УДК 004.054

ВЕЛЬМА А.М.
ЛАКТІОНОВ Є.Ю.

ВИБІР СИСТЕМИ ВІДСТЕЖЕННЯ ПОМИЛОК В ЗАЛЕЖНОСТІ ВІД КОНФІГУРАЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

В статті представлений огляд деяких систем відстеження помилок, їх порівняння і рекомендації до вибору. На сьогоднішній день спостерігається перехід від продуктивності до якості в основних вимогах до програмного забезпечення. Тому компаніям необхідно організувати свої бізнес-процеси так, щоб відповідати цим вимогам. Актуальність даної статті полягає в тому, що компаніям-розробникам доцільно використовувати системи відстеження помилок для забезпечення ефективності в їх виявленні і виправленні.

The paper presents the review of some tracking systems, their comparison and recommendations to choice. Today there is a transition from the productivity to quality in the basic requirements to software. Therefore companies must organize the business process to conform to these requirements. Actuality of this article consists of that developing companies are expedient to utillize the tracking systems for providing of efficiency in their discovery and correction

1. Вступ

У даній статті розглядаються різні системи відстеження помилок в програмному за безпеченні і виконується аналіз критеріїв вибору такої системи залежно від конфігурації програмного забезпечення, що розробляється.

2. Постановка задачі

Існує багато систем відстежування помилок у досить широкому діапазоні функціональності і вартості. Як критерій вибору відповідної системи для конкретної розробки можна вибрати оцінку надійності програмної системи, що розробляється, керуючись її мірою складності по Холстеду.

3. Місце систем відстеження помилок у розробці програмних засобів

Методи тестування ПЗ, що існують на сьогоднішній день, не дозволяють однозначно і повністю виявити всі дефекти і встановити коректність функціонування аналізованої програми, тому всі існуючі методи тестування діють в рамках формального процесу перевірки ПЗ, що досліджується або розробляється.

Такий процес формальної перевірки або верифікації може довести, що дефекти відсутні з погляду використовуваного методу (тобто немає ніякої можливості точно встановити або гарантувати відсутність дефектів в програмному продукті з урахуванням людського чинника, присутнього на всіх етапах життєвого циклу ПЗ).

Якість програмного продукту характеризується набором властивостей, що визначають, наскільки продукт «гарний» з погляду зацікавлених сторін, таких як замовник продукту, спонсор, кінцевий користувач, розробники і тестувальники продукту, інженери підтримки, співробітники відділів маркетингу, навчання і продажу [1]. Кожен з учасників може мати різне уявлення про продукт і про те, наскільки він гарний або поганий, тобто про те, наскільки висока якість продукту. Таким чином, постановка завдання забезпечення якості продукту виливається в завдання визначення зацікавлених осіб, їх критеріїв якості і потім знаходження оптимального рішення, що задовольняє цим критеріям. Тестування є одним з найбільш сталих способів забезпечення якості розробки програмного забезпечення і входить в набір ефективних засобів сучасної системи за безпечення якості програмного продукту.

З технічної точки зору тестування полягає у виконанні застосування на деякій множині початкових даних і звірці отримуваних результатів із заздалегідь відомими (еталонними) з метою встановити відповідність різ-

них властивостей і характеристик застосування замовленим властивостям. Як одна з основних фаз процесу розробки програмного продукту (Дизайн додатку – Розробка коду – Тестування), тестування характеризується достатньо великим внеском в сумарну трудомісткість розробки продукту. Широко відома оцінка розподілу трудомісткості між фазами створення програмного продукту: 40% – 20% – 40% (рис. 1) з чого виходить, що найбільший ефект в зниженні трудомісткості може бути отриманий перш за все на фазах дизайну і тестування.

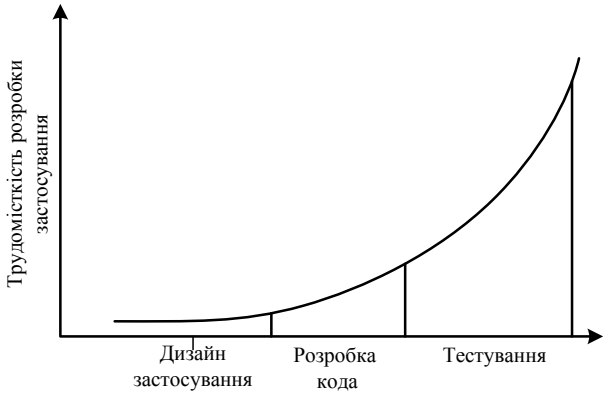

Рис. 1. Розподіл трудомісткості розробки застосування

Для скріплення роботи програміста і тестувальника при розробці програмного забезпечення, як великі, так і маленькі софтверні компанії використовують системи обліку завдань, помилок, управління проектами (bug tracker, issue tracking system, project management application).

На даний момент такого роду продуктів існує немало. Є прості системи, функціонал яких обмежується обліком помилок і відстеженням їх статусу. Є складніші, які дозволяють, наприклад, будувати різні графіки з проектних ризиків, інтегруватися з системами контролю версій, здійснювати складний пошук у проектній документації і так далі.

В ідеалі система управління проектами - це деяке серверне застосування, яке дозволяє робити наступне:

- в реальному часі відстежувати поточний стан проектів, збирати статистику проектів;
- вести облік помилок, завдань, поліпшень відповідно до заданого життєвого циклу;
- зберігати проектну документацію (мати вбудовану базу знань, або Wiki);
- конфігурувати права доступу користувачів, їх ролі, відправку нотифікацій;
- інтегруватися з різними third-party продуктами (наприклад, з тими ж системами контролю версій);
- доступ до функцій системи програмним способом (через відповідний API).

Принцип використання системи відстеження помилок зображений на рис. 2.

Також можливі і інші способи вико ристання, наприклад без web-сервера система відстеження помилок працюватиме як standalone сервер.

Слід відмітити, що автоматизована система відстеження проблем, перш за все, вирішує зовсім не технічні, а політичні питання. Це могутній організаційний засіб з великими можливостями.

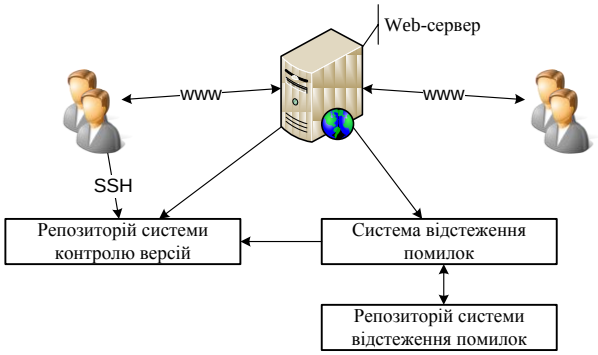

Рис. 2. Використання системи відстеження помилок і інтеграція з системою контролю версій

Система є засобом відстеження ходу робіт. Інформація, яка традиційно була в розпорядженні тільки у керівництва і декількох програмістів, стає доступною широкому кругу співробітників різних рівнів. Система надає своїм користувачам об'єктивну і незалежну оцінку ходу виконання робота і їх відповідності графіку. У будь-який момент можна проглянути повний список завдань, які ще мають бути виконані. Поточний стан продукту і його якість завжди відомі. І кожен може побачити, як просуваються роботи і наскільки швидко вирішуються поставлені завдання.

Система також є засобом організації взаємодії між співробітниками. У кожній компанії, так або інакше, вирішується цілий ряд організаційних питань. Коли процес їх рішення автоматизований, дії співробітників стають більш регламентованими і впорядкованими. Більш того, система висвічує деякі внутрішні проблеми компанії, що раніше

залишилися прихованими. Гарна система, особливо мережева, дозволяє практично повністю відстежувати взаємодію між співкобітниками, що беруть участь у виявленні недоліків програмного продукту, щоб вчасно вирішувати спірні питання і приймати заходи по оптимізації робіт. Зокрема, з її допомогою можна зафіксувати випадки некоректної поведінки окремих співробітників або груп, зловживання чужим робочим часом, необґрунтованих претензій і тому подібне

Система відображає продуктивність роботи кожного співробітника. З бази даних неважко отримати будь-яку статистичну інформацію, наприклад, про кількість звітів, що здаються тестувальником за день, середній кількості помилок що допускаються програмістом за тиждень. Такі дані можуть бути корисні для аналізу ходу розробки і вирішення поточних проблем.

4. Існуючі системи відстеження помилок і їх властивості

Розглянемо деякі системи відстеження помилок [2 – 3].

Bugzilla – застосування для відстеження помилок, розроблене компанією Mozilla. Фактично, Bugzilla використовується в багатьох корпораціях для розробки власного програмного забезпечення для корпоративних потреб.

За минулий час Bugzilla став зрілим продуктом, що по праву гордиться розвиненою функціональністю, куди, зокрема, входять:

- інтегрована система безпеки з можливістю визначення безпеки на рівні продуктів;
- система залежностей помилок і виведення залежностей помилок в графічному вигляді;
- розвинена система складання звітів;
- стабільний, перевірений часом back-end на основі RDBMS;
- розвинена система конфігурації;
- дуже зрозумілий і добре продуманий протокол виправлення помилок
- API для електронної пошти, XML, HTTP і консолі;
- доступна інтеграція з системами управління автоматичної конфігурації програ

много забезпечення, зокрема Perforce and CVS (через інтерфейс електронної пошти Bugzilla і скрипти для checkin/checkout).

Trac – це досить проста система відстеження помилок, що проте помітно спрощує життя розробника при належному терпінні. Основою для Trac'a є SVN репозиторій. Тому найбільш функціональним рішенням буде робота в зв'язці Subversion – Trac.

Основними функціями цієї системи є:

- управління проектом
  а) розділення проекту на етапи (milestones);
  б) контроль виконання (roadmap);
  в) всі зміни за проектом заносяться на часову шкалу (timeline);
  г) підтримка RSS;
- tickets – облік помилок, зауважень, побажань з можливістю фільтрації і занесення відповідно в milestone, roadmap;
- перегляд репозиторія – достатньо зручний модуль перегляду Subversion репозиторія проекту. Дозволяє переглядати вихідний код з урахуванням ревізії, а також змін;
- управління користувачами. Проста система – вказується, що можуть робити користувачі, а що ні;
- Wiki – в Trac вбудована система wiki з можливістю робити посилання на milestone, roadmap, ticket. Органічно вписується і зручна у використанні при веденні проекту.

Redmine – відкрите серверне застосування для управління проектами і відстеження помилок. Redmine написаний на Ruby і є застосуванням на основі широко відомого вебфреймворка Ruby on Rails.

Даний продукт надає наступні можливості:

- ведення декількох проектів;
- гнучка система доступу, що базується на ролях;
- система відстеження помилок;
- діаграми Ганта і календар;
- ведення новин проекту, документів і управління файлами;
- сповіщення про зміни за допомогою RSS-потоків і електронної пошти;
- вики для кожного проекту;
- форуми для кожного проекту;

- облік витрат часу;
- довільні поля, що настроюються, для інцидентів, витрат часу, проектів і користувачів;
- легка інтеграція з системами управління версіями (SVN, CVS, Git, Mercurial, Bazaar і Darcs);
- створення записів про помилки на основі одержаних листів;
- підтримка множинної аутентифікації LDAP;
- можливість самостійної реєстрації нових користувачів;
- багатомовний інтерфейс (зокрема російський);
- підтримка СУБД MYSQL, POSTGRESQL, SQLite, Oracle.

Порівняння представлених систем відстеження помилок представлене у табл. 1.

Також слід відмітити, що функціональність даних систем можна розширити або змінити за допомогою доповнень (plugins). Також, оскільки дані системи відстеження помилок є проектами з відкритим кодом, можливо самостійно додати необхідну функціональність.

Розглянемо випадки, коли необхідно використовувати системи відстеження помилок.

Можна представити ситуацію, коли в отриманому продукті кількість передбачуваних помилок не перевищуватиме певного показника (наприклад, для програми в 1000 рядків можна припустити, що кількість помилок не перевищить 20). У такому разі на установку і налаштування системи відстеження помилок може витратитися більше ресурсів і часу, ніж на виправлення отриманих помилок. Тобто можна вважати, що використання системи відстеження помилок є недоцільним.

Але як можна оцінити кількість помилок в програмному забезпеченні? На жаль, поки немає загальної моделі оцінювання надійності ПЗ, яка дає точні оцінки надійності довільного програмного забезпечення. Тому в світовій практиці використовується декілька десятків різних моделей оцінки надійності. Кожна з них дає більш-менш прийнятні результати, всі мають своє найбільш раціональні сфери застосування.

Табл. 1. Критерії вибору систем відстеження помилок

Критерій	**Bugzilla**	**Trac**	**Redmine**
Ліцензія	Mozilla Public License	BSD variant	GPL v.2
Мова реалізації	Perl	Python	Ruby on Rails
Призначений для користувача інтерфейс	Web, e-mail, RSS, Web service, command-line	Web	Web, E-mail, Atom
Інтеграція з системами управління версіями	CVS, Subversion, Perforce, AccuRev	Subversion, Darcs, Bazaar, Mercurial, Perforce, Git	Subversion, CVS, Bazaar, Darcs, Mercurial, Git
Інтеграція або генерація динамічної документації	Інтегровані звіти і діаграми, планування розкладу звітів	Інтегрована wiki	Інтегрована wiki, дискусійні форуми, новинні блоги, email інтеграція, календарі
Інтеграція з плануванням тестів	Testopia	–	–
Workflow, що настроюється	Так	Так	Так
Підтримка юнікода	Так	Так	Так
LDAP авторизація	Так	–	Так

Математичні моделі по-різному підходять до оцінки надійності ПЗ, використовуючи для цього різні характеристики [4 – 5]:

- інтервали часу між відмовами (моделі Джелінського-Моранді, Шика-Волвертона, геометричні);
- внесені штучно до ПЗ помилки (модель Міллса);
- простір вхідних даних (модель Нельсона);
- складність ПЗ (модель Холстеда).

Скористаємося мірою складності Холстеда. Експерименти показують, що кількість помилок в непротестованих програмах пропорційна:

$$E_n = K \cdot E^{2/3}, \quad K = \frac{1}{3200},$$

де E_n – міра складності Холстеда, і визначається за формулою:

$$E = \frac{n_1 N_2 N \log_2 n}{2n_2},$$

де n_1 – кількість різних операторів;

n_2 – кількість різних операндів;

$n = n_1 + n_2$ – словник програми;

N_1 – кількість появ операторів;

N_2 – кількість появ операндів;

$N = N_1 + N_2$ – довжина програми.

Розрахуємо кількість помилок для малої програми (близько 1000 рядків), середньої (близько 100000 рядків) і великої програми (більше 1 млн. рядків).

Для малої програми $N_1 = 1000$, $N_2 = 2000$, $n_1 = 200$, $n_2 = 100$ – оцінна кількість складатиме 37 помилок.

Для середньої програми кількість помилок зросте до 2500 і для великої – ця кількість перевищить 20 тис. помилок.

Розглянувши ці випадки, можна зробити вивід, що людських здібностей може вистачити на роботу з помилками у разі малої програми, але у випадку з великими і середніми програмами вкрай необхідне впровадження і супровід системи відстеження помилок.

5. Висновок

Багато компаній знаходять, що інтегрована система відстеження помилок зменшує час простою, збільшує продуктивність і підвищує задоволеність клієнта від роботи з їх системами. Разом з повним розкриттям інформації, відкрита система відстеження помилок дозволяє виробникам програмного забезпечення підтримувати контакт зі своїми клієнтами і реселерами, для передачі повідомлень про помилки по всьому ланцюгу управління даними. Багато корпорацій також виявляють, що відстеження помилок зменшує витрати, забезпечуючи службі підтримки IT систему обліку, телефонній підтримці - бази знань, а всім - добре зрозумілу систему для обліку проблем з програмним забезпеченням.

Система відстеження помилок може бути легко адаптована до різних ситуацій. У зв'язці з такими системами як CVS або Subversion, система відстеження помилок забезпечує могутнє, легке у використанні рішення для управління конфігурацією і проблемами, що виникають при реплікації.

Вибір системи відстеження помилок необхідно робити, спираючись на урахування наступних особливостей системи:

- функції, які необхідні в роботі людини або організації (слід також враховувати розширюваність за допомогою додаткових модулів);
- зручність у використанні і обслуговуванні;
- швидкість роботи в конкретній конфігурації сервера і системи відстеження помилок.

Найбільш вірогідна ситуація, коли необхідно випробувати конкретну систему на тестовому комп'ютері, для затвердження цієї системи відстеження помилок або відмови від неї.

Перелік посилань

1. Канер С. Тестирование программного обеспечения. Фундаментальные концепции менеджмента бизнес-приложений / Канер С., Фолк Д., Нгуен Е.; пер. с англ. – К.: Издательство ДиаСофт, 2001. – 544 с.
2. The Bugzilla Guide (электронный ресурс). Режим доступа к ресурсу: https://bugzilla.mozilla.org/
3. Trac and Sunversion (электронный ресурс). Режим доступа к ресурсу: http://trac.edgewall.org/
4. Черноножкин С.К. Меры сложности программ (обзор). Системная информатика. / Черноножкин С.К. – Новосибирск: Наука, 1997. – Вып. 5: Архитектурные, формальные и программные модели. – 560 с.
5. Лаврищева Е.М. Методы и средства инженерии программного обеспечения / Е.М. Лаврищева, В.А. Петрухин. – М.: МФТИ, 2006. – 304 с.