного і комплексного налагодження, тестування та приймально-здаточних випробуваннях [1]. За даними праці [2] пошук і виправлення помилок у програмах на ЕОМ у деяких екстремальних випадках може потребувати до 80% фінансових ресурсів проект.

У міжнародному стандарті ISO 9126:1991 [3] надійність виділена як одна з основних характеристик якості програмного забезпечення (ПЗ). Стандартний словник термінів програмного інжинірингу [4] визначає надійність програмного забезпечення як здатність системи або компонента виконувати необхідні функції в заданих умовах протягом зазначеного періоду часу. Сама проблема надійності програмного забезпечення має, принаймні, два аспекти: забезпечення і оцінка (вимір) надійності [5]. Практично вся наявна література присвячена першому аспекту, а питання оцінки надійності комп'ютерних програм недостатньо опрацьовано. При цьому ніякої загальноприйнятої кількісної міри надійності програм досі не існує.

У статті пропонується інформаційна технологія, яка є поєднанням методів та алгоритмів, належних до двох складових надійності ПЗ – аналізу надійності та забезпечення надійності, в єдиній придатній для практичної реалізації системі, яка окрім цього здатна здійснювати автоматичну категоризацію звітів о помилках.

Огляд систем відстеження помилок програмного забезпечення

Забезпечення надійності програмного забезпечення неможливо вирішити без використання системи відстеження помилок (англ. bug tracking system). Багтрекер – це прикладна програма, яка дозволяє тестувальникам та програмістам відстежувати історію звітів про баги (bugs) під час своєї роботи [6]. Система відслідковування помилок є необхідним компонентом хорошої інфраструктури розробки програмного забезпечення. Головний компонент багтрекера – база даних, що записує факти про відомі баги. Факти можуть включати час звіту про баг, його серйозність, неправильну поведінку програми, деталі про те, як відтворити помилку, а також особу, що повідомила про помилку, та програмістів, котрі могли працювати над її виправленням. Головна перевага багтрекера полягає в забезпеченні чіткого централізованого огляду запитів розробки та їх стану. До найбільш відомих та розповсюджених багтрекерів належать:

1. Bugzilla є одним з провідних інструментів відстеження помилок та використовуються багатьма організаціями протягом вже досить тривалого часу. Це ПЗ має дуже простий у використанні вебінтерфейс. Воно має усі необхідні риси, зручності та інструменти. Також однією з важливих характеристик є те, що це ПЗ має повністю відкритий програмний код і може вільно використовуватись [7].

2. Atlassian JIRA є в першу чергу інструментом управління інцидентами, який також широко використовується для відстеження помилок. Це ПЗ забезпечує повний набір запису, звітності, документообігу та інших зручних функцій. Цей інструмент інтегрується безпосередньо з середовищами розроб-

ки коду, що робить його ідеальним вибором для розробників. Крім того, це ПЗ не обов'язково має бути зосереджено тільки в індустрії розробки програмного забезпечення та підтримує гнучку (agile) розробку. Це комерційний продукт з великою кількість надбудов [8].

3. MantisBT – це найбільш проста в освоєнні та використанні система, що має майже усі функції необхідні для роботи. Ця система зроблена як Вебдодаток та навіть має мобільну версію. Уся система написана на PHP та є безкоштовною для використання, якщо користувач не потребуєте послуг хостингу [9].

4. Trac написаний на мові програмування Python. Окрім відстеження дефектів, ця система також надає можливість інтеграції з Subversion. Веб інтерфейс є дуже зручним у використанні та надає такі можливості як відстеження етапів та карту проекту [10].

В той же час в праці [11] зазначається, що у більшості існуючих багтрекерів відсутня підтримка відстеження та автоматичного збереження звітів о помилках, що є досить вагомим їх недоліком.

Метою цієї роботи є розробка алгоритмічного забезпечення і програмних компонент для оцінки надійності мобільного ПЗ шляхом поєднання методики аналізу надійності ПЗ та алгоритмів аналізу даних, а також, розробка методу автоматичного сортування та категоризації звітів про помилки.

Основний розділ

Постановка задачі дослідження

Ґрунтуючись на наведеному огляді, можна зробити висновок, що серед систем оцінки або забезпечення надійності ПЗ не має гострої потреби у нових програмних або алгоритмічних рішеннях. Замість створення радикально нових алгоритмів або їх модифікацій, доцільно поєднати ці дві області та створити модель продукту, який забезпечить можливість оцінювати та забезпечувати надійність ПЗ за допомогою єдиного додатку.

Крім того, важливе значення має вирішення проблеми автоматичного збору та сортування звітів о помилках. У більшості масштабних проектів зазвичай реалізовано засоби автоматичного збору звітів о помилках, але це призводить до необхідності проведення аналізу дуже великих обсягів інформації. Досить відомим фактом є те, що єдиний програмний дефект може створювати різні помилки у різних програмних модулях системи. Найпростішим прикладом може слугувати помилка у методі класу обчислення даних, яка призводить до припинення роботи програми у випадку запуску розрахунків для існуючих даних і також спричинює помилку й у модулі імпорту даних, коли вони автоматично перевіряються на коректність тим самим методом. У праці [12] наведено приклад, коли багато звітів може відповідати єдиному дефекту або помилці (далі позначені як подвійні звіти), а єдиний дефект може призводити до виникнення багатьох помилок з споріднеими звітами. Обидва розглянуті сценарії призводять до засмічення системи якщо звіти не сортируються та групуються. Групування звітів на категорії згідно до помилки, яка спричинила некоректну поведінку ПЗ має ряд переваг, серед яких можна виділити зменшення обсягу інформації для аналізу та можливість швидкого перегляду інформації з різних звітів які належать до однієї групи помилок, а тому й спрощення знаходження дефектного модуля.

Таким чином, існує необхідність спрощення процесу оформлення, аналізу та перегляду звітів про помилки і підвищення ефективності та надійності процесу тестування шляхом автоматичної категоризації звітів.

Архітектура та моделі програмної системи

Програмна система була реалізована як вебдодаток на базі існуючої системи менеджменту дефектів з відкритим кодом JTrac, яка створена за допомогою мови Java і є безкоштовним гнучким засобом відстеження дефектів [10]. При цьому була вибрана трирівнева архітектура (англ. three-tier або Multitier architecture), яка передбачає наявність наступних компонент програми: клієнтський застосунок («тонкий клієнт» або термінал), підключений до сервера застосунків, який в свою чергу підключений до серверу бази даних. При цьому клієнтський рівень не має прямих зв’язків з базою даних (за вимогами безпеки) і зберігає стан програми (за вимогами надійності). На перший рівень винесена найпростіша бізнес-логіка: інтерфейс авторизації, алгоритми шифрування, перевірка значень, що вводяться, на допустимість і відповідність формату, нескладні операції (сортування, групування, підрахунок значень) з даними, вже завантаженими на термінал. Сервер застосунків розташовується на другому рівні, де зосереджена і більша частина бізнес-логіки. Сервер реляційної бази даних забезпечує зберігання даних і виноситься на третій рівень. У порівнянні з клієнт-серверною або файл-серверною архітектурою трирівнева архітектура забезпечує масштабованість, конфігурованість, високий рівень безпеки і надійності та невисокі вимоги до швидкості каналу між терміналами, серед яких може бути, наприклад, мобільний телефон, і сервером застосунків.

При проектуванні програми було використано патерн MVC (Model-Viewer-Controller). Розробка здійснювалася в стандарті UML 2.0 діаграм, які моделювали статичну та динамічну структуру програмної системи. На рис. 1 показано ER-діаграму бази даних програмної системи відслідковування помилок.

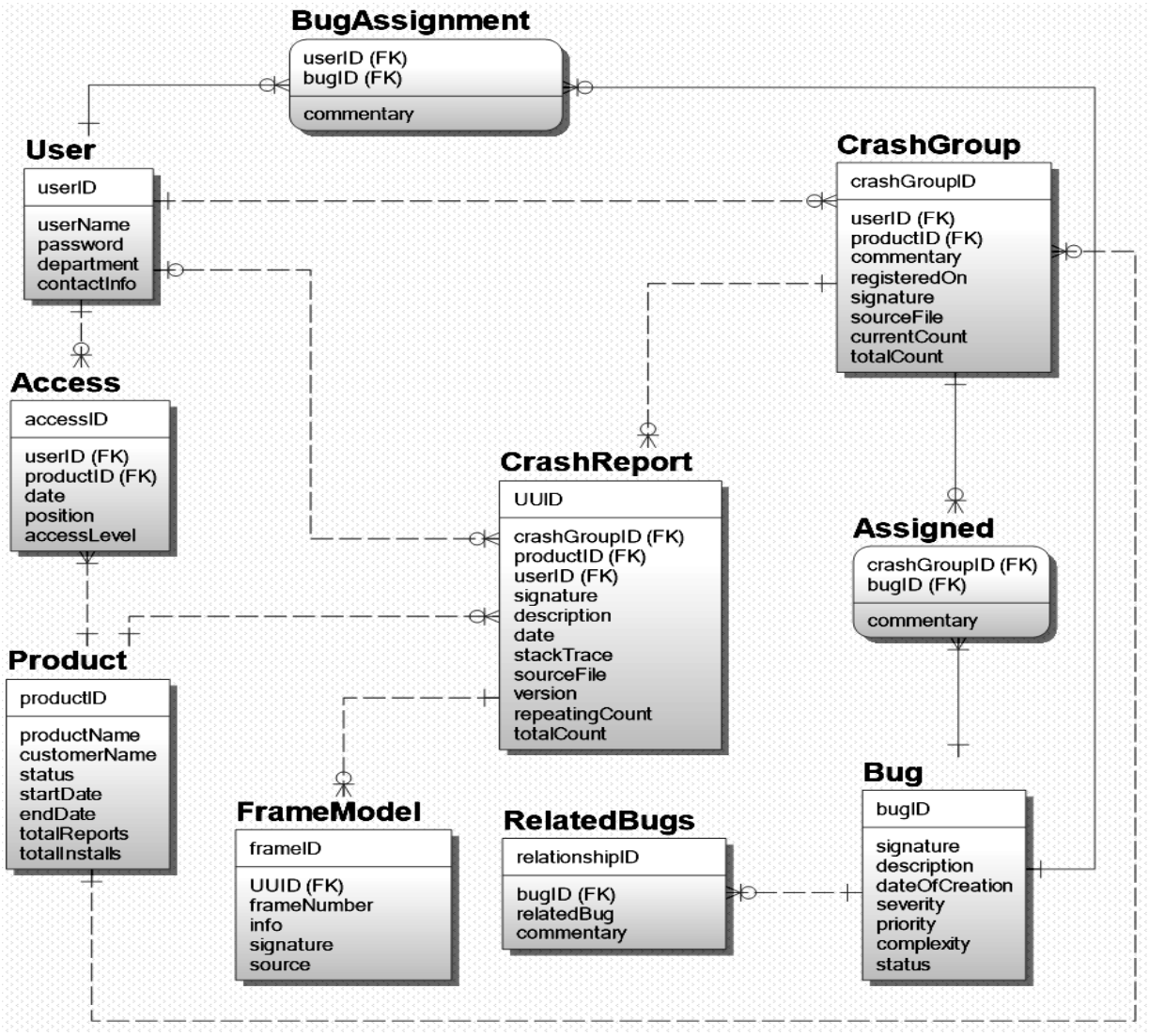

Рис. 1. ER-діаграма бази даних програмної системи

У процесі розробки формального опису системи відслідковування помилок було використано такі CASE-засоби:

1). Visual Paradigm, який забезпечує розробнику зручну платформою для побудови програмних рішень у швидший, кращий та дешевший спосіб.

2). Visual Paradigm, який забезпечує розробнику зручну платформою для побудови програмних рішень у швидший, кращий та дешевший спосіб.

3). CA ERwin Data Modeler – це CASE-засіб для проектування та документації баз даних, яке дозволяє створювати, документувати і супроводжувати бази даних, сховища і вітрини даних.

Для написання цільового ПЗ використовувалася мова програмування Java EE з підтримкою технології Servlet. Причиною такого вибору є наявність безкоштовних серверів та інтегрованого середовища розробника, кросплатформність та можливість розробки комерційних програм без необхідності придбання додаткових ліцензій.

Як сервер застосунків для цієї роботи було обрано Jetty. Перевагами цього серверу є простота розгортання та використання, низькі затрати системних ресурсів, відсутність необхідності придбання платної ліцензії, висока пристосованість до розробки та розгортання додатків на Java мові, а також підтримка сервлетів.

Як сервер застосунків для цієї роботи було обрано Jetty. Перевагами цього серверу є простота розгортання та використання, низькі затрати системних ресурсів, відсутність необхідності придбання платної ліцензії, висока пристосованість до розробки та розгортання додатків на Java мові, а також підтримка сервлетів.

Як модель оцінки надійності ПЗ було вибрано модель Коркорена [12]. Відмінність цієї моделі полягає в тому, що в ній не використовуються параметри часу тестування, а враховується тільки результат N випробувань, в яких виявлено N_i помилок і-го типу.

Показник рівня надійності обчислюється за формулою

$$R = \frac{N_o}{N} + \sum_{i=1}^{K} \frac{Y_i \cdot (N_i - 1)}{N}, \qquad (1)$$

де N_0 – число безвідмовних випробувань, виконаних у серії з N випробувань; K – відоме число типів помилок; Y_i – імовірність появи помилки і-го типу, яка оцінюється на підставі статистичних даних попереднього періоду функціонування

Кількість тестів N_i для формули Коркорена при неповному наборі тестових звітів визначається як

$$N_i = \frac{R_i \cdot N_t \cdot 0,6}{R_t},$$

де R_i – кількість імпортованих до системи звітів про помилки; R_t – загальна кількість звітів на сервері Socorro; N_t – загальна кількість інсталяцій продукту.

Результати тестування програмної системи

Для розробленого ПЗ було проведено функціональне та навантажувальне тестування. Функціональне тестуванні здійснювалося за типом чорної скриньки, яка засновує свої тестові випадки на специфікаціях програмного компонента в процесі тестування. На рис. 2 показано інтерфейс програми автоматичного генерування групи звітів та обчислення за формулою (1) показника надійності Коркорена для згенерованих груп звітів.

Action	Group ID	Signature	Product	Version	Count	Impact
info / delete	3	mozilla::gl::CreateSurfaceForWindow	FennecAndroid	49.0a1	14	0.066
info / delete	2	java.lang.Exception: Error loading gecko libraries at org.mozilla.gecko.mozglue.GeckoLoader.loadGeckoLibsNative(Native Method)	FennecAndroid	49.0a1	3	0.002
info / delete	8	@0xf738438c	FennecAndroid	49.0a1	3	0.014
info / delete	1	@0x99293258	FennecAndroid	49.0a1	2	0.009
info / delete	4	mozilla::gl::GLContextProviderEGL::CreateForWindow	FennecAndroid	49.0a1	2	0.009
info / delete	5	@0x668f3166	FennecAndroid	49.0a1	2	0.009
info / delete	6	java.lang.UnsatisfiedLinkError: at java.lang.Runtime.loadLibrary(Runtime.java)	FennecAndroid	49.0a1	2	0.009
info / delete	7	@0xf6f59ce0	FennecAndroid	49.0a1	2	0.009
info / delete	9	java.lang.NullPointerException: at org.mozilla.gecko.GeckoNetworkManager.updateNetworkStateAndConnectionType(GeckoNetworkManager.java)	FennecAndroid	49.0a1	2	0.009
info / delete	10	@0x9c83a6ea	FennecAndroid	49.0a1	2	0.009
info / delete	11	@0x978f4258	FennecAndroid	49.0a1	2	0.009
info / delete	12	java.lang.NullPointerException: lock == null at java.io.Reader.<init>(Reader.java)	FennecAndroid	49.0a1	2	0.009
info / delete	13	java.lang.UnsatisfiedLinkError: Couldn"t load mozglue: findLibrary returned null at java.lang.Runtime.loadLibrary(Runtime.java)	FennecAndroid	49.0a1	2	0.009
info / delete	14	OOM | small	FennecAndroid	49.0a1	2	0.009
info / delete	15	@0xb6c9f35c	FennecAndroid	49.0a1	2	0.009
info / delete	16	@0xf6a8b41e	FennecAndroid	49.0a1	2	0.009

Рис. 2. Автоматично генеровані групи звітів та показник надійності

На рис. 2 кнопка “info” відображає сторінку з усіма звітами належними до цієї групи та пропонує виконати набір стандартних дій щодо цих звітів (переглянути інформацію, видалити). Загальний показник надійності Коркорена (1) для програмної системи обчислюється на основі автоматично створених груп звітів. Рівень впливу кожної групи на загальну надійність системи зображено у останній колонці таблиці. Цей показник позначає на скільки достовірність успішного виконання програми збільшиться, якщо дефект, представлений цією групою звітів, буде усунутий.

Для проведення базового навантажувального тестування web-компонентів системи був використаний Apache JMeter. Для першого тест-кейса для 500 користувачів система показала високу функціональну ефективність: рівень відмов – 0 % і максимальний час завантаження – 10 мс (рис. 3).

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
login get	500	634	1	1494	400.68	0.00%	165.2/sec	456.95	2831.8
login post	500	2274	12	5589	1813.79	0.00%	76.6/sec	723.34	9672.0
options get	500	697	1	1998	384.59	0.00%	68.5/sec	369.84	5530.0
spaces get	500	975	2	4993	899.47	0.00%	58.7/sec	311.41	5430.0
space form1 get	500	811	3	4814	668.16	0.00%	57.4/sec	363.28	6478.0
space form1 p...	500	753	2	4098	590.10	0.00%	50.8/sec	331.13	6673.0
space form2 p...	500	738	2	2156	544.55	0.00%	45.4/sec	213.70	4820.0
space form3 p...	500	694	2	2109	534.37	0.00%	42.2/sec	199.74	4847.0
space allocate...	500	721	2	2146	550.45	0.00%	39.7/sec	187.83	4847.0
dashboard get	500	335	2	3869	369.12	0.00%	40.7/sec	378.90	9523.0
logout	500	3950	426	10022	1664.62	0.00%	34.1/sec	67.83	2039.0
TOTAL	5500	1144	1	10022	1351.73	0.00%	342.0/sec	1903.54	5699.2

Рис. 3. Jmeter: результат для 500 одночасних потоків

Для другого тест-кейса для 1000 користувачів рівень помилок збільшився до 2,7 %, а максимальний час завантаження – до 28 мс (рис. 4).

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
login get	1000	701	0	1845	438.33	2.10%	337.2/sec	925.82	2811.9
login post	1000	3192	12	11886	2584.97	2.60%	73.1/sec	676.00	9464.1
options get	1000	1670	1	3520	946.27	2.70%	60.4/sec	319.99	5424.0
spaces get	1000	2059	1	12314	1698.76	2.70%	49.4/sec	256.75	5326.4
space form1 get	1000	1561	1	7793	1235.56	2.70%	38.3/sec	237.41	6346.1
space form1 p...	1000	1616	1	9536	1249.48	2.70%	36.4/sec	232.40	6535.8
space form2 p...	1000	1663	1	5854	1262.24	2.70%	34.4/sec	158.98	4732.9
space form3 p...	1000	1824	0	5849	1367.75	2.70%	34.0/sec	158.17	4759.1
space allocate...	1000	2079	1	5869	1522.82	2.70%	33.5/sec	155.56	4759.1
dashboard get	1000	1066	2	7365	997.28	0.00%	33.3/sec	306.55	9416.0
logout	1000	7306	1	27916	4233.41	2.60%	30.5/sec	60.46	2027.6
TOTAL	11000	2249	0	27916	2531.53	2.38%	332.1/sec	1816.29	5600.3

Рис. 4. Jmeter: результат для 1000 одночасних потоків

Порівняльний аналіз рис. 3–4 показує, що при навантаженні на систему більше 1000 користувачів, необхідне перенесення системи на більш продуктивну апаратну платформу. Статистичний аналіз роботи автоматичної функції видалення подвійних звітів показав, що на кожні 100 звітів розроблене ПЗ може знайти та видалити у середньому 21 подвійний звіт, тобто біля 20 % від загальної кількості звітів. У масштабах систем, де є необхідність збереження мільйонів звітів, 20 % може дати велику прибавку к ефективності системи, а саме тому тестування можна вважати успішним.

Висновки

1. Аналіз найбільш відомих систем-аналогів, призначених для оцінки або забезпечення надійності ПЗ підтвердив необхідність спрощення процесу оформлення, аналізу та перегляду звітів і підвищення ефективності та надійності процесу тестування.

2. В роботі розроблено алгоритмічне і програмне забезпечення системи тестування, реалізація якого дозволяє здійснювати аналіз звітів про помилки з метою їх автоматичного сортування і одночасно обчислювати загальний показник надійності Коркорена.

3. Для подальшого удосконалення розробленого алгоритмічного забезпечення необхідно провести більш детальний аналіз основних метрик надійності ПЗ та типів помилок і застосувати інтелектуальні інформаційні технології аналізу тексту з метою вилучення корисної інформації з коментарів та описів у звітах.

Список літератури

1. Мищенко В.О. CASE-оценка критических программних систем: в 3-х т. – Т. 1: Качество / В.О. Мищенко, О.В. Поморова, Т.А. Говорущенко; под ред. В.С. Харченко. – Х.: Нац. Аэрокосмический университет "ХАИ", 2012. – 201 с.

2. Zeller, Andreas, “Why Programs Fail: A Guide to Systematic Debugging”, Morgan Kaufmann. – 2005. – 480 p.

3. ISO 9126:1991. Software engineering – Product quality. – 186 p. IEEE Std 610.12-1990.

4. IEEE Standard Glossary of Software Engineering Technology (ANSI). – 1283 p.

5. Brian, Marick. When Should a Test Be Automated? Reliable Software Technologies, Pisa, Italy, June 13-17, 2016, Proceedings, 213 p.

6. Spolsky, Joel. Painless Bug Tracking [Electronic resource]. Access mode: http://www.joelonsoftware.com/articles/fog0000000029.html/, 06.006.2017.

7. Система відстеження помилок і ведення завдань Bugzilla. [Електронний ресурс]. Режим доступу: https://www.bugzilla.org/, 06.06.2017.

8. Система відстеження помилок і ведення завдань Atlassian JIRA. [Електронний ресурс]. Режим доступу: https://www.atlassian.com/software/jira/, 06.06.2017.

9. Безкоштовна система відстеження помилок з відкритим вихідним кодом Mantis. [Електронний ресурс]. Режим доступу: https://www.mantisbt.org/, 06.06.2016.

10. Вільний браузерний застосунок для управління проектами Trac. [Електронний ресурс]. Режим доступу: https://trac.edgewall.org/.

11. Jonathan Corbet. Distributed bug tracking. [Electronic resource]. Access mode: http://www.webcitation.org/6HyFiJjnW/, 18.01.2018.

12. Martin Wayne. Methodology for assessing reliability growth using multiple information sources [Electronic resource]. 2013 – 261 p. Access mode: http://drum.lib.umd.edu/bitstream/handle/1903/14818/Wayne_umd_0117E_14762. pdf/, 18.01.2018.

References

1. Myshchenko, V.O., Pomorova, O.V., Hovorushchenko, T.A. (2012), "*CASE-otsenka krytycheskykh prohrammnykh system: v 3-kh t*., T. 1: Kachestvo" [*CASE-evaluation of critical software systems: in 3 t.*] / pod red. V.S. Kharchenko. – Kharkov: Nats. Aэrokosmycheskyi unyversytet "KhAY", 201 s.
2. Zeller, Andreas (2005), "*Why Programs Fail: A Guide to Systematic Debugging*", Morgan Kaufmann, 480 p.
3. ISO 9126:1991. Software engineering – Product quality, 186 p.
4. IEEE Std 610.12-1990. 4. IEEE Standard Glossary of Software Engineering Technology (ANSI), 1283 p.
5. Brian, Marick (2016), "When Should a Test Be Automated? Reliable Software Technologies", Pisa, Italy, June 13-17, Proceedings, 213 p.
6. Spolsky, Joel. Painless Bug Tracking [Electronic resource]. Access mode: http://www.joelonsoftware.com/articles/fog0000000029.html/, 19.01.2017.
7. Systema vidstezhennia pomylok i vedennia zavdan Bugzilla. [Electronic resource]. Access mode: https://www.bugzilla.org/, 06.06.2017.
8. Systema vidstezhennia pomylok i vedennia zavdan Atlassian JIRA. [Electronic resource]. Access mode:https://www.atlassian.com/software/jira/, 06.06.2017.
9. Bezkoshtovna systema vidstezhennia pomylok z vidk-rytym vykhidnym kodom Mantis. [Electronic resource]. Access mode: https://www.mantisbt.org/, 06.06.2016.
10. Vilnyi brauzernyi zastosunok dlia upravlinnia proektamy Trac. [Electronic resource]. Access mode: https://trac.edgewall.org/, 06.06.2017.
11. Corbet, Jonathan. Distributed bug tracking. [Electronic resource]. Access mode: http://www.webcitation.org/6HyFiJjnW/, 18.01.2018.
12. Wayne, Martin (2013), "Methodology for assessing reliability growth using multiple information sources" [Electronic resource]. Access mode: http://drum.lib.umd.edu/bitstream/handle/1903/14818/Wayne_umd_0117E_14762. pdf/, 18.01.2018.

Надійшла до редколегії 21.02.2018
Схвалена до друку 15.05.2018

Відомості про авторів:

Шматко Олександр Віталійович
кандидат технічних наук доцент
доцент кафедри Національного технічного
університету «Харківський
політехнічний інститут»,
Харків, Україна
https://orcid.org/0000-0002-2426-900X

Мироненко Микита Ігорович
магістр національного технічного університету
«Харківський політехнічний інститут»,
Харків, Україна

Information about the authors:

Oleksandr Shmatko
Candidate of Technical Sciences Associate Professor
Senior Lecturer
of Nacional Tehnical University
«Kharkiv Polytechnic Institute»
Kharkiv, Ukraine
https://orcid.org/0000-0002-2426-900X

Mykyta Myronenko
Master of Nacional Tehnical University
«Kharkiv Polytechnic Institute»
Kharkiv, Ukraine

ИНФОРМАЦИОННАЯ ТЕХНОЛОГИЯ ОТСЛЕЖИВАНИЯ ОШИБОК ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

А.В. Шматко, Н.И. Мироненко

Рассматривается информационная технология тестирования, которая осуществляет автоматическую сортировку и категоризацию отчетов об ошибках программного обеспечения. Новизна технологии заключается в создании программной системы отслеживания ошибок, которая позволяет осуществлять анализ отчетов об ошибках с целью их автоматической сортировки и при этом оценивать надежность программного обеспечения по Коркорэн. По результатам базового нагрузочного тестирования web-компонентов было подтверждено высокую функциональную эффективность разработанной программной системы.

Ключевые слова: *программное обеспечение, система отслеживания ошибок, надежность, тестирование, автоматическая сортировка.*

INFORMATION TECHNOLOGY OF DEPENDING OF ERRORS SOFTWARE

O. Shmatko, M. Myronenko

The information technology of testing, which carries out automatic sorting and categorization of software error reports, is considered. The novelty of technology is to create a software error tracking system that allows you to analyze the error reports in order to automatically sort them out and at the same time evaluate the reliability of the software by the integral index of Korkorena. For automatic analysis the text of the error reports, a structured measurement method for editing distance is used. How to distance editing is used is the Levenstein distance, which was determined by Wagner-Fisher's algorithm for dynamic programming. The software system is implemented as a web-based application based on the existing JTrac open source defect management system, which is created using Java language and is a freeware defective software tracking software. For the developed software, functional and load testing was carried out. In this case, functional testing was carried out according to the type of black box. The statistical analysis of the automatic function of removing duplicate reports has shown that it finds and removes more than 20 % of the total number of reports, which significantly increases the functional efficiency of the software crawl tracking system. The results obtained can be used to create error tracking systems that store large ones. Volume Reporting Software Deficiencies.

Keywords: *software, error tracking system, reliability, testing, automatic sorting.*