

Лабораторна робота №3

Тема: Формування звітів про дефекти(баги).

Мета: Навчитися формувати звіти про дефекти для тестування програмного забезпечення.

Теоретичні відомості:

На даний момент дані про дефекти(склад, критичність) для аналізу щодо визначення якості тестування програмного забезпечення можливо отримати за допомогою систем відстеження дефектів(багтрекерів), які автоматизують процес занесення, збереження та обробки дефектів. Багтрекер – це прикладна програма, розроблена, щоб допомогти тестувальникам та програмістам відстежувати історію звітів про дефекти (баги) під час своєї роботи. Головний компонент багтрекера — база даних, що записує факти про відомі баги. Ці факти можуть включати час звіту про дефект, 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. Отже, багтрекер розглядається, як допоміжний інструмент взаємодії тестувальників та програмістів для обліку, контролю та обробки дефектів. Для аналізу було обрано найпоширеніші системи багтрекінгу, зокрема особливості надання ними можливості створення звіту про дефект та його зміст - наявність полів які визначають склад дефекту. Для початку розглянемо загальні функціональні можливості кожної з обраних систем багтрекінгу.

Система відстеження помилок Redmine.

Redmine – відкритий серверний веб-додаток для управління проектами та відстеження помилок. Redmine написаний на Ruby і являє собою додаток на основі розповсюдженого веб-фреймворку Ruby on Rails. Розповсюджується згідно GNU General Public License. Даний продукт надає наступні можливості: ведення декількох проектів; гнучка система доступу, заснована на ролях; система відслідковування помилок; діаграми Ганта та календар; ведення новин проекту, документів та управління файлами; оповіщення про зміни за допомогою RSS-потоків та електронної пошти; форуми для кожного проекту; облік тимчасових витрат; налаштування довільних полів для інцидентів, тимчасових витрат, проектів та користувачів; завдання є центральним поняттям всієї системи, що описує задачу, яку необхідно виконати; ролі визначаються гнучкою моделлю визначення прав доступу користувачів; кожне завдання має статус, що представляє собою окрему сутність з можливістю визначення прав на призначення статусу для різних ролей; завдання можуть бути взаємопов'язані: наприклад, одна задача є підзадачею іншої або передує їй; інтеграція з різними системами контролю версій (репозиторіями); відсутня можливість приховування внутрішньої переписки між програмістами від клієнта; система підтримує облік витраченого часу завдяки сутності «витрачений час», пов'язаної з користувачами і завданням; управління файлами і документами в Redmine зводиться до їх додавання, видалення та редагування; гнучка система конфігурування; протокол рішення помилок; відкритий вихідний код; низька вартість впровадження і технічного супроводу системи за рахунок відсутності витрат на придбання ліцензій; легка інтеграція з системами управління версіями (SVN, CVS, Git, Mercurial, Bazaar і Darcs); створення записів про помилки на основі отриманих листів; підтримка множинної аутентифікації LDAP; можливість самостійної реєстрації нових користувачів; багатомовний інтерфейс; підтримка СУБД MySQL, PostgreSQL, SQLite, Oracle. Розглянемо структуру баг-репорту системи Redmine. Структура баг-репорту системи Redmine має свої особливості, які проявляються у найменуванні окремих полів, що у значній мірі відрізняється від найменування, яке представлено у загальновідомій структурі звіту про знайдені дефекти. Причиною цього є особливості СВП Redmine, що полягають у структурі системи, її головних об'єктах та сутностях, які доцільно розглянути далі. Користувачі. Користувачі є одним з центральних понять предметної області. Модель користувача використовується для ідентифікації, аутентифікації, взаємодії з системою персоналу і клієнтів, а також для їх авторизації відповідно різних ролей, проектів. Ролі. Ролі визначаються гнучкою моделлю визначення прав доступу користувачів. Ролі включають в себе набір привілеїв, що дозволяють

розмежовувати доступ до різних функцій системи. Користувачам призначаються ролі у відповідних проектах, до яких вони залучені, наприклад «тестувальник у проєкті з розробки програми Х». Користувач може мати кілька ролей. Призначення ролей у межах окремого завдання здійснити неможливо. Проєкти. Є одним з основних понять у предметній області систем управління проєктами. Завдяки цій сутності можливо організувати спільну роботу та планування декількох проєктів одночасно з розмежуванням доступу різним користувачам. Проєкти допускають ієрархічну вкладеність. Трекери. Трекери є основною класифікацією, відповідно до якої сортуються завдання в проєкті. Саме по собі поняття «трекер» відноситься до систем обліку помилок. У «Redmine» трекери представляють собою аналог підкласів класу «Завдання» і є основою для поліморфізму різного роду завдань, дозволяючи визначати для кожного їх типу різні поля. Прикладами трекерів є «Поліпшення», «Помилка», «Документування», «Підтримка». Завдання. Завдання є центральним поняттям всієї системи, що описує задачу, яку необхідно виконати. Для кожного завдання повинні обов'язково існувати опис, автор та трекер, до якого відноситься завдання. Кожне завдання має статус, що представляє собою окрему сутність з можливістю визначення прав на призначення статусу для різних ролей (наприклад, статус «відхилено» може присвоїти тільки менеджер) або визначення актуальності завдання (наприклад, «відкритий», «призначений» - актуальні, а «закритий», «відхилено» - ні). Для кожного проєкту окремо визначаються набір етапів розробки і набір категорій завдань. Також у системі присутні такі поля як «час згідно оцінювання», що служить основою для побудови діаграм управління; поле вибору спостерігачів за завданням. Існує можливість додавання файлів з використанням поля «Додаток». Відстеження зміни статусу завдань. За відстеження змін параметрів завдань користувачами в системі відповідають дві сутності: «Запис журналу змін» і «Змінений параметр». Запис журналу відображає одну дію користувача по редагуванню параметрів задачі та додавання до неї коментаря (таким чином служить одночасно інструментом ведення історії завдання та інструментом ведення діалогу). Сутність «Змінений параметр» прив'язана до окремого запису журналу і призначена для зберігання старого і нового значення зміненого користувачем параметра. Зв'язки між завданнями. Завдання можуть бути взаємопов'язані: наприклад, одна задача є підзадачею іншої або передуює їй. Ця інформація може бути корисна в ході планування розробки програми (за її зберігання в Redmine відповідає окрема сутність). Облік витраченого на проєкт часу. Система підтримує облік витраченого часу завдяки сутності «витрачений час», пов'язаної з користувачами і завданням. Сутність дозволяє зберігати витрачений час, вид діяльності користувача (розробка, проектування, підтримка) і короткий коментар до роботи. Ці дані можуть бути використані, наприклад, для аналізу внеску кожного учасника в проєкт або для оцінки фактичної трудомісткості і вартості розробки. Прив'язка репозиторіїв. Redmine надає можливість інтеграції з різними системами контролю версій (репозиторіями). Інтеграція полягає у відстеженні змін у зовнішньому репозиторії, їх фіксації у базі даних, аналізі змін з метою їх прив'язки до певних завдань. У інфологічній структурі системи за інтеграцію із зовнішніми репозиторіями відповідають три сутності: «Репозиторій», «Редакція» і «Зміна». «Репозиторій» представляє собою пов'язану з проєктом сутність, що зберігає тип підключеного сховища, його місцезнаходження та ідентифікаційні дані користувача. «Редакція» є відображенням редакції сховища, і, крім інформаційних полів, може бути прив'язана до конкретного завдання і до користувача, автора редакції. Сутність «Зміна» призначена для зберігання списку змінених (доданих, віддалених, переміщених, модифікованих) файлів в кожній редакції. Отримання повідомлень. Повідомлення користувачів про зміни, що відбуваються на сайті, здійснюється з допомогою суті «Спостерігачі», що зв'язує користувачів з об'єктами різних класів (проєкти, завдання, форуми та ін.) У базі даних зберігаються також ключі доступу до підписки RSS, що дозволяють отримувати повідомлення за допомогою цієї технології. Також повідомлення розсилаються за допомогою електронної пошти. Окрім розглянутих особливостей Redmine, ця система має ряд недоліків: управління файлами і

документами в Redmine зводиться до їх додавання, видалення та редагування, правами доступу ні до файлів, ні до окремих документів управляти не можна; відсутні оповіщення про зміну документів; не можна керувати правами доступу на рівні окремих полів завдання (відсутня можливість приховування деталей розробки від замовника); існує можливість керування правами доступу на рівні проектів, але відсутня можливість призначення прав на відповідну версію проекту або окреме завдання (якщо користувачу потрібен доступ усього до однієї задачі, то доведеться давати доступ до всього проекту); відсутня можливість обмеження доступу відповідно типів завдань (не можна дозволити переглядати тільки власні завдання або дозволити створювати завдання тільки якогось певного типу); всі додаткові поля доступні всім користувачам, всі учасники проекту зможуть їх бачити і змінювати. Це обмеження може призвести до складностей при наявності неоднорідної команди, коли доступ до проекту мають і менеджери, і розробники, і клієнти; відсутні права на окремі типи конвертацій у workflow (наприклад, у поточний момент розробки не можливо вказати, що при закінченні виправлення помилки, розробник повинен вибрати відповідальним тестувальника та вказати номер білду); відсутня можливість приховування внутрішньої переписки між програмістами від клієнта.

Система відстеження помилок Jira.

Atlassian JIRA - комерційна система відстеження помилок, призначена для організації спілкування з користувачами, та управління проектами. Розроблена компанією Atlassian Software Systems . Має веб- інтерфейс . Назва системи (JIRA) отримано шляхом модифікації назви конкуруючого продукту - Bugzilla. JIRA створювалася в якості заміни Bugzilla і багато в чому повторює його архітектуру. Система дозволяє працювати з декількома проектами. Для кожного з проектів система дозволяє створити та супроводжувати схеми безпеки і оповіщення. Ключовими перевагами системи Jira є: веб-інтерфейс користувача; підтримка користувальницьких полів багатьох типів; підтримка налаштувань схеми робочого процесу; відкрита архітектура, API та безліч плагінів; значна комерційна підтримка.

Система відстеження помилок Bugzilla.

Bugzilla має веб-інтерфейс, початково створена і використана у проекті Mozilla. Система опублікована як відкрите програмне забезпечення компанією Netscape Communications в 1998 і прийнята багатьма організаціями для використання у якості багтрекеру при створенні програмних продуктів. Має ліцензію Mozilla Public License, яка надає продукту статус відкритого коду та вільного використання. Основним поняттям системи є баг — завдання, запит, рекламація стосовно помилки в системі, чи просто повідомлення, яке вимагає зворотного зв'язку. Bugzilla має наступні функціональні можливості: інтегрована система безпеки з можливістю її визначення на рівні продуктів; система залежностей помилок та можливість їх виведення у графічному вигляді; розвинена система складання звітів; гнучка система конфігурування; протокол рішення помилок; API для електронної пошти, XML, HTTP та консолі; доступна інтеграція з системами управління автоматичного конфігурування програмного забезпечення (Perforce, CVS).

Система відстеження помилок Trac.

Окрім надання функціональних можливостей для відстеження дефектів, система Trac також використовується у якості інструменту для управління проектами. Trac є відкритим програмним забезпеченням, розробленим і підтримуєним компанією Edgewall Software. Система використовує мінімалістичний веб-інтерфейс, заснований на технології Wiki, яка дозволяє організувати перехресні гіперпосилання між базою даних зареєстрованих помилок, системою управління версіями і вікі-сторінками. Це дає можливість використовувати Trac як веб-інтерфейс для доступу до системи контролю версій Subversion, а також, за допомогою використання Mercurial, Git, Bazaar. У системі підтримуються бази даних SQLite, PostgreSQL, MySQL та MariaDB. Trac написаний на мові програмування Python і в даний час поширюється за модифікованою ліцензією BSD. У перших версіях системи (до версії 0.11) в якості системи HTML-шаблонів веб-інтерфейсу

використовувався ClearSilver. Нові версії, використовують розроблену в Edgewall систему шаблонів Genshi. При цьому існує сумісність з плагінами, які використовують ClearSilver. Система Trac відрізняється мінімалістичною структурою звіту про дефект, де доступні тільки базові поля

Система відстеження помилок Bontq.

Bontq - веб-додаток для управління проектами та відстеження помилок. Bontq написаний на PHP та Java, основною його відмінністю від конкурентів є крос-платформенний клієнт, який може робити знімки екранів і записувати відео для складання візуальних звітів про помилки. На відміну від аналогічних систем, звіт про помилки у системі Bontq відрізняється гнучкістю щодо відображення певного набору полів відповідно до типу звіту, що додається: дефект, запит на додавання нової функціональності, задачі, нотаток, вікі- документ, тестовий сценарій. Bontq має сцєпіфічні функціональні можливостей відносно обробки дефектів. Глобальна активність. Існує можливість перегляду глобальної активності команди, що містить період часу відносно діяльності щодо усіх задач та проектів. Також існують 3 фільтра - за користувачами, за проектами та за елементами. Вони дають можливість надання більш детального запиту. Різноманітні типи завдань. Помилка - класичний дефект, який повинен бути закритий в найкоротший термін; запит відносно функціональності – пропозиція щодо привнесення нової функціональності до вже існуючої; нотаток – призначений для подальшої конвертації у будь-який інший тип завдань; завдання - призначається відповідній проектній групі (управління 41 проектами); тестовий сценарій – послідовність дій, що складає тестовий сценарій; вікі - окремий розділ для зберігання і управління документами та специфікаціями. Новий дефект Запит на додавання нової функціональності Нове завдання. Нова нотатка. Новий вікі-документ. Новий тестовий сценарій. Пріоритети та мітки, що визначаються користувачем. Можливість налаштування пріоритетів та міток, їх порядку, назви та кольорів окремих елементів. Оповіщення за допомогою електронної пошти. Оповіщення з приводу будь-який змін проєктах.

Система відстеження помилок Chili Project.

ChiliProject це веб-система управління проектами. Вона призначена для використання протягом усього життєвого циклу проекту, починаючи від створення плану, відстеження дефектів і закінчуючи системою спільного обміну знаннями. Розглянемо структуру звіту про дефекти, що надається системою ChiliProject. ChiliProject є форком проекту Redmine, тому загальна структура побудови звіту про помилки Redmine зберіглася у ChiliProject. Але ChiliProject має свої функціональні особливості. Планування проекту. Налаштування дорожньої мапи, що містить усі завдання та дедлайни. Застосування діаграми Ганта. Відстеження дефектів та звітність відносно ходу роботи. Автоматичне сповіщення про будь-які зміни у відповідних задачах. Модифікація задачі з використанням електронної пошти (за допомогою відправлення відповіді на лист оповіщення). Отримання даних про активність у проєкті у вигляді стрічки новин.

Система відстеження помилок Mantis.

MantisBT - вільна система відстеження помилок, яка забезпечує взаємодію розробників з користувачами та дозволяє користувачам створювати звіти про помилки та контролювати подальший процес їх опрацювання з боку розробників. Система представлена у вигляді веб- додатку. Баг-репорт системи Mantis відрізняється детальним представленням окремих елементів, зокрема загальноприйняте поле Description, яке зазвичай відображається одним полем, у Mantis представлено декількома: можливість відтворення (Reproducibility), опис (Description), кроки для відтворення (Steps to reproduce), додаткова інформація (Additional information). Система має гнучкі можливості конфігурації, що дозволяє її налаштовувати як для роботи над програмними продуктами так і у якості системи обліку заявок для Helpdesk.

Система відстеження помилок Bug Tracker .NET.

BugTracker .NET - це вільна реалізація системи баг-трекінгу на платформі .NET, яка є безкоштовною системою Free Software з відкритим вихідним кодом Open Source і розповсюджується під ліцензією GNU General Public License. Основними перевагами цієї системи є простота і зручність роботи, мінімізація використання програмування при налаштуванні. Відповідно, особливостями Bug Tracker.NET є: відкритий вихідний код; низька вартість впровадження і технічного супроводу системи за рахунок відсутності витрат на придбання ліцензій; повнофункціональний веб-інтерфейс; загальна інформаційна база для всіх модулів системи. Таким чином основними функціональними можливостями BugTracker.NET є: ведення списку проектів; інтеграція з електронною поштою; різноманітні механізми аутентифікації користувачів; обробка дефектів та завдань; гнучка можливість налаштування категорій для issue-трекінгу; система звітів; система Dashboard; можливість додавання до категорій нових полів. Більш детально слід зупинитися на можливості додавання до категорій нових полів. Це дозволяє самостійно додавати поля, для їх подальшого відображення у баг-репорті. Додавання полів здійснюється у режимі адміністратора, при цьому можливо задати наступні параметри: назва поля; тип елемента (звичайне поле, випадаючий список і т.д.); тип даних, що можуть міститися у полі; довжина поля; обов'язковість поля; значення по замовченню; значення для випадаючого списку; послідовність сортування. Таким чином, система відстеження помилок BugTracker.NET має значну відмінність, яка полягає у гнучкості, що досягається можливістю використання у баг-репортах самостійно створених полів.

Система відстеження помилок BugNET.

Це система багтрекінгу, що має відкритий вихідний код. BugNET випущений згідно ліцензії GPL та побудований на Microsoft ASP.NET 3.5 платформі. Система забезпечує підтримку декількох баз даних, зокрема MS SQL 2005, онлайн підтримку спільноти, підтримку декількох проектів. Також система має ряд особливостей для обробки задач, дефектів: блокування доступу до системи за необхідністю (надання привілейованого доступу виключно команді підтримки); можливість призначення крайнього терміну та надання оцінку часу на опрацювання дефекту; наявність користувальницьких полів для подальшого відображення у баг-репортах; можливість створення користувацьких статусів та пріоритетів з відповідними користувацькими графічними мітками; робота з електронною поштою; гнучка система утворення ідентифікаторів задач. Таким чином, система відстеження помилок BugNET має значну відмінність яка полягає у гнучкості, що досягається можливістю використання у баг-репортах як самостійно створених полів так і можливістю створення користуальницьких статусів та пріоритетів.

Приклад опису дефекту

Назва установи НУЛП

Defect ID #13

Короткий опис: Кнопка "Clear list" неактивна

Повний опис: "Clear list" кнопка є неактивною, коли в списку є лише один запис

Тип звіту: помилка кодування

Пропозиції по виправленню: зробити кнопку "Clear list" активною, шляхом додавання перевірки.

Стан: Відкрита

Резолюція: Розглядається

Кроки відтворення	Очікувані результати	Фактичні результати	Відтвореність	Степінь важливості	Пріоритет
1. додати один запис	1. запис доданий до списку	1. запис доданий до списку	так	серйозна	низький

“123” до списку 2. натиснут и кнопку “Clear list”	2. “Clear list” кнопка активна і список очищено	2. “Clear list” кнопка неактивна, список не очищено			
--	---	---	--	--	--

Додатки:

Product Page : DNR-322L Firmware Version: 2.40

D-Link

DNR-322L // LIVE VIDEO PLAYBACK **SETUP** MAINTENANCE STATUS SUPPORT

Wizard
Network Setup
Camera Search
Camera Setup
Audio and Video
2nd Stream Settings
Live & Playback Setup
Recording Schedule
Auto-Backup Setup
Event Setup
E-Mail Setup
Time and Date
Logout

CAMERA SETUP

To manually add a camera, enter its IP address.

Save Settings Don't Save Settings

CAMERA SETUP

Index **Camera 2**

Camera Name 5222

Address 192.168.0.14 Port 80

Administrator Name admin Password

Camera Channel 1 Protocol ☐ TCP ☐ UDP ☒ HTTP

Vendor D-Link Model DCS-5222LB

Auto Detection Clear

CAMERA LIST

Index	Camera Name	Address	Port	Vendor	Model
1	3714	192.168.0.2	80	D-Link	DCS-3714
2	5222	192.168.0.14	80	D-Link	DCS-5222LB
3			80	-- none --	-- none --
4			80	-- none --	-- none --
5			80	-- none --	-- none --
6			80	-- none --	-- none --

Helpful Hints..

The following list briefly describes the **Camera Setup** menu items:

Camera Channel: Select the number of analog cameras supported by one video server from the list.

Vendor: Displays the manufacturer's name.

Auto Detection: Click to automatically detect all the information (such as camera channel, protocol used, vendor, model number) of the camera.

Note: Some cameras will limit the login authority to the administrator only.

Можливі варіанти полів:

Тип звіту (1-6) ____	Степінь важливості (1-3) _____	Додатки (Так\ні)
1 - Помилка кодування 2 - Помилка проектування 3 - Пропозиція 4 - Розбіжності з документацією 5 - Взаємодія з апаратурою 6 – Питання	1 - фатальна 2 - серйозна 3 - незначна	Якщо так, які :

<i>Стан (1-2)_____</i>	<i>Пріоритет (1-3)_____</i>
<i>1 - Відкрита</i>	<i>1 - Високий</i>
<i>2 - Закрита</i>	<i>2 - Середній</i>
	<i>3 - Низький</i>

Резолюція (1-9)_____

<i>1 - Розглядається</i>	<i>5 - Відповідає проекту</i>
<i>2 - Виправлено</i>	<i>6 - Не може бути виправлено</i>
<i>3 - Не відтворюється</i>	<i>7 - Відкликано розробником</i>
<i>4 - Відкладено</i>	<i>8 - Потрібна додаткова інформація</i>
	<i>9 - Не згідний з пропозицією</i>

Практичне завдання:

Сформувати звіти про 10 дефектів (багів), що могли отриматись в результаті проходження тест-кейсів з лабораторної роботи 2.